2015

# ვებ ინტერფეისის დიზაინერი

სსიპ განათლების ხარისხის განვითარების ეროვნული ცენტრი

ავტორები: სერგო კარაკოზოვი, გიორგი კევლიშვილი, ეკატერინე ჩიკაშუა, ნიკოლოზ ელიზბარაშვილი

# შესავალი

წინამდებარე სახელმძღვანელო შედგენილია პროფესიული სტუდენტებისათვის და მოიცავს ყველა იმ თეორიულ და პრაქტიკულ საკითხებს, რაც აუცილებელია ვებ ინტერფეისის დიზაინერის პროფესიით დასაქმებისათვის.

რეცენზენტები:

**ხატია ლაპიაშვილი**
სს „აი თი დი სი“ ვებ დიზაინერი

**ნინო ლორთქიფანიძე**
კავკასიის უნივერსიტეტი - ტექნოლოგიების სკოლის ასისტენტი

# 1. რასტრული გამოსახულების შექმნა და დამუშავება

## 1.1. დოკუმენტის საჭირო პარამეტრებით შექმნა

პარაგრაფის შესაბამისი თემატიკა

- რასტრული გამოსახულება და მისი შედარება ვექტორულ გამოსახულებასთან
- ფერთა მოდელების განხილვა
- რასტრული გამოსახულების გრაფიკული ფორმატები
- გრაფიკული რედაქტორის ინტერფეისის ტიპების გაცნობა
- ფანჯრების განლაგების რეჟიმების მიმოხილვა
- გრაფიკული რედაქტორის პარამეტრების გაცნობა
- დოკუმენტის შესაბამისი ფორმატების გაცნობა

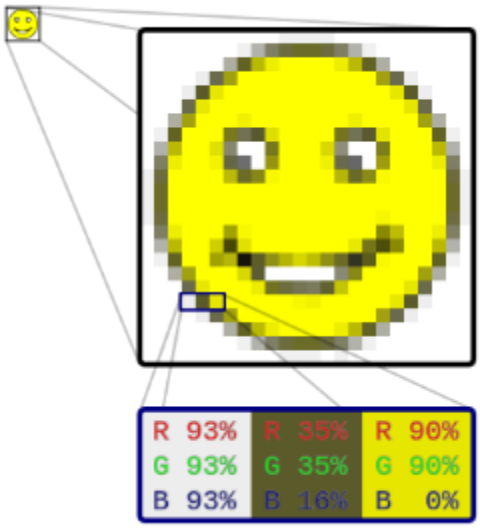
### რასტრული გამოსახულება

**რასტრი** (raster) - გრაფიკული გამოსახულება, რომელიც წერტილოვანი სტრუქტურის სახითაა წარმოდგენილი. როგორც წესი, წერტილს გააჩნია წრის ფორმა, თუმცა სხვა ფორმის წერტილებიც გამოიყენება: ელიფსური, რომბისებური და კვადრატული.

კომპიუტერულ გრაფიკაში რასტრული გამოსახულება იყოფა კვადრატულ ელემენტებად (ე. წ. პიქსელებად), რომელთა ერთობლიობა ქმნის გამოსახულებას ქაღალდზე, მონიტორზე ან სხვა რომელიმე ამსახველ მოწყობილობაზე. სიტყვა პიქსელი (pixel), ინგლისური Picture element – ის შემოკლებული ვარიანტია.

რასტრული გამოსახულების ძირითადი მახასიათებლებია:

- გამოსახულების ზომა პიქსელებში სიგანეზე და სიმაღლეზე (800x600px, 1024x768px და სხვა);
- გამოყენებული ფერების რაოდენობა ანუ ფერის სიღრმე;
- ფერთა მოდელები (RGB, CMYK, Lab და სხვა)
- გამოსახულების გარჩევადობა - სიდიდე, რომელიც განისაზღვრება წერტილების რაოდენობით მოცემულ ფართობზე, მაგალითად დუიმზე (1 დუიმი = 2.54 სმ).




სურ. 1.1-1. რასტრული გამოსახულების მაგალითი.

არსებობს გარჩევადობის ორი სახის საზომი ერთეული - ppi და dpi.

**PPI (pixel per inch)** - პიქსელების რაოდენობა დუიმზე. გამოიყენება ამსახველი მოწყობილობების გარჩევადობის შესაფასებლად: მონიტორი, ციფრული კამერის ეკრანი და სხვა;

**DPI (dot per inch)** - წერტილების რაოდენობა დუიმზე. გამოიყენება საბეჭდი მოწყობილობების ან სკანერების გარჩევადობის შესაფასებლად.

საბოლო ჯამში გარჩევადობა ნიშნავს ინფორმაციას. რაც უფრო მეტია გარჩევადობა, მით მეტია ინფორმაცია. რაც უფრო მეტი პიქსელია გამოსახულებაში, მით უფრო დიდ ფორმატზე შეიძლება იყოს ის აღბეჭდილი; რაც უფრო მჭიდროა პიქსელების განლაგება გამოსახულებაში, მით უფრო დეტალური და ხარისხიანია გამოსახულება.

ძირითადად გვხვდება შემდეგი ხარისხები:

72-150 pixel/inch - ამ ხარისხის გამოსახულებები ძირითადად ინტერნეტისთვის ან ელექტრონული პუბლიკაციებისთვის გამოიყენება;

150-250 pixel/inch - გამოიყენება ელექტრონული პუბლიკაციებში და პრინტერზე ბეჭდვისათვის;

300 pixel/inch და მეტი - გამოიყენება პოლიგრაფიაში და ბეჭდვით ტექნოლოგიებში. *შენიშვნა:* გამონაკლის შემთხვევაში, შესაძლებელია გამოყენებული იყოს მინიმუმ 250 pixel/inch ხარისხის გამოსახულება.

რასტრულ გრაფიკას გააჩნია როგორც დადებითი ისე უარყოფითი მხარეებიც.

დადებითი მხარეებია:

- რასტრულ გრაფიკაში შეიძლება ნებისმიერი სირთულის გამოსახულების შექმნა;
- პოპულარობა - რასტრული გრაფიკა პრაქტიკულად ყველგან გამოიყენება;
- რთული გამოსახულებების სწრაფად დამუშავება;

უარყოფითი მხარეები:

- მიღებული ფაილის დიდი მოცულობა;
- გამოსახულების ზომების გაზრდისას მისი ხარისხი მკვეთრად ეცემა.

## ვექტორული გამოსახულება

ვექტორულ გრაფიკაში გამოსახულება წარმოდგენილია მათემატიკური ობიექტების, კონტურების სახით, რომლებშიც ჩასხმულია ფერი. თავის მხრივ, კონტური შედგება საკვანძო წერტილებისგან, რომლებიც ერთმანეთს უკავშირდებიან მრუდეებით. საკვანძო წერტილებზე მოქმედებისას ჩვენ შეგვიძლია ვცვალოთ მრუდის მოხაზულობა და საერთო ჯამში ფიგურის ან ფორმის იერსახე.

ვექტორულ გამოსახულებას ისევე, როგორც რასტრულს, თავისი დადებითი და უარყოფითი მხარეები გააჩნია.

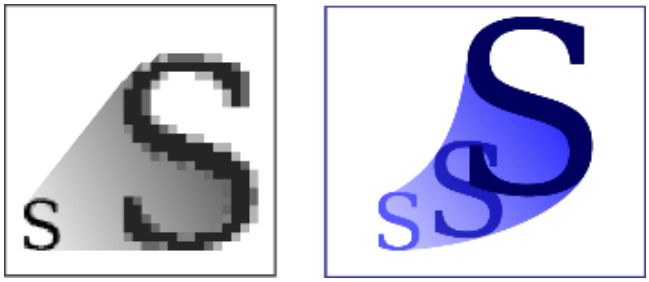
დადებითი მხარეებიდან შეგვიძლია გამოვყოთ:

- ფაილის მცირე ზომა;
- შესაძლებელია გამოსახულების გადიდება უსასრულოდ. მაგალითად თუ გავზრდით წირის მრუდეს, ის ისეთივე სუფთა დარჩება, როგორიც იყო; თუ მრუდე წარმოდგენილია ტეხილი ხაზით (რაც მცირე ზომისას არ ჩანს), გადიდებისას ის აუცილებლად გამოჩნდება;

უარყოფითი მხარეები:

- შეუძლებელია ყველა ობიექტი მარტივად გამოისახოს ვექტორული სახით; იმისათვის, რომ ვექტორული გამოსახულება დაემსგავსოს ორიგინალს, შესაძლებელია დაგჭირდეთ ძალიან დიდი რაოდენობის ვექტორები;

- ვექტორული გამოსახულების გარდაქმნა რასტრულად ძალიან მარტივია. ხოლო უკან დაბრუნება (რასტრულიდან ვექტორისკენ) საკმაოდ რთული;



სურ. 1.1-2. რასტრული და ვექტორული გამოსახულებების შედარება

### ფერთა მოდელები

ცნობილია, რომ მზის სხივი შეიცავს ბუნებაში არსებულ ყველა ფერს. მზის სხივის უწყვეტ სპექტრში არჩევენ 130-მდე ფერის ტონალობას. რეალურ ცხოვრებაში ამ ფერების ნახვა ცისარტყელაზე შეიძლება. მზის სხივის უწყვეტ სპექტრში ნიუტონმა გამოყო 7 ძირითადი ფერი: წითელი, ნარინჯისფერი, ყვითელი, მწვანე, ცისფერი, ლურჯი და იისფერი. ყველა სხვა ფერი, სწორედ ამ 7 ფერის სხვადასხვა რაოდენობით შერევის გზით მიიღება.

მრავალი ცდისა და დაკვირვების შედეგად, საერთაშორისო კოლორიმეტრული სისტემის მთავარი ფერები გამოავლინეს: წითელი (Red), მწვანე (Green) და ლურჯი (Blue). ინგლისური დასახელების პირველი ასოების მიხედვით, ფერთა ამ სისტემას დაერქვა - RGB.

RGB ფერთა სისტემა საფუძვლად უდევს მონიტორებს, სკანერების ერთ ნაწილს და კომპიუტერული პროგრამების უმრავლესობას. კომპიუტერის ეკრანზე ყველა ფერის მიღება, სწორედ ამ სამი ფერის სხვადასხვა პროპორციის შერევით არის შესაძლებელი. RGB მოდელის გარდა, არის ფერთა სხვა მოდელებიც: HSB; Lab; CMY; CMKY; YIQ და YCC.

მათ შორის ძირითადი მოდელები გამოიყენება: RGB – კომპიუტერულ გრაფიკაში; CMYK – პოლიგრაფიაში; Lab – ტექნიკური მიზნებისთვის; Grayscale – შავ-თეთრი ფოტო-სურათების რეჟიმი; Bitmap (Monochrome) – ორფეროვანი გამოსახულების რეჟიმი.

ფერთა მოდელის გარდა ასევე უნდა განისაზღვროს ფერის სიღრმე, რომელიც იზომება ბიტებში. რაც უფრო დიდია ფერის სიღრმე ე. ი. რაც უფრო მეტ ბიტიანია გამოსახულება, მით უკეთესად გადმოსცემს პიქსელი ფერს. პიქსელი გრაფიკული გამოსახულების უმცირესი ელემენტია და მისი დანიშნულება არის შეინახოს და გადმოსცეს ინფორმაცია ფერის შესახებ.

გამოსახულებაზე, რომლის ფერის სიღრმე 1 ბიტია ($2^1$=2), არის მხოლოდ ორი ფერი: თეთრი და შავი. 2 ბიტიან ($2^2$=4) გამოსახულებაზე ოთხი ფერია - თეთრი, შავი და ორი ნაცრისფერი. შავ-თეთრი ფოტოსურათების ფერის სიღრმე, უმეტეს შემთხვევაში, 8 ბიტია ($2^8$=256): შავი, თეთრი და 254 ნაცრისფრის სხვადასხვა ტონალობა. ე. ი. ამ შემთხვევაში პიქსელს შეუძლია გადმოსცეს ფერის 256 შესაძლო ვარიანტიდან ერთი. ნულოვან მნიშვნელობას შეესაბამება შავი ფერი, ხოლო 255-ს – თეთრი.

ფერად ციფრულ ფოტოსურათს აქვს 3 არხი (წითელი, მწვანე, ლურჯი) და 8 ბიტიანი ფერის სიღრმე თითო არხზე. ამიტომ, ასეთ გამოსახულებას 24 ბიტიანს ($2^{24}$=16777216) უწოდებენ და ამ შემთხვევაში, პიქსელს ფერის გადმოცემის 16,7 მილიონი შესაძლო ვარიანტი არსებობს. ფერის ეს რაოდენობა სავსებით საკმარისია გამოსახულების ფერის რეალურად გადმოცემისათვის, თუმცა არსებობს 48 ბიტიანი გამოსახულებები (16 ბიტი თითო არხზე), რაც 281 ტრილიონი ფერის გადმოცემის შესაძლო ვარიანტს იძლევა. ადამიანის თვალი ამდენ ფერს ვერ არჩევს, მაგრამ გამოსახულების რედაქტირებისას, ბიტების დიდი რაოდენობა შედეგზე დადებითად აისახება. 16 ბიტიანი გამოსახულებით პროფესიონალი ფოტოგრაფები მუშაობენ.

დღესდღეობით ფოტოშოპს შეუძლია 8, 16 და 32 ბიტიან გამოსახულებებთან მუშაობა.

### რასტრული გამოსახულების ფაილების ფორმატები

რასტრული გამოსახულების შესანახად ბევრი ფორმატი გამოიყენება. მუშაობის პროცესში თქვენ მოგიწევთ შეარჩიოთ, რომელ სასურველ ფორმატში გჭირდებათ გამოსახულების შენახვა. ეს შეიძლება ორ ეტაპად დავყოთ:

- **დაუმუშავებელი (ორიგინალი)** ფაილების შენახვა. ეს შესაძლოა იყოს ფოტოსურათები, მოძიებული ან შემენილი ფოტომასალა ფოტობანკებიდან, სკანირებული გამოსახულებები და სხვა. ეს მასალა მომავალში შესაძლოა გამოყენებული იყოს განმეორებით მსგავსი პროექტებისთვის.
- **დამუშავებული** ფაილების შენახვა პროექტისთვის. პროექტის დანიშნულების გათვალისწინებით (ბეჭდური თუ ელექტრონული), ფაილებისთვის შესაბამისი ფორმატის შერჩევა დაგჭირდებათ.

ქვემოთ ჩამოთვლილია ფაილების ფორმატები, რომელიც მუშაობის პროცესში შეგიძლიათ გამოიყენოთ:

**JPEG (Join Photographic Expert Group)** - ამ ფორმატის უპირატესობა არის მისი მოქნილობა. გამოსახულების რედაქტირების ყველა პროგრამა მუშაობს ამ ფორმატთან, შენახული ფაილების მოცულობა მცირეა, რადგან შენახვისას გამოიყენება *შეკუმშვა*. შეკუმშვისას გარკვეული ინფორმაცია *იკარგება* - ხდება ცალკეული პიქსელების ფერის გაშუალება, რის შედეგადაც მცირდება დეტალიზაცია და იცვლება ფერის ტონალობები. რაც უფრო მაღალია შეკუმშვის ხარისხი, მით უფრო მცირეა ფაილის მოცულობა, მაგრამ ამავდროულად გამოსახულების ხარისხი მკვეთრად ფუჭდება; და პირიქით, რაც უფრო მცირეა შეკუმშვის ხარისხი, ფაილის ზომა არ მცირდება და გამოსახულების ხარისხის მეტნაკლებად მისაღები ხდება. ერთი და იმავე ფაილზე მუშაობისას, ყოველ ახალ შენახვაზე ხდება ინფორმაციის მორიგი პორციის დაკარგვა. ამის გამო ამ ფორმატის გამოყენება ბეჭდვითი პროექტებისთვის მიუღებელია. მისი გამოყენება მისაღებია ელექტრონული პუბლიკაციებისთვის, ელფოსტით გასაგზავნად ან ინტერნეტში განსათავსებლად.

**GIF (Graphics Interchange Format)** - ფორმატი შექმნილია სპეციალურად რასტრული გამოსახულების გადასაცემად გლობალურ ქსელში. ორიენტირებულია კომპაქტურობაზე და იყენებს LZW (Lempel – Ziw - Welch) ინფორმაციის დაკარგვის გარეშე შეკუმშვის ალგორითმს, რომელიც უზრუნველყოფს შეკუმშვის ძალიან მაღალ ხარისხს. ამ ფორმატს შეუძლია ანიმირებული გამოსახულების შენახვა (ფაილში ინახება არა მარტო გამოსახულების კადრები, არამედ მისი დემონსტრაციის პარამეტრებიც). ფორმატის ნაკლად შეიძლება ჩაითვალოს, რომ ის მუშაობს მხოლოდ ინდექსირებულ ფერებთან.

**PNG (Portable Network Graphics)** - GIF ფორმატის ჩასანაცვლებლად შეიქმნა. მისგან განსხვავებით, მუშაობს ფერების დიდ რაოდენობასთან და ასვე შეუძლია შეინახოს ფენის გამჭვირვალობა. მასში გამოყენებულია ისეთი მძლავრი უდანაკარგო შეკუმშვის ალგორითმი, როგორიცაა LZW. GIF და PNG ფორმატში შენახული გამოსახულებები ძირითადად გამოიყენება ინტერნეტში განსათავსებლად და ელექტრონულ პუბლიკაციებში.

**BMP (BitMap)** - ფორმატი შექმნილია Microsoft-ის მიერ და ორიენტირებულია Windows-ის ოპერაციულ სისტემაში გამოსაყენებლად. იხმარება რასტრული გამოსახულების წარმოსადგენად პროგრამულ რესურსებში. იყენებს ინფორმაციის დაკარგვის გარეშე შეკუმშვის მარტივ ალგორითმს RLE (Run Length Encoding).

**TIFF** - ამ ფორმატს აქვს მრავალფენიანი დოკუმენტების მხარდაჭერა; გამოსახულება შეუძლია შეინახოს, როგორც შეკუმშვით (LZW ან ZIP), ისე შეკუმშვის გარეშე და მისი ხარისხი არ ფუჭდება. ამიტომ ეს ფორმატი შესაძლოა გამოდგეს თქვენი ნამუშევრების არქივში შესანახად. თუმცა მასთან მუშაობისათვის თქვენ დაგჭირდებათ პროგრამა Photoshop ან რომელი მსგავსი რედაქტორი. უნდა გახსოვდეთ, რომ შენახული ფაილის მოცულობა ბევრად აღემატება JPEG ფაილის მოცულობას.

**PSD (PhotoShop Document)** - ფორმატი არის Adobe PhotoShop-ის საკუთარი ფორმატი. ერთადერთი ფორმატი, რომელიც ინახავს ყველაფერს ნებისმიერი სახით და პროგრამის ყველა მოთხოვნას უჭერს მხარს. შენახვის დროს ინარჩუნებს დოკუმენტის ფენოვან სტრუქტურას, რათა მარტივი იყოს გამოსახულების შემდგომი რედაქტირება. ამ ფორმატში შეიძლება შუალედური ნამუშევრის შენახვა რედაქტირების დასრულებამდე.

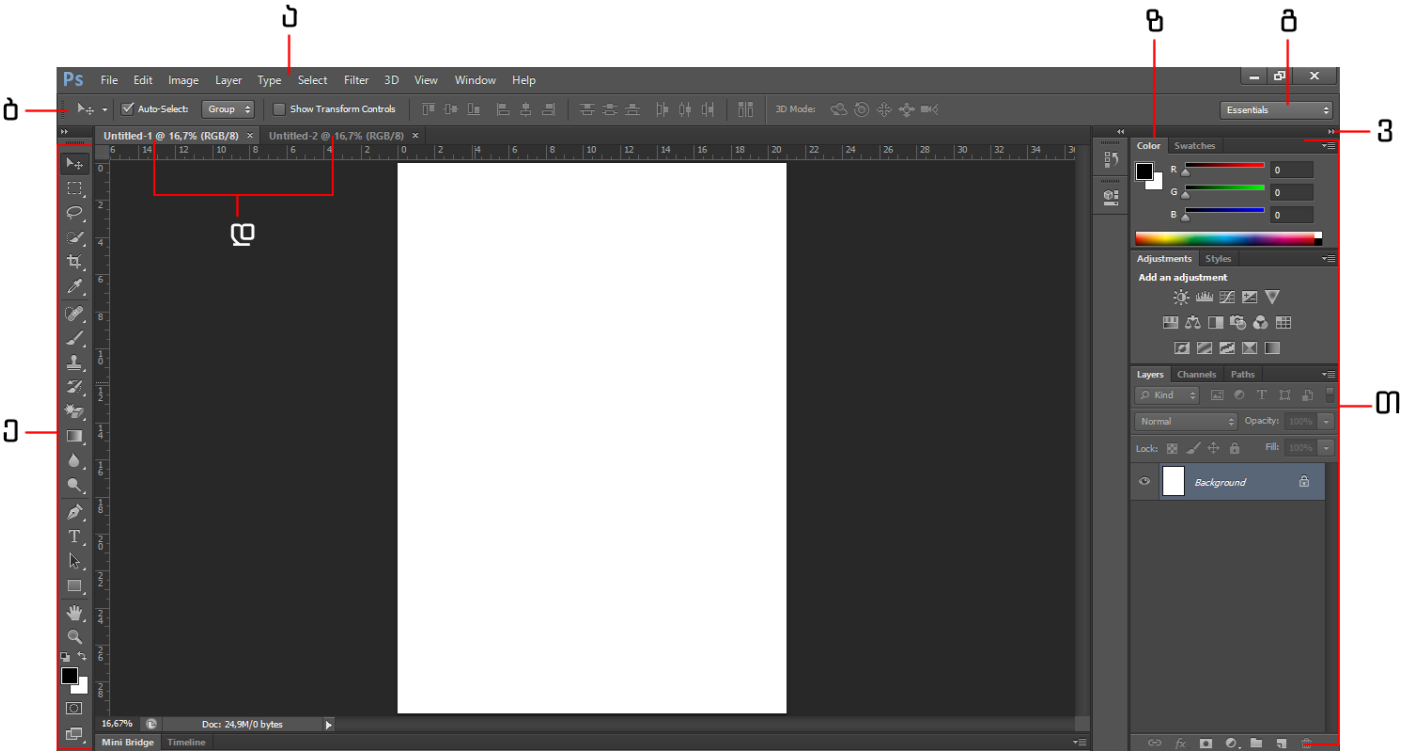
**RAW** - ეს არ არის რომელიმე კონკრეტული ფაილის ფორმატი. ის ზოგადად აღნიშნავს ე. წ. ნედლ, დაუმუშავებელ მონაცემებს, რომელსაც ფოტოკამერის სენსორი აფიქსირებს. ფოტოკამერების მწარმოებლებს ნედლი ინფორმაციის შესანახად საკუთარი ფორმატები აქვთ შემუშავებული. მაგალითად Canon-ს კამერები იყენებენ **CR2** გაფართოებას, ხოლო Nikon-ის კამერები **NEF**-ს. ასეთი ტიპის ფაილებთან სამუშაო შეგიძლიათ ისარგებლოთ პროგრამებით Photoshop, Bridge, Lightroom და დამუშავების შემდეგ ფაილი თქვენთვის სასურველ ფორმატში შეინახოთ (**JPEG**, **PNG**, **TIFF**, **PSD**). კომპანია Adobe-მა ასევე შეიმუშავა ფორმატი **DNG (Digital Negative)**, რომლის მიზანია მსგავსი დაუმუშავებელი მონაცემების შენახვა.

**EPS (Encapsulate PostScipt)** - ფაილების ფორმატი, რომელიც დაფუძნებულია PostSript[1]-ის ენაზე და განკუთვნილია პროგრამებს შორის გრაფიკული ინფორმაციის გასაცვლელად. ფორმატი ფართოდ გამოიყენება პოლიგრაფიაში და შეიძლება შეიცავდეს როგორც რასტრულ, ისე ვექტორულ გამოსახულებებს ან ორივეს ერთად. EPS ფორმატს ასევე გააჩნია Grayscale, RGB, CMYK და Lab ფერთა მოდელების მხარდაჭერა.

**PDF (Portable Document Format)** - ელექტრონული დოკუმენტების ფორმატი, რომელიც კომპანია Adobe System-მა შეიმუშავა. მისი პირველადი დანიშნულებაა პოლიგრაფიული პროდუქციის წარმოდგენა ელექტრონული სახით. PDF საშუალებას იძლევა ფაილში ჩაშენებული იყოს საჭირო შრიფტები, ვექტორული და რასტრული გამოსახულებები, ბმულები, ფორმები, მულტიმედია (აუდიო/ვიდეო) კონტენტი. გააჩნია Grayscale, RGB, CMYK და Lab ფერთა მოდელების მხარდაჭერა და პოლიგრაფიისათვის საჭირო საკუთარი ტექნიკური ფორმატები, მაგალითად PDF/X-1a, PDF/X-3.

[1] PostSript - გვერდების აღწერის ენა. ძირითადად გამოიყენება სამაგიდო საგამომცემლო სისტემებში.

## პროგრამა Adobe Photoshop-ის ინტერფეისი




სურ. 1.1-3. ა - პროგრამის მენიუ; ბ - ფორმატირების ზოლი. მისი პარამეტრები იცვლება იმის მიხედვით, თუ რომელი სამუშაო ინსტრუმენტი გაქვთ არჩეული; გ - სამუშაოო სივრცეებს შორის გადამრთველი; დ - სანიშნეები, რომლებიც მიუთითებენ გახსნილ ფაილს. მათი საშუალებით შესაძლებელია ფაილებს შორის გადასვლა; ე - ინსტრუმენტთა პანელი; ვ - სამუშაო პანელის ნიშნულის სახით ჩაკეცვის ღილაკი; ზ - სამუშაო პანელის დასახელება; თ - ვერტიკალურად ჩამაგრებული სამუშაო პანელები.

### სამუშაო სივრცე

თქვენი სამუშაო პროფილის ან კონკრეტული ამოცანის სპეციფიკიდან გამომდინარე, შესაძლოა დაგჭირდეთ სხვადასხვა ტიპის სამუშაო პანელები. გარდა იმ სამუშაო პანელებისა, რომელსაც სურათზე ხედავთ (სურ. 1.1-3, თ), შესაძლებელია კიდევ სხვების გამოჩენაც. ამისათვის უნდა შეხვიდეთ პროგრამის მენიუში Window და ჩამოშლილი სიიდან აირჩიოთ თქვენთვის სასურველი პანელი ან დამალოთ უსარგებლო.

პანელები შეგიძლიათ დააჯგუფოთ, შეურჩიოთ მას პოზიცია და დააყენოთ, როგორ უნდა ჩანდეს ის ეკრანზე (ღილაკის სახით თუ გაშლილ მდგომარეობაში) ანუ მოაწყოთ სამუშაო სივრცე ისე, როგორც თქვენ გჭირდებათ.

ფოტოშოპს გააჩნია რამდენიმე მზა სამუშაო სივრცე. სასურველი სამუშაო სივრცის არჩევისას ხდება ინტერფეისის ნაწილობრივ შეცვლა და ამ სივრცისთვის დამახასიათებელი სამუშაო პანელების გამოტანა. გარდა ამისა ეს მოცემული სამუშაო სივრცე შეგიძლიათ გადააწყოთ თქვენი სურვილისებრ.

**Essentials** – ძირითადი სამუშაო სივრცე. ამ დროს გამოტანილია ძირითადი სამუშაო პანელები, როგორიცაა ფერების პალიტრა და ფერის ნიმუშები, ჩვეულებრივი და კორექტირების ფენებთან სამუშაო პანელები და სხვა.

**New in CS6** - მათთვის, ვინც გადმოერთო ფოტოშოპის უფრო ძველი ვერსიიდან (CS4, CS5), სასარგებლო იქნება გაიგონ, რა სიახლეებია დამატებული პროგრამაში. გამოტანილი იქნება ინსტრუმენტთა პანელების დიდი ნაწილი, ასევე მენიუში ახალი ან განახლებული ფუნქცია ფერით იქნება მონიშნული.

**3D** - ამ დროს გამოტანილი იქნება ის პანელები და ინსტრუმენტები, რომლებიც სამგანზომილებიან გრაფიკასთან სამუშაოდ გამოიყენება.

**Motion** - სივრცე, რომელიც განკუთვნილია მათთვის ვინც ვიდეო ან ანიმაციასთან მუშაობს.

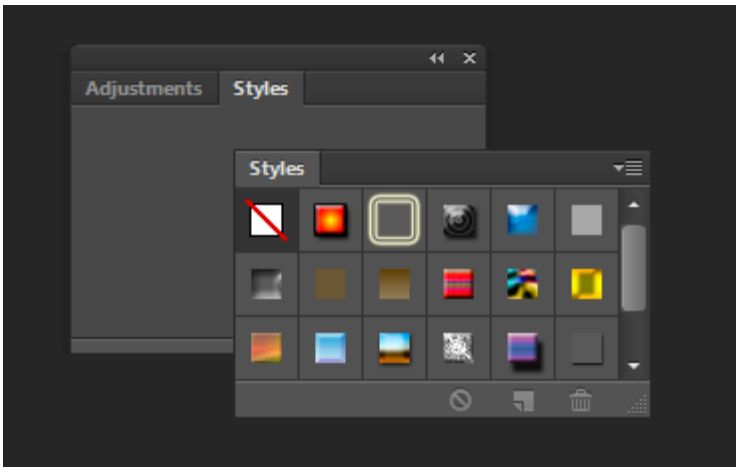
**Painting** - სივრცე, რომელიც გამოადგება მხატვარს ან ილუსტრატორს. აქ გამოტანილია ფერების პალიტრა, მზა ფუნჯები და ფუნჯების რეგულირების ფანჯარა

**Photography** - ფოტომასალის დასამუშავებლად გამზადებული სამუშაო სივრცე. გამოტანილია ის აუცილებელი ინსტრუმენტები, რომლებიც მომხმარებელს დაეხმარება ფოტოს დამუშავებაში: ფერთა კორექცია, ფენები, ჰისტოგრამა და სხვა.

**Typography** - სამუშაო სივრცე მათთვის, ვინც აპირებს იმუშაოს შრიფტების გამოყენებით და შექმნას ტიპოგრაფიული ეფექტები პოსტერებისთვის, ბანერისთვის და სხვა მსგავსი სახის პროექტისთვის.

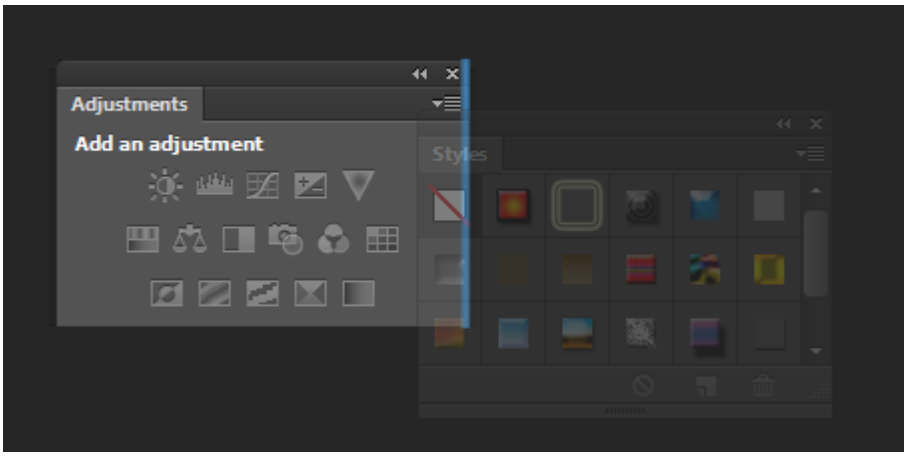
სამუშაო პანელებთან რომ გაგიადვილეთ მუშაობა, შესაძლებელია მათი დაჯგუფება. მაგალითისთვის თუ შეხედავთ გამოსახულებას (სურ. 1.1-3, თ), დაინახავთ რომ პანელები Layers, Channels და Path ერთად არიან დაჯგუფებული. შესაბამის დასახელებაზე დაჭერისას ხდება პანელებს შორის გადართვა და მისი ფუნქციების გამოჩენა. ასევე შეგიძლიათ უკვე არსებული ჯგუფების დაშლა და პანელების გადალაგება თქვენი სურვილის მიხედვით.

ჯგუფიდან რომ მოხსნათ პანელი, ამისათვის მაუსის კურსორით მოკიდეთ და გაიტანეთ ცალკე:

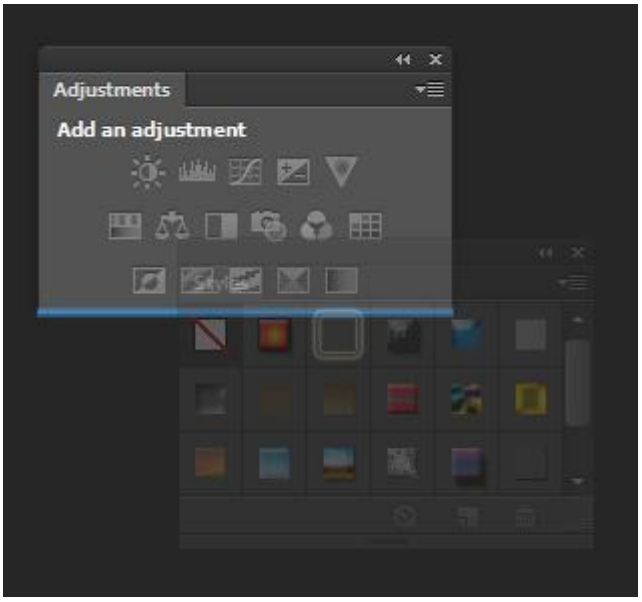



სურ. 1.1-4. პანელების ჯგუფის დაშლა

იმისათვის, რომ პანელები ჰორიზონტალურად ან ვერტიკალურად ჩაამაგროთ, პანელი მიიტანეთ მეორე პანელის კიდესთან (გვერდიდან ან ზემოდან/ქვემოდან). გამოჩნდება ლურჯი ზოლი, რაც კავშირის მაჩვენებელია:

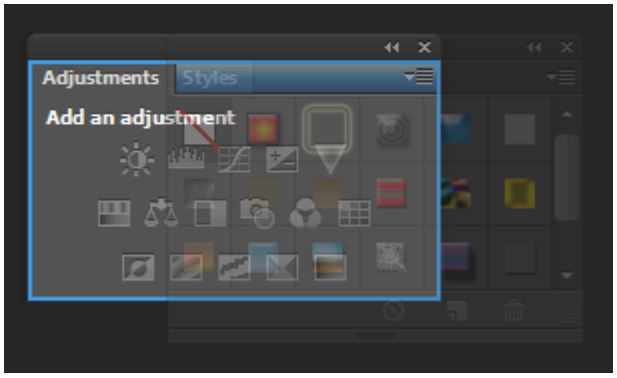



სურ. 1.1-5. პანელების ჰორიზონტალურად დაჯგუფება




სურ. 1.1-6. პანელების ვერტიკალურად დაჯგუფება

იმისათვის, რომ დააჯგუფოთ პანელები (გვერდიგვერდ), პანელი მიიტანეთ მეორე პანელის დასახელებასთან ახლოს:

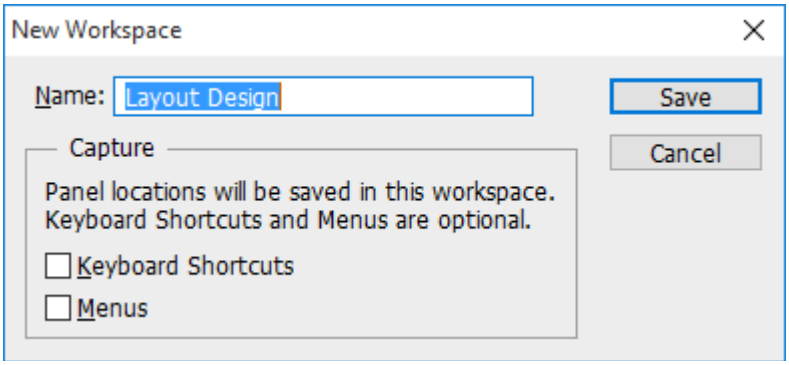



სურ. 1.1-7. პანელების დაჯგუფება

მას შემდეგ, რაც თქვენ დაალაგებთ საჭირო პანელებს, სასურველი იქნება ეს განლაგება შეინახოთ, რადგან სხვა სამუშაო სივრცეზე გადართვის შემდეგ მოგიწევთ ყველა პროცედურის (დაჯგუფების) ხელახლა გავლა.

ამისათვის შედით Window -> Workspaces. აქ თქვენ ნახავთ სამუშაო სივრცის განამზადებს და დამატებით სამ ფუნქციას.

იმისათვის, რომ შეინახოთ თქვენს მიერ დაყენებული სამუშაო სივრცე, აირჩიეთ Window -> Workspaces -> New Workspace (ახალი სამუშაო სივრცე).




სურ. 1.1-8. ახალი სამუშაო სივრცის შენახვის ფანჯარა

გამოსულ ფანჯარაში შეიყვანეთ სასურველი სახელი და დააჭირეთ OK. თქვენს მიერ შექმნილი სივრცე დაემატება სამუშაო სივრცეების სიაში.

მუშაობის პროცესში შესაძლოა დაგჭირდეთ სხვა პანელის გამოძახება, არსებულის დახურვა, ფანჯრების გადაჯგუფება და სხვა. რომ დაბრუნდეთ სამუშაო სივრცის საწყის განლაგებაზე, შედით Window -> Workspaces -> Reset Workspace (სამუშაო სივრცის გადატვირთვა). სამუშაო სივრცე აღდგება იმ სახით, რა სახითაც თქვენ შეინახეთ.

არასასურველი სამუშაო სივრცის წასაშლელად აირჩიეთ Window -> Workspaces -> Delete Workspace (სამუშაო სივრცის წაშლა). ჩამოსაშლელი მენიუდან აირჩიეთ სამუშაო სივრცე და დააწექით Delete.

ცალკე ყურადღება უნდა დაეთმოს ინსტრუმენტთა პანელს (სურ. 3: ე), რომელზეც ყველა საჭირო ინსტრუმენტია განთავსებული. ყოველი ინსტრუმენტი შეიცავს დამატებით ფუნქციებს, რომელთა მართვა შესაძლებელია ფორმატირების ზოლიდან (სურ. 3: ბ) ან მართვის პანელის (სურ. 3: თ) დახმარებით.

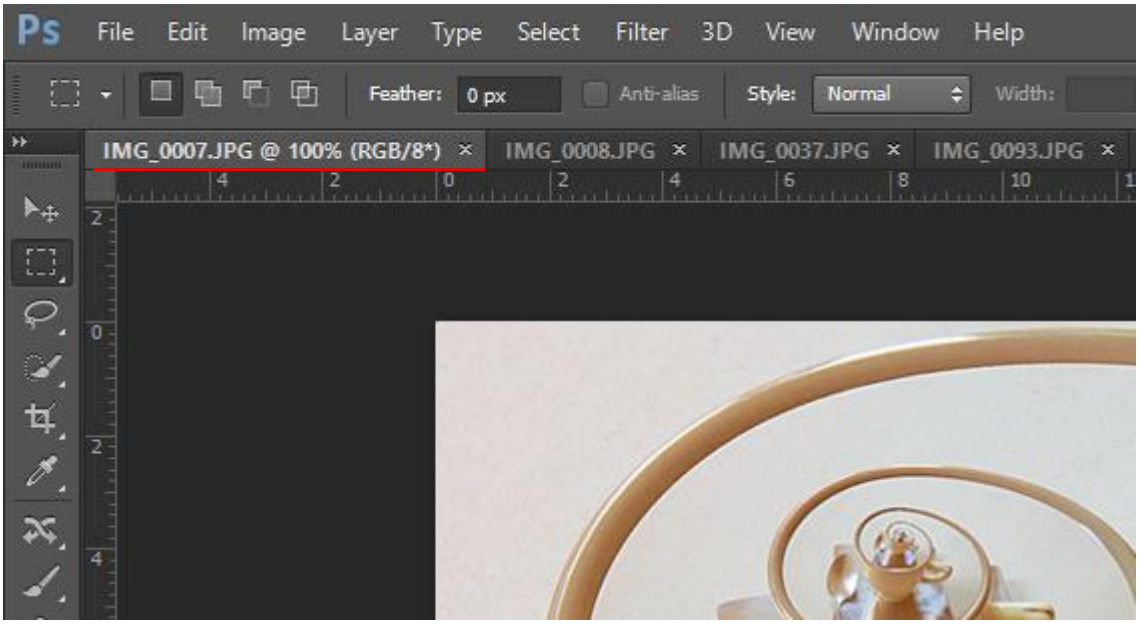
## ინსტრუმენტთა პანელი


სურ. 1.1-9. ინსტრუმენტთა პანელი და მისი მიმოხილვა

## ფანჯრების განლაგება

ფოტოშოპში შესაძლებელია ერთდროულად რამდენიმე ფაილის გახსნა და მათთან მუშაობა. ეს გაძლევთ საშუალებას ფენების, ობიექტების ან რაიმე ელემენტის ერთი ფაილიდან მეორეში მარტივად, უბრალო გადათრევით გადაიტანოთ, მოახდინოთ გამოსახულებების შედარება ან უბრალოდ რამდენიმე მოცემული კადრიდან გსურთ აარჩიოთ საუკეთესო პროექტში გამოსაყენებლად.

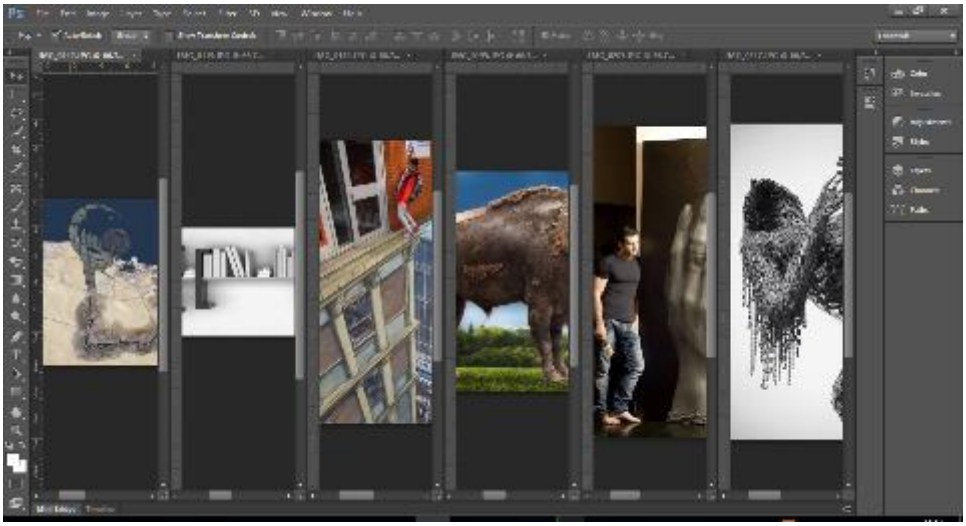
უნდა გაითვალისწინოთ, რომ თითოეული ფაილი დამოუკიდებელ ფანჯარაში იხსნება.



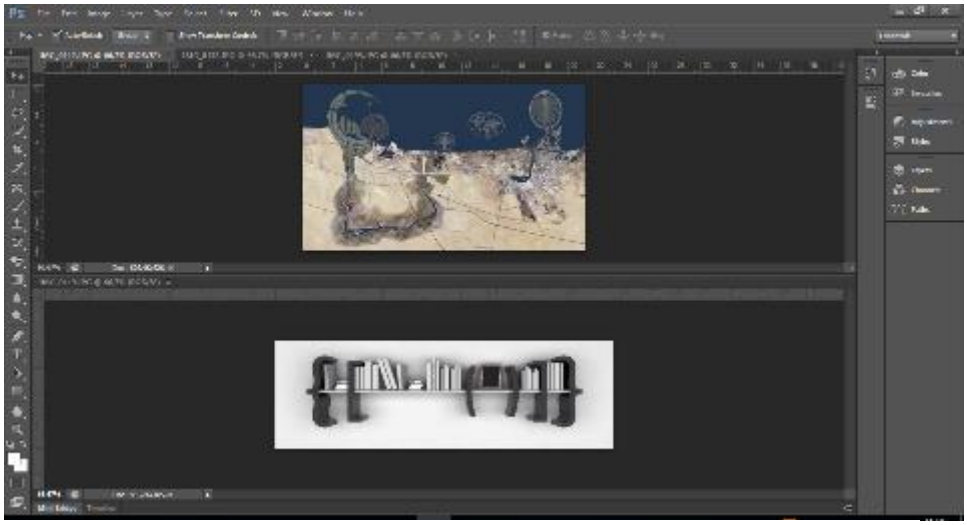

სურ. 1.1-10 ფანჯრის დასახელება ემთხვევა ფაილის სახელწოდებას. ეს გეხმარებათ დაინახოთ, რომელი ფაილები გაქვთ გახსნილი. ასევე აქ არის სხვა სახის ინფორმაციაც: IMG_0007.JPG - ფაილის სახელწოდება; @ - გამყოფი ნიშანი; 100% - გვაჩვენებს გამოსახულების სამუშაო მასშტაბს; RGB - გვაჩვენებს რა ფერთა მოდელი აღწერს გამოსახულებას; 8/16/32 - გვაჩვენებს რამდენ ბიტიანი ფერებია გამოსახულებაში; * - მიუთითებს, რომ ფაილში შეტანილია ცვლილებები.

სტანდარტულად პროგრამაში ფანჯრების განლაგება არის ჩანართის სახით (სურ. 3: დ). თუმცა ჩვენ მისი ცვლა შეგვიძლია და გამოვაჩინოთ გახსნილი ფაილები ისე, რომ გაგვიადვილდეს მათთან მუშაობა ან მთლიანობაში ინფორმაციის აღქმა.

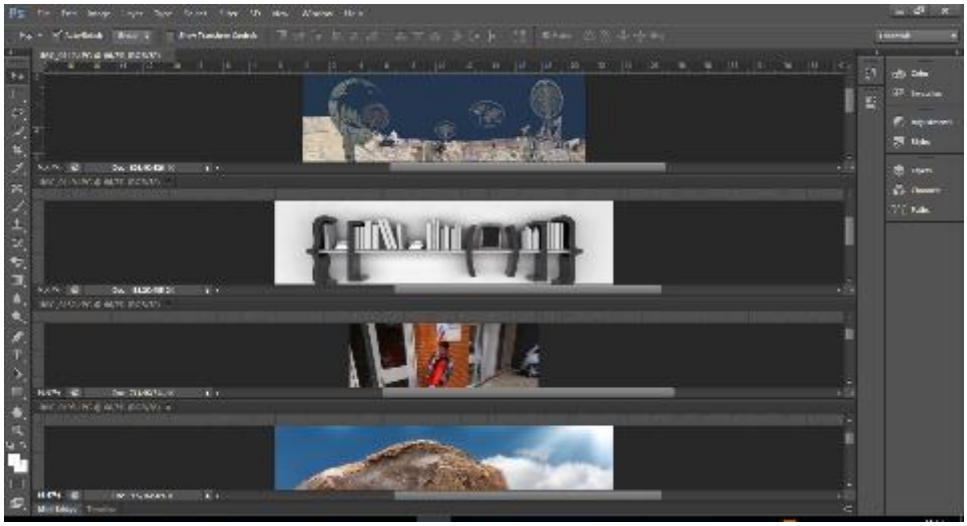
იმისათვის, რომ მართოთ ფანჯრების განლაგება, შედით Window -> Arrange. დამატებით მენიუში ჩამოთვლილია ფანჯრების განლაგების სხვადასხვა ვარიანტები:



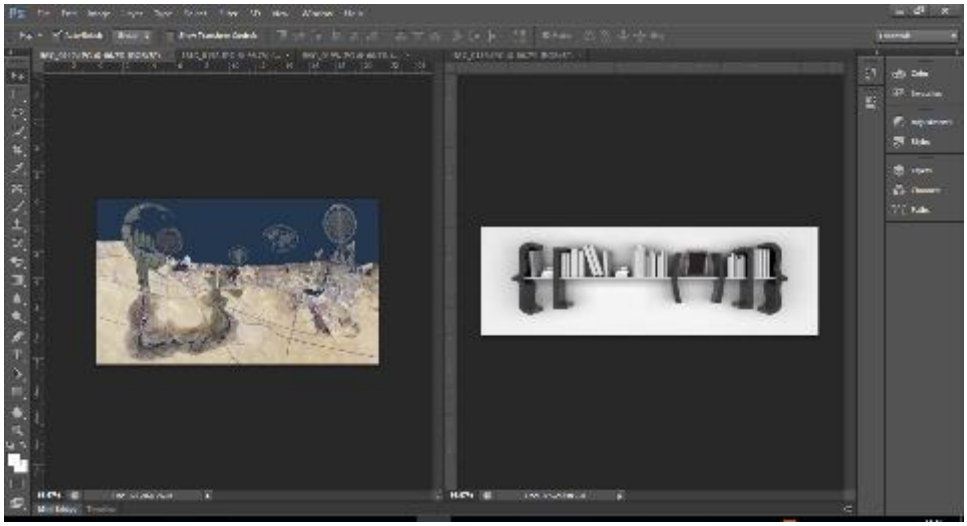
სურ. 1.1-11. Tile All Vertically - გვერდიგვერდ ვერტიკალურად დალაგება.



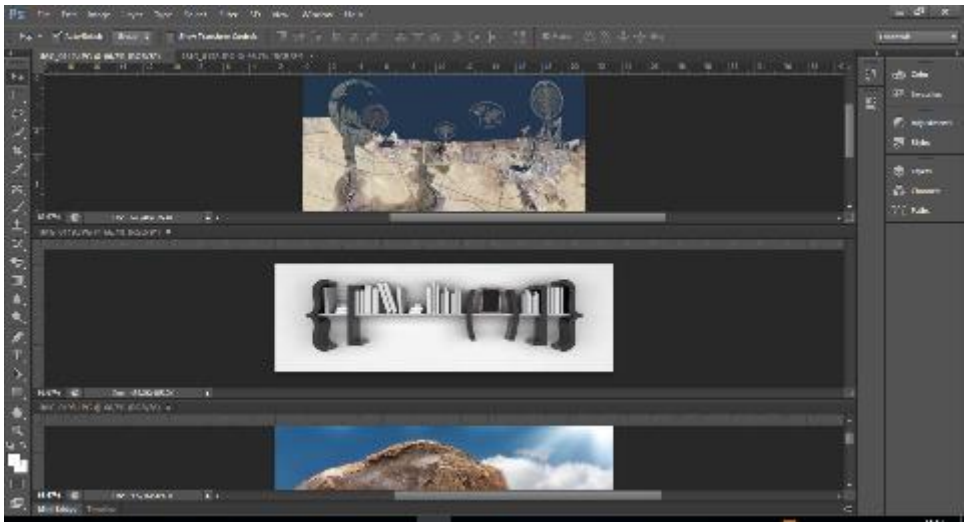
სურ. 1.1-13. 2-up Horizontal - მხოლოდ ორი გამოსახულების დალაგება ჰორიზონტალურად.



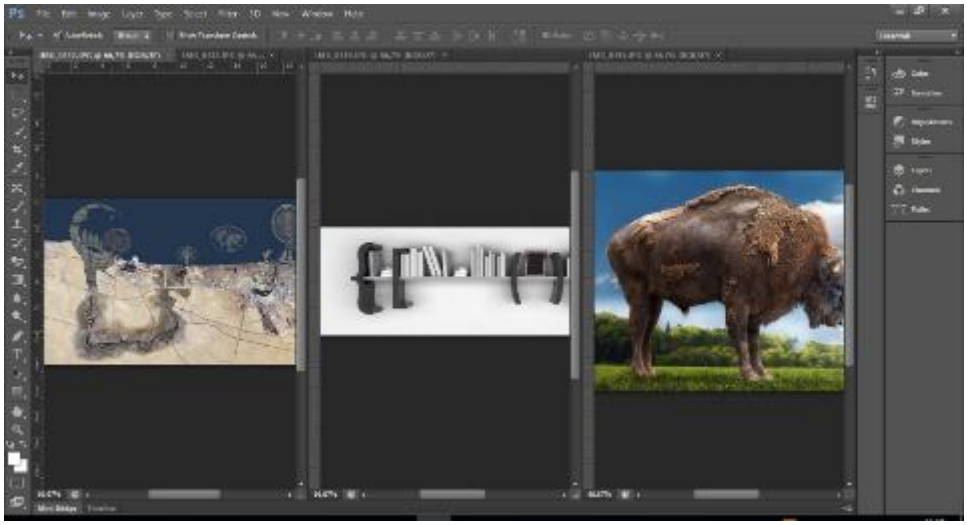
სურ. 1.1-14. All Tile Horizontally - გვერდიგვერდ ჰორიზონტალურად დალაგება.



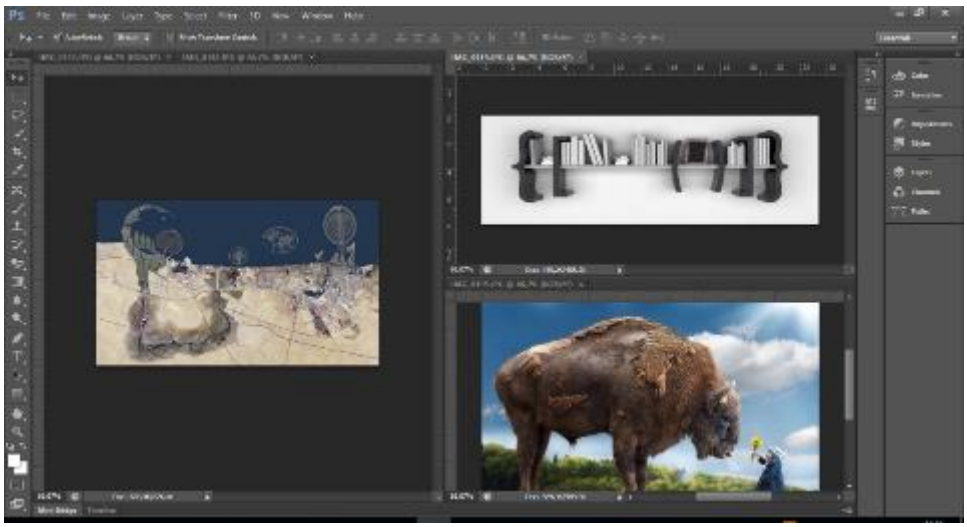
სურ. 1.1-12. 2-up Vertical - მხოლოდ ორი გამოსახულების დალაგება ვერტიკალურად.



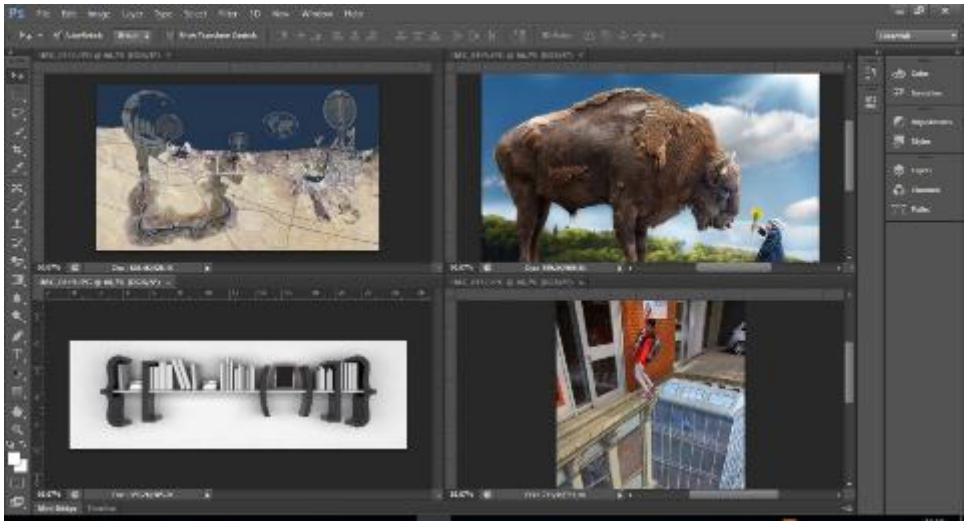
სურ. 1.1-16. 3-up Horizontal - მხოლოდ სამი გამოსახულების დალაგება ჰორიზონტალურად.



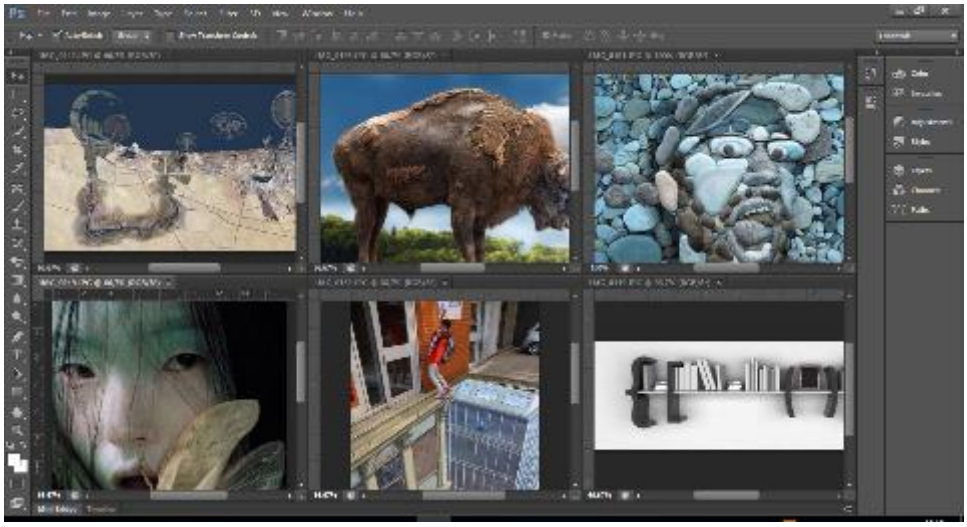
სურ. 1.1-15. 3-up Vertical - მხოლოდ სამი გამოსახულების დალაგება ვერტიკალურად.



სურ. 1.1-17. 3-up Stacked - სამად დალაგებული გამოსახულებები.

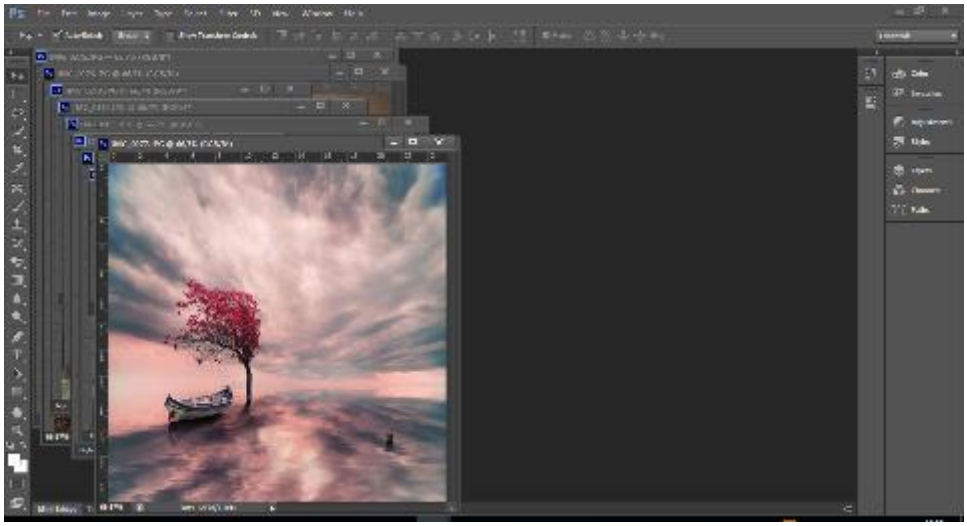


სურ. 1.1-19. 4-up - ოთხად დალაგებული გამოსახულებები.

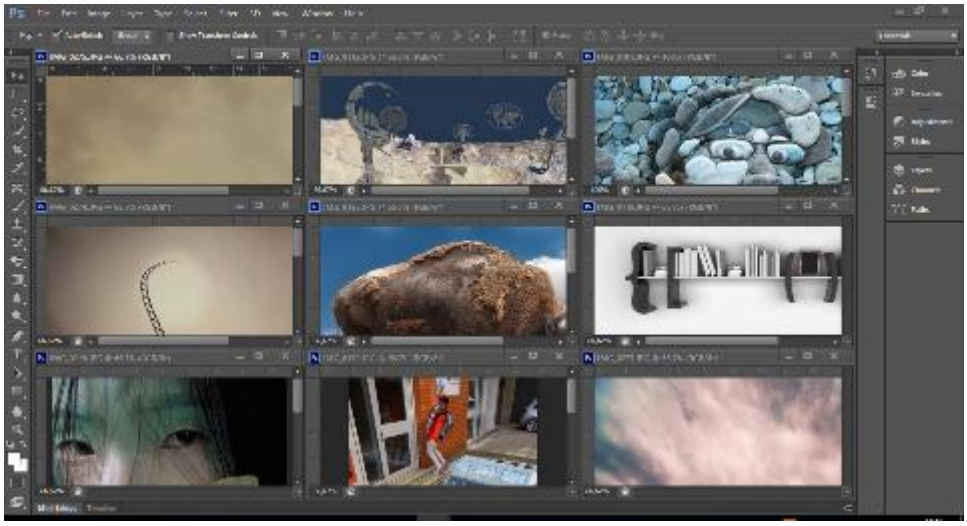


სურ. 1.1-18. 6-up - ექვსად დალაგებული გამოსახულებები.

ფანჯრების ამ მეთოდებით დალაგებისას, ისინი მაინც მიმაგრებულები არიან ძირითად ფანჯარაზე ჩანართების სახით. გარდა ამისა, შეგვიძლია ფანჯრების სხვაგვარად დალაგებაც - Cascade (კასკადურად), Tile (მოზაიკურად), Float in Window (ერთი ფანჯარა თავისუფალ მდგომარეობაში), Float All in Window (ყველა ფანჯარა თავისუფალ მდგომარეობაში).

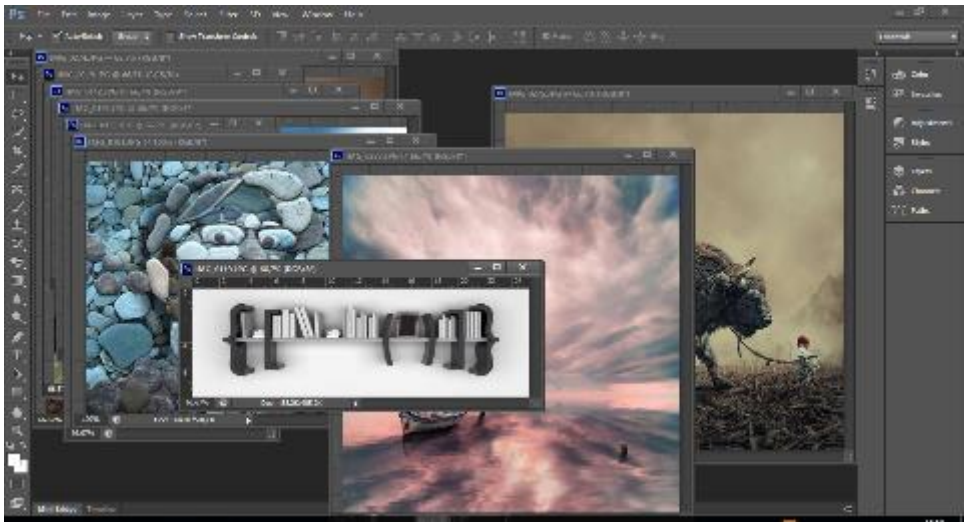


სურ. 1.1-20. Cascade - გამოსახულებების კასკადურად დალაგება.

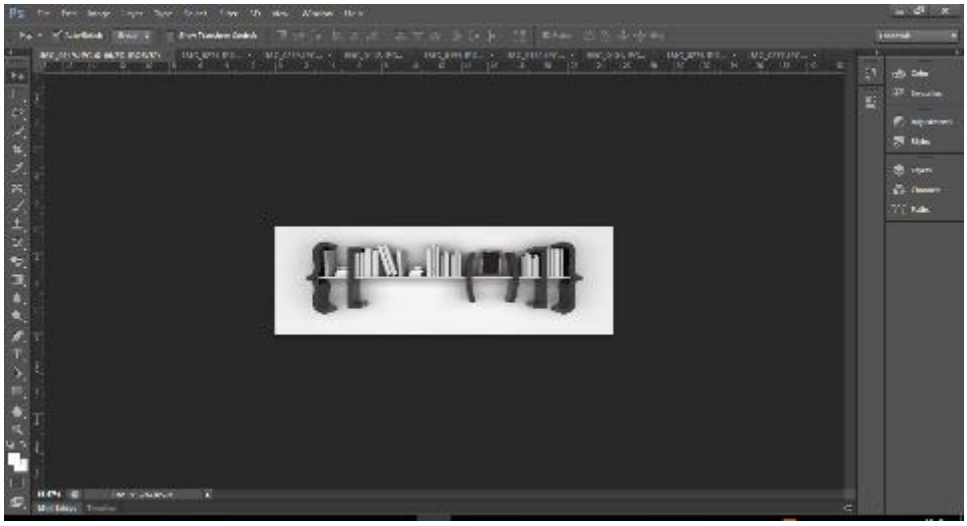


სურ. 1.1-22. Tile - გამოსახულებების მოზაიკურად დალაგება.

სურ. 1.1-21. Float in Window - ერთი გამოსახულება თავისუფალ მდგომარეობაში (ცალკე ფანჯარაში).



სურ. 1.1-23. Float All in Window - ყველა გამოსახულება თავისუფალ მდგომარეობაში (ცალკე ფანჯარაში).

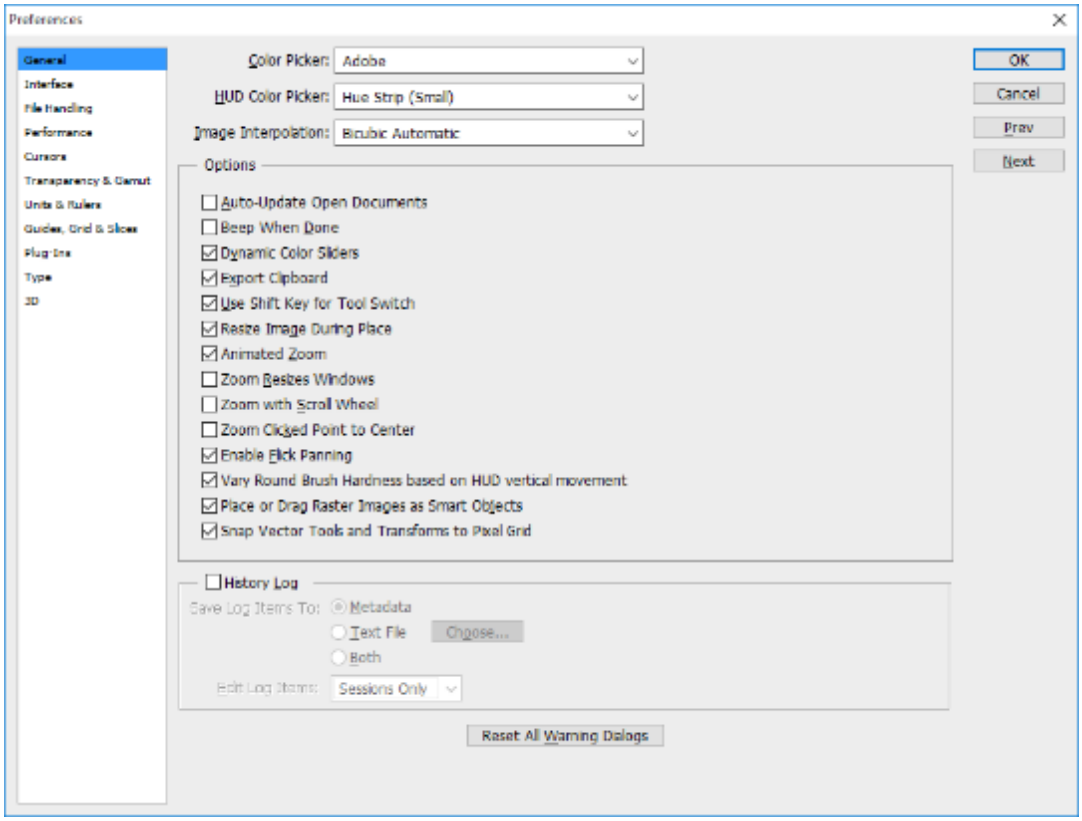


სურ. 1.1-24. Consolidate All to Tabs - ყველა ცალკე გამოტანილ ფანჯრებს ამაგრებს ისევ სანიშნის სახით.

## გრაფიკული რედაქტორის პარამეტრების განხილვა/დაყენება

სანამ მუშაობას შეუდგებით სასურველია გარკვეული ცვლილებები შეიტანოთ პროგრამის პარამეტრებში. ზოგიერთი პარამეტრის შეცვლა თქვენ დაგჭირდებათ მხოლოდ ერთხელ, ხოლო ზოგიერთი შესაძლოა პერიოდულად ცვალოთ. პარამეტრების დაყენება თვენი საქმიანობიდან და დასახული ამოცანიდან გამომდინარე შეიძლება. ფოტოგრაფი, ილუსტრატორი, მხატვარი, დიზაინერი, ტექნიკური დიზაინერი, 3D მოდელების შემქმნელი, ანიმატორი - ყველა ირგებს პროგრამას სათავისოდ და აყენებენ მათთვის სასურველ პარამეტრებს.

იმისათვის, რომ შეცვალოთ პროგრამის პარამეტრები, გამოიძახეთ Edit -> Preferences -> General (Ctrl +K). გამოსულ ფანჯარა შეიცავს ყველა იმ მახასიათებელს, რომელიც მოქმედებს სამუშაო პროცესზე ან პროგრამის წარმადობაზე:




სურ. 1.1-25. პროგრამა ფოტოშოპის მოწყობის ფანჯარა.

General – პროგრამის ზოგადი მახასიათებლების დაყენება, როგორიცაა ფერების პალიტრა, მასშტაბირება ან მოქმედებების აღნუსხვა;

Interface – ინტერფეისის პარამეტრების დაყენება, როგორიცაა მაგალითად ინტერფეისის ფერის დაყენება

File Handling – პროგრამის ფაილებთან მუშაობის პარამეტრები, მაგალითად ავტომატურად შენახვის დრო, Raw ტიპის ფაილებთან მუშაობა და სხვა

Performance – წარმადობა. აქ შეგიძლიათ დააყენოთ, ოპერატიული მეხსიერების რა ნაწილი გამოიყენოს პროგრამამ. დისკი, რომელზეც პროგრამა განათავსებს ვირტუალური მეხსიერების ფაილს. მოქმედებების ისტორიის რაოდენობა და სხვა;

Cursors – აქ შეგიძლიათ დააყენოთ პროგრამაში როგორ გამოჩნდეს კურსორი სხვადასხვა ინსტრუმენტისთვის, მაგალითად ფუნჯისთვის.

Transparency & Gamut – გამჭვირვალობისა და გამა (ფერთა). აქ შეგიძლიათ დააყენოთ სამუშაო ფაილში გამჭვირვალობა როგორ გამოჩნდეს. ასევე, როდესაც მოხდება ფერთა ცდომილება, მაგალითად RGB-დან CMYK-ში გადაყვანისას, აცდენილი ფერები რა ფრად გამოისახოს ეკრანზე.

Units & Rulers – აქ შეგიძლიათ დააყენოთ თქვენი სამუშაო საზომი ერთეული. მაგალითად ჩვენ შემთხვევაში ეს იქნება სანტიმეტრი, თუმცა როდესაც იმუშავებთ გამოსახულებებზე, რომელიც ინტერნეტისთვის მზადდება, იქ შეგიძლიათ გამოიყენოთ პიქსელები. აქ ასევე შეგიძლიათ დააყენოთ ახალი დოკუმენტებისთვის ნაგულისხმევი გარჩევადობის ხარისხი. სტანდარტულად მოცემული 300 და 72 px/inch, თუმცა ეს შეგიძლიათ შეცვალოთ რასაკვირველია.

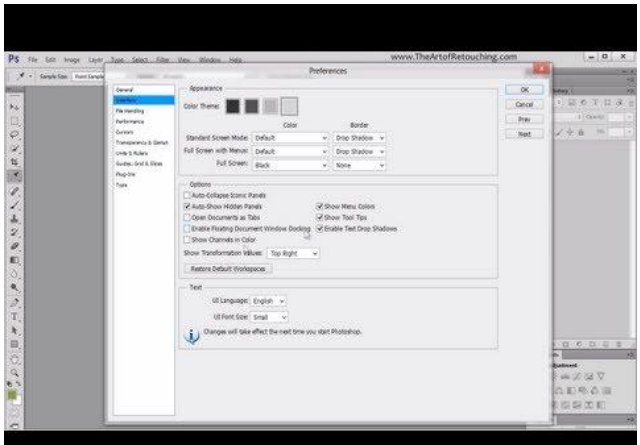
Guides, Grid & Slices – აქ შეგიძლია შეცვალოთ მიმმართველების, „ჭკვიანი“ მიმმართველების ფერები, დამხმარე ბადის დაყოფის მაჩვენებლები და სხვა.

Plug-Ins – პროგრამისთვის გამოსულია სხვადასხვა სახის დამატებები, რომელთა ადგილმდებარეობა შეგიძლიათ აქედან მიუთითოთ. ეს დამატებები გაძლევენ საშუალებას ისეთი ეფექტები შექმნათ გამოსახულებისთვის, რომელიც შეუძლებელია მიიღოთ პროგრამაში არსებული ხელსაწყოების დახმარებით.

Type – რამდენიმე პარამეტრი, რომელიც დაგეხმარებათ ტექსტთან მუშაობაში.

3D – აქ შეგიძლიათ დააყენოთ ვიდეოადაპტერის მახასიათებლები, გამოსახულების გარდაქმნის მეთოდები და სხვა.

**შენიშვნა:** *გახსოვდეთ, რომ ზოგიერთი პარამეტრის შეცვლა მოქმედებს კომპიუტერის წარმადობაზე, მაგალითად ოპერატიული მეხსიერების გამოყენება. ამიტომ მახასიათებლები უნდა შეირჩეს ისე, რომ არ მოხდეს მთლიანობაში კომპიუტერის წარმადობის შემცირება.*




ვიდეო 1.1-1. პროგრამის პარამეტრების მართვა

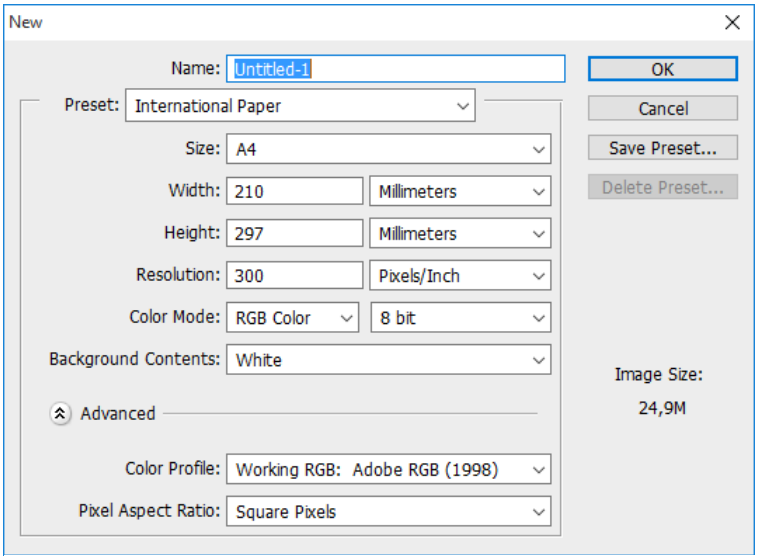
## დოკუმენტის კლასიფიკაცია და საზომი ერთეულები

მას შემდეგ, რაც მოხდება სამუშაოდ ყველანაირი პარამეტრის დაყენება: პროგრამის პარამეტრები, სამუშაო სივრცე, საზომი ერთეულები და სხვა, შეგიძლიათ დაიწყოთ გამოსახულებასთან მუშაობა.

გამოსახულებასთან მუშაობა შეგიძლიათ დაიწყოთ ორი გზით: შექმნათ ახალი გამოსახულება ან/და გახსნათ უკვე არსებული.

მივყვეთ რიგს და დავიწყოთ პირველით ანუ ახალი გამოსახულების შექმნა.

ამისათვის ვიძახებთ File -> New (Ctrl + N). გამოსულ დიალოგურ ფანჯარაში ჩვენ უნდა დავაყენოთ ახალი დოკუმენტის მახასიათებლები.




სურ. 1.1-26. ახალი დოკუმენტის შექმნის ფანჯარა.

ამ ფანჯარაში უნდა შევიყვანოთ ისე მახასიათებლები, როგორებიცაა ფაილის დასახელება, მისი ზომა, გარჩევადობის ხარისხი და სხვა. პროგრამას გააჩნია წინასწარ გამზადებული დოკუმენტის ტიპები, რომლებიც თქვენ შეგიძლიათ ირჩიოთ ან შექმნათ თქვენი საკუთარი. და თუ მას ხშირად იყენებთ, შეინახოთ ის სიაში და მომავალში მიმართოთ მას.

გამზადებული დოკუმენტები მოცემულია ჩამოსაშლელ მენიუში Preset:

**Clipboard** - თუ თქვენ რომელიმე პროგრამაში ან თუნდაც ფოტოშოპში, დაკოპირებული გაქვთ რაიმე გამოსახულება, ფოტოშოპი ავტომატურად კითხულობს მის მონაცემებს (სიგანე, სიმაღლე, გარჩევადობა) და გთავაზობთ შესაბამისი მახასიათებლების დოკუმენტის შექმნას.

**Default Photoshop Size** - ფოტოშოპის ნაგულისხმევი ზომა, გამზადებული დოკუმენტის ერთ-ერთი ნაირსახეობა.

**U.S. Paper** - დოკუმენტის ნაირსახეობა, რომელსაც იყენებენ ა.შ.შ.-ში. ამ დროს აქტიურდება ჩამოსაშლელი მენიუ Size, სადაც ფურცლის სხვადასხვა ვარიანტებიც შეგიძლიათ შეარჩიოთ. გარჩევადობა 300 px/inch.

**International Paper** - დოკუმენტის ნაირსახეობა, სადაც მოცემულია საყოველთაოდ გამოყენებადი ფურცლის ზომები. მენიუ Size-ში თქვენ ამ ფურცლის ზომებს იხილავთ: A4, A3, B5 და სხვა. გარჩევადობა 300 px/inch.

**Photo** - აქ მოცემული დოკუმენტების ზომები განკუთვნილია ფოტოსურათებისთვის, როგორც ვერტიკალური ისე ჰორიზონტალური ორიენტაციისთვის. თუმცა ეს ფორმატები მორგებულია ამერიკაში მიღებული სტანდარტისთვის. გარჩევადობა 300 px/inch.

**Web** - აქ მოცემულია ფორმატები, რომლებიც ინტერნეტისთვისაა განკუთვნილი. საზომი ერთეულებიც შესაბამისია აღებული. გარჩევადობა 72 px/inch.

**Mobile & Device** - ფორმატები, რომლებიც მოწყობილობების ეკრანის ზომას ასახავს. საზომი ერთეულები აქაც პიქსელებშია მოცემული. გარჩევადობა 72 px/inch.

**Film & Video** - დოკუმენტის ნაირსახეობა, რომლის პარამეტრები ორიენტირებულია ვიდეოპროდუქციაზე. მენიუ Size-ში თქვენ ნახავთ უამრავ ვიდეო ფორმატს, რომელიც შეგიძლიათ აირჩიოთ თქვენი ამოცანიდან გამომდინარე. გარჩევადობა 72 px/inch.

**Custom** - მომხმარებლის მიერ დაყენებული პარამეტრები.

ახალი დოკუმენტი შეგიძლიათ შექმნათ უკვე არსებული მზა დოკუმენტების ბაზაზე ან შექმნათ თქვენი საკუთარი. რა ძირითად მონაცემებს ვუთითებთ, როდესაც გვინდა შევქმნათ ახალი გამოსახულება:

**Width** - გამოსახულების სიგანე

**Height** - გამოსახულების სიმაღლე

მათ გვერდით ჩამოსაშლელ მენიუში ვუთითებთ საზომ ერთეულს - დუიმი, სანტიმეტრი, მილიმეტრი ან პიქსელი. თუ ვმუშაობთ დუიმებში და საბოლოო შედეგი მილიმეტრებში უნდა დაიბეჭდოს, უნდა გავითვალისწინოთ, რომ 1 დუიმი = 2,54 სანტიმეტრს

**Resolution** - გარჩევადობა. აქ ვუთითებთ გამოსახულების გარჩევადობის ხარისხს. თუ გამოსახულება განკუთვნილი იქნება ინტერნეტისთვის, მაშინ ვუთითებთ 72-96 px/inch, ხოლო თუ პოლიგრაფიული მიზნებისთვის უნდა გამოვიყენოთ, მაშინ 300 px/inch ან მეტი.

**Color Mode** - ფერთა მოდელი, რომელიც გამოიყენება ამ შემთხვევაში. ისევ და ისევ RGB თუ გამოსახულებას ვამზადებთ ინტერნეტისთვის, ხოლო თუ პოლიგრაფიული მიზნისთვის, მაშინ ვირჩევთ CMYK ფერთა მოდელს. ხოლო მის გვერდით ვირჩევთ, რამდენ ბიტიანი იქნება ფერთა სიღრმე. 8 ბიტი სავსებით საკმარისია ორივე მიზნისთვის, თუ ჩვენ წინაშე არ დგას რაიმე კონკრეტული ამოცანა, რომელიც უფრო მაღალბიტიან ფერთა სიღრმეს მოითხოვს. დანარჩენი ამ ეტაპზე დავტოვოთ უცვლელად და დავაწვეთ OK. იმ შემთხვევაში თუ თქვენ საკმაოდ ხშირად გიწევთ ამ ზომებთან მუშაობა, მაშინ ღილაკი Save Preset-ის დაგეხმარებათ შეინახოთ დოკუმენტი შაბლონის სახით და შემდეგში თქვენ მას აირჩევთ დოკუმენტების სიიდან და აღარ დაგჭირდებათ მონაცემების შეყვანა ხელით.

შემდეგ ეტაპზე ჩვენ განვიხილავთ, თუ როგორ უნდა ვიმუშაოთ უშუალოდ ფაილში და გამოვიყენებთ შესაბამის ხელსაწყოებს, რომლებიც დაგვეხმარებიან გამოსახულების შექმნასა თუ რედაქტირებაში.

## სავარჯიშოები

1. დაასახელეთ სხვაობა რასტრულ და ვექტორულ გამოსახულებებს შორის;
2. რა არის გარჩევადობა? ჩამოთვალეთ ყველაზე გავრცელებული.
3. ჩამოთვალეთ ფაილის ფორმატები;
4. ჩამოთვალეთ სამუშაო სივრცის ტიპები;
5. გამოაჩინეთ სასურველი სამუშაო პანელები; მოაწყეთ სამუშაო სივრცე, როგორც თქვენ ჩათვლით საჭიროდ და შეინახეთ.
6. მოიძიეთ ინფორმაცია ფბ-ს მთავარი სურათის (cover) შესახებ; გაეცანით ფბ-ს რეკომენდაციების გამოსახულების მახასიათებლებთან დაკავშირებით; შექმენით ახალი ფაილი, რომელიც დააკმაყოფილებს ამ მახასიათებლებს (ზომა, გარჩევადობა, ფერის მოდელი, ფაილის ფორმატი) და შეინახეთ ის შაბლონის სახით;
7. შექმენით ფაილი, რომელიც მომავალში გამოყენებული იქნება წიგნის ყდის გასაფორმებლად. ზომა 130 x 195 მმ. ფაილი შექმენით ბეჭდვისთვის საჭირო პირობების გათვალისწინებით (ფერთა მოდელი, გარჩევადობა, ფაილის ფორმატი).

## 1.2. მარტივი გამოსახულების შექმნა

პარაგრაფის შესაბამისი თემატიკა

- მონიშვნის ხელსაწყოების გაცნობა
- მოქმედებები მონიშნულ არეზე
- დაფარვის მეთოდების მიმოხილვა
- ფუნჯი და მისი ნაირსახეობები
- საშლელი და მისი ნაირსახეობები

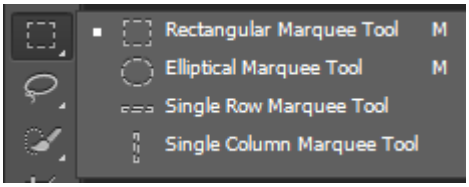
### მონიშვნის ხელსაწყოები და მონიშვნა

მონიშვნის ინსტრუმენტები ერთ-ერთი უმნიშვნელოვანესია ფოტოშოპში. მოქმედების პრინციპი მარტივია: როდესაც ჩვენ ვნიშნავთ რაიმე არეს, ჩვენი შემდეგი მოქმედებები შეეხება მხოლოდ არის შიგნით მყოფ გამოსახულების ფრაგმენტს. ინსტრუმენტების დახმარებით ჩვენ შეგვიძლია მოვნიშნოთ როგორც ერთი პიქსელი, ისე პიქსელთა დიდი ჯგუფი და მათზე გარკვეული მოქმედებები მოვახდინოთ: ფერთა კორექცია, შეფერვა, კოპირება, ტრანსფორმირება და სხვა.

ფოტოშოპში მონიშვნის სხვადასხვა ხელსაწყო არსებობს. თვისებებიც განსხვავდება ერთმანეთისგან - შეგვძლია გამოსახულებაზე მოვნიშნოთ მართკუთხა, ოვალური ან თავისუფალი ფორმის ფრაგმენტი, ერთფეროვანი პიქსელები, მოვნიშნოთ გამოსახულების ორი ან მეტი ფრაგმენტი და სხვა. ინსტრუმენტთა პანელზე არსებობს მონიშვნის ხელსაწყოთა მთელი ჯგუფი (სურ. 10: ა). პროგრამის მენიუში მას ეთმობა ცალკე მენიუ Selection - აქ მოთავსებული ფუნქციები გვაძლევენ საშუალებას მოვახდინოთ ჩვენი მონიშვნა გავხადოთ უფრო ზუსტი, მოვახდინოთ მასზე სხვადასხვა მოქმედებები, რომ შემდეგში მიღებული შედეგი ჩვენს მოთხოვნებს შეესაბამებოდეს.

### მონიშვნის ინსტრუმენტები (Marque Tool)

აქ გაერთიანებული ოთხი ინსტრუმენტი: Rectangular, Oval, Single Row და Single Column




სურ. 1.2-1. Rectangular Marque Tool - მართკუთხა მონიშვნის ინსტრუმენტი.

Single Row და Single Column შემთხვევაში ხდება 1 პიქსელის სიგანის არის მონიშვნა მთელი გამოსახულების გასწვრივ შესაბამისად ჰორიზონტალურად და ვერტიკალურად.

**შენიშვნა:** Shift კლავიშთან ერთად მონიშვნის შემთხვევაში ხდება სიმეტრიული არეების (კვადრატი და წრე) მონიშვნა. ხოლო Shift და Alt კლავიშების კომბინაციით მონიშვნა სრულდება ცენტრიდან და მიიღება კვადრატი და წრე.

ნებისმიერი ინსტრუმენტის არჩევის შემდეგ, ფორმატირების ზოლზე (სურ. 3: ბ) ჩნდება ინსტრუმენტისთვის დამახასიათებელი პარამეტრები. ამ შემთხვევაშიც აქ გამოჩნდება ის პარამეტრები, რომელიც ამ მონიშვნის ინსტრუმენტს ახასიათებს:


სურ. 1.2-2. ფორმატირების ზოლი. მასზე გამოტანილია მონიშვნის ინსტრუმენტის დამატებითი პარამეტრები.

### *ა. მონიშვნის მეთოდები (მარცხნიდან მარჯვნივ):*

ახალი მონიშვნა (New Selection) - ახალი მონიშნული არე. ამ დროს თუ გაქვთ რამე მონიშნული, ის გაუქმდება და შეიქმნება ახალი;

მონიშვნის დამატება (Add to Selection) - უკვე მონიშნულ არეს დაემატება თქვენს მიერ მონიშნული ახალი არე;

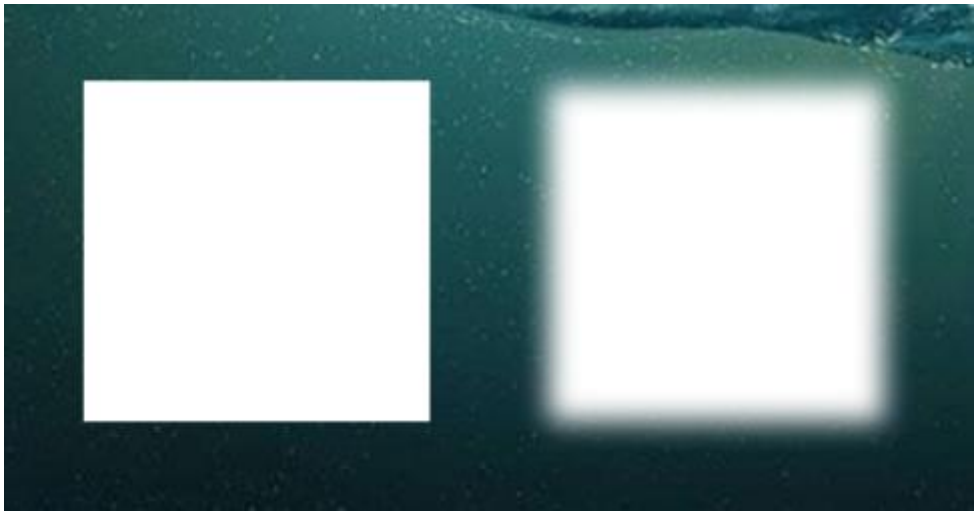
მონიშვნის გამოკლება (Subtract from Selection) - უკვე მონიშნულ არეს შეგიძლიათ გამოაკლოთ ფრაგმენტი;

მონიშნული არეების გადაკვეთა (Intersect with Selection) – ორი (ან მეტი) მონიშნული არის გადაკვეთიდან შეგიძლიათ გამოყოთ ერთი სასურველი არე.

დეტალური ინფორმაცია, თუ როგორ უნდა გამოიყენოთ ეს ოთხი ინსტრუმენტი, შეგიძლიათ იხილოთ სტატიაში Unlock The Full Power Of Basic Selections In Photoshop.

### *ბ. დარბილება (Feather)*

მისი საშუალებით ხდება მონიშნული არის საზღვრების დარბილება. უთითებთ რამდენ პიქსელიანი იყოს საზღვრის დარბილება. რაც მეტია მაჩვენებელი, მით უფრო რბილია საზღვარი.



სურ. 1.2-3. ამ გამოსახულებაზე მოხდა მონიშნული ფრაგმენტის წაშლა. მარცხნივ დარბილება 0 პქს, მარჯვნივ დარბილება 5 პქს.

### *გ. მონიშვნის სტილი (Style)*

ჩვეულებრივი მონიშვნის დროს ჩვენ ვხაზავთ ნებისმიერი პროპორციისა თუ ზომის არეს. თუმცა, პროპორციების დაცვა და წინასწარ განსაზღვრული ზომის მონიშნული არის შექმნა შესაძლებელია. ამისათვის ჩამოსაშლელი მენიუ Style-ში არის სამი რეჟიმი:

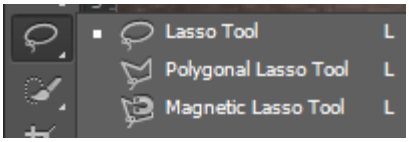
**Normal** (ჩვეულებრივი) მეთოდი ავტომატურად არის დაყენებული და მონიშვნის პროპორციებსა და ზომებს მომხმარებელი არეგულირებს.

**Fixed Ratio** (ფიქსირებული პროპორცია). ამ შემთხვევაში, აქტიურდება სიგრძისა (Width) და სიმაღლის (Height) ველები, რომლებშიც უთითებთ მონიშვნის სიგანისა და სიმაღლის პროპორციებს: 3:4, 16:9 და ა.შ.;

**Fixed Size** (ფიქსირებული ზომა). ამ დროს შესაძლებელია სიგრძისა (Width) და სიმაღლის (Height) ველებში სასურველი ზომების მითითება. შემდეგ საკმარისია გამოსახულებაზე მაუსის დაწკაპუნება და ავტომატურად შეიქმნება მითითებული ზომების შესაბამისი მონიშვნის არე.

## ინსტრუმენტები ლასო (Lasso Tool)

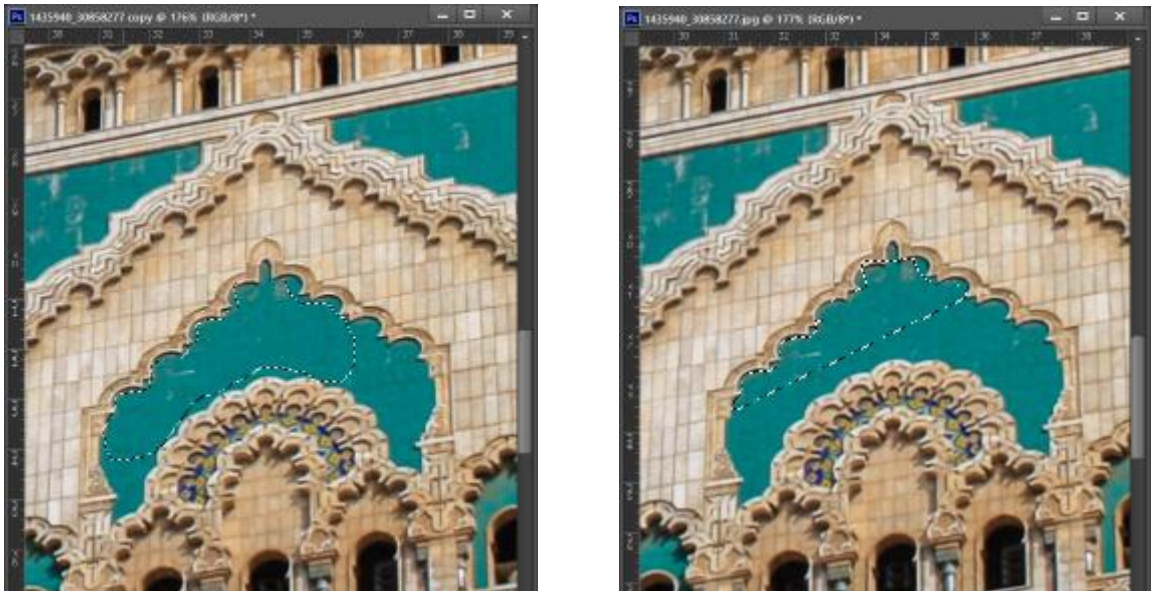
ამ ინსტრუმენტების დახმარებით ჩვენ ნებისმიერი ფორმის ობიექტი შეგვიძლია მოვნიშნოთ. შედის სამი ინსტრუმენტი: Lasso Tool, Polygonal Lasso და Magnetic Lasso.




ფორმატირების ზოლზე თქვენ შეგიძლიათ დააყენოთ სხვადასხვა პარამეტრი. აქ თქვენ შეხვდებით ისეთივე ღილაკებს, როგორებიცაა მონიშნული არის დამატება, გამოკლება, დარბილება. ასევე მაგნიტური ლასოს შემთხვევაში ჩნდება კიდევ რამდენიმე დამატებითი ფუნქცია:


### *ა. ლასო (Lasso Tool)*

მისი დახმარებით ჩვენ შეგვიძლია არასწორი ფორმის ობიექტების მონიშვნა. მუშაობის პრინციპი ესეთია: ხელაუღებლად უნდა შემოვატაროთ მოსანიშნი ობიექტის კონტურზე და მივიყვანოთ კურსორი იმ წერტილში, საიდანაც დავიწყეთ. იმ შემთხვევაში თუ გავუშვებთ ხელს, მაშინ ბოლო და თავი შეერთდებიან უმოკლესი წერტილით.



სურ. 1.2-4. მარცხენა სურათზე თქვენ ხედავთ, რომ მონიშნულია გარკვეული არე, ხოლო მარჯვენა სურათზე პირდაპირ შეერთდა ბოლო და საწყისი წერილი.

ლასოთი მუშაობა მოითხოვს გარკვეულ სიზუსტეს, იმისათვის, რომ ფრაგმენტი მოინიშნოს ისე, როგორც საჭიროა. დიდი და უსწორმასწორო ფრაგმენტების მონიშვნა მოუხერხებელია ამ ინსტრუმენტით. ასეთი მეთოდისთვის ძალიან კარგია მეორე ინსტრუმენტი - **სწორხაზოვანი ლასო** (Polygonal Lasso).

### *გ. სწორხაზოვანი ლასო (Polygonal Lasso)*

**სწორხაზოვანი ლასოს** გამოყენების დროს თქვენ კი არ ხაზავთ ერთ სრულ ფორმას, არამედ წერტილ-წერტილ მიყვებით ობიექტის კონტურს და როდესაც შემოავლებთ მთლიანად, ბრუნდებით საწყის წერტილში და ბოლო წერტილს აერთებთ საწყისთან. თუ რომელიმე ადგილას ორჯერ დააწკაპებთ სწრაფად, ისევე როგორც ჩვეულებრივი **ლასოს** შემთხვევაში, ბოლო წერტილი დაუკავშირდება საწყისს უმოკლესი გზით.

სწორხაზოვანი ლასო კარგია გამოიყენოთ ისეთი ობიექტების მოსანიშნად, რომელსაც არასწორი გეომეტრიული ფორმა აქვს.

სურ. 1.2-5. სწორხაზოვანი ლასოს გამოყენების მაგალითი.

**შენიშვნა:** დააწექით Backspace კლავიშს, რომ წაშალოთ დასმული წერტილი; თუ მონიშვნისას შეგხვდათ გამრუდებული კონტური, დააწექით Alt კლავიშს და ინსტრუმენტი დროებით გადაიქცევა ჩვეულებრივ ლასოთ.

### *გ. მაგნიტური ლასო (Magnetic Lasso)*

მაგნიტური ლასო ძალიან მოსახერხებელია გამოიყენოთ იქ, სადაც ჭარბობს ფერი და მისი ტონალობა. როდესაც მოსანიშნი ობიექტის ტონალობა ძალიან მიახლოებულია ფონის ფერთან, ეს ინსტრუმენტი შეუცვლელია. ფორმატირების ზოლზე სტანდარტული მახასიათებლების გარდა (მოსანიშნი არის დამატება, გამოკლება და სხვა), არის კიდევ სამი პარამეტრი:

**სიგანე (Width)** - ვუთითებთ, ცენტრალური წერტილიდან რა მანძილზე უნდა განისაზღვროს ობიექტის საზღვრები. მერყეობს 1-დან 256 პიქსელამდე;

**კონტრასტი (Contrast)** – ინსტრუმენტისთვის აქ ვუთითებთ, რამდენად კონტრასტულია ობიექტი ფონთან შედარებით. თუ ობიექტი კარგად გამოიკვეთება უკანა ფონზე, მიუთითეთ 30% და მეტი. თუ სუსტად გამოიკვეთება, მაშინ ნაკლები;

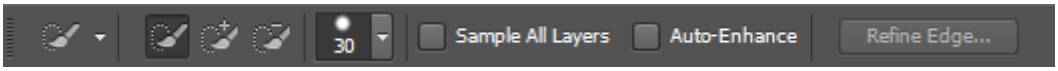
**სიხშირე (Frequency)** - განსაზღვრავს, რამდენად ხშირად უნდა დასვას ინსტრუმენტმა საყრდენი წერტილები. რაც უფრო მეტია სიხშირე, მით მეტ საყრდენ წერტილს დასვამს.

### სწრაფი მონიშვნის ინსტრუმენტები

სწრაფი მონიშვნის ინსტრუმენტებს მიეკუთვნებიან **სწრაფი მონიშვნა** (Quick Selection), **ჯადოსნური ჯოხი** (Magic Wand). მათი მოქმედება ფერისა და მის ტონალობაზეა დამყარებული.

#### *ა. სწრაფი მონიშვნა (Quick Selection)*

ინსტრუმენტის მოქმედება დაფუძნებულია მოსაზღვრე პიქსელების მსგავსებაზე. მონიშვნის პროცედურა ფუნჯით ხატვის პროცესს გავს. **ჯადოსნური ჯოხისგან** (Magic Wand) განსხვავებით არ გააჩნია პარამეტრი **შეღწევადობა** (Tolerance) და გაძლევთ საშუალებას თქვენ თვითონ განსაზღვროთ მონიშვნის არე (მოსაზღვრე პიქსელები) გამოსახულების შემოხატვით.




მისი პარამეტრებია:

**ახალი მონიშნული არე** (New Selection) - გამოსახულებაზე ახალი მონიშნული არის შექმნა;

**მონიშნული არის დამატება** (Add to Selection) - ახალი მონიშნული არის დამატება გამოსახულებაზე უკვე არსებულ მონიშნულ არესთან;

**მონიშნული არის გამოკლება** (Subtract from Selection) - გამოსახულებაზე უკვე არსებულ მონიშნულ არეს ნაწილის გამოკლება;

**ფუნჯი და მისი მახასიათებლები:**

**ზომა** (Size) - ფუნჯის ზომა. მინიმალური სივრცე, რომელიც გამოსახულებაზე იქნება მონიშნული.

**სიმკვეთრე** (Hardness) - განსაზღვრავს მონიშნული არის კიდეების სიმკვეთრეს.

**ინტერვალი** (Spacing) - მონიშნულ არეებს შორის მონიმალური მანძილი, რომელიც გამოსახულებაზე ხატვისას წარმოიქმნება.

**კუთხე** (Angle) - გამოსახულებაზე ხატვისას ფუნჯის დახრის კუთხე.

**ფორმა** (Roundness) - მონიშნული არის ფორმა, რომელიც გამოსახულებაზე ხატვისას წარმოიქმნება.

**ყველა ფენის ნიმუში** (Sample all Layers) - თუ ეს პარამეტრი ჩართულია მონიშვნისას გათვალისწინებული იქნება ყველა არსებულ ფენაზე მყოფი პიქსელების მსგავსება;

**ავტომატური გაძლიერება** (Auto-Enhance) - თუ ეს ფუნქცია ჩართულია, შესაძლებელია კიდეების გასწორება მოსანიშნი ობიექტის კონტურების მიხედვით.

სურ. 1.2-6. სწრაფი მონიშვნის ინსტრუმენტის გამოყენების მაგალითი, ფოტო: გიორგი კევლიშვილი

### *ბ. ჯადოსნური ჯოხი (Magic Wand)*

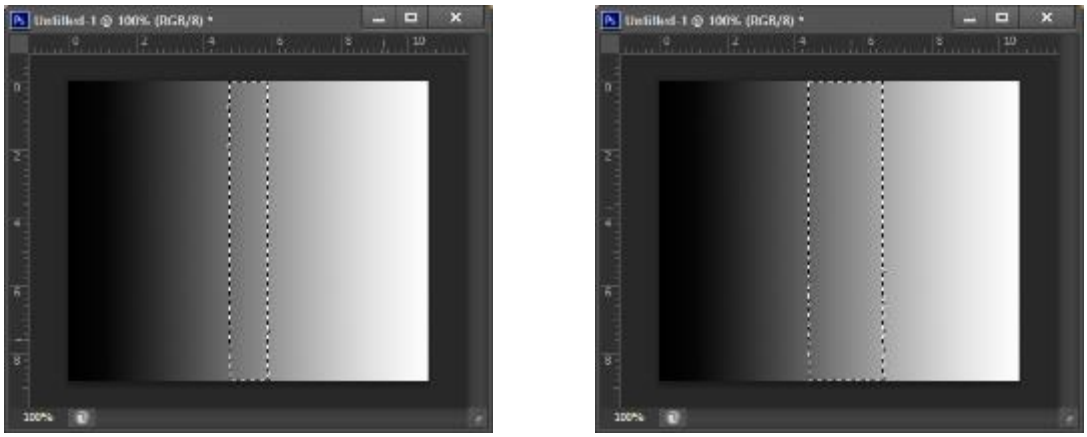
მისი მოქმედება დამოკიდებულია 4 მახასიათებელზე: **ნიმუშის ზომა** (Sample Size), **შეღწევადობა** (Tolerance), **მომიჯნავე პიქსელები** (Contiguous) და **ყველა ფენის ნიმუში** (Sample All Layers).

**შეღწევადობა** (Tolerance) ძირითადი პარამეტრია, რომელიც განსაზღვრავს, სიკაშკაშის რამდენ მახასიათებელს მოიცავს ინსტრუმენტი მონიშნულ არეში. მისი მნიშვნელობა მერყეობს 0-დან 255-მდე. აქ 0 არის სიკაშკაშის მხოლოდ ერთი მაჩვენებელი, 32 (რაც ნაგულისხმევად გამოიყენება) მოიცავს სიკაშკაშის 32 მაჩვენებელს, 255 კი ყველ მაჩვენებელს (ფაქტობრივად მთლიან გამოსახულებას). ინსტრუმენტი მაშინვე წყვეტს მონიშვნას, როდესაც აღმოაჩენს, რომ ფერის ტონალობა განსხვავდება **შეღწევადობაში** მითითებული მაჩვენებლისგან.

მიაქციეთ ყურადღება, რომ აქ საუბარია სიკაშკაშის დონეზე და არა ფერზე, რადგან ინსტრუმენტი ფერს საერთოდ არ ითვალისწინებს. ამის ნაცვლად ინსტრუმენტი აანალიზებს გამოსახულების RGB (CMYK) არხებს, რომლებიც წარმოადგენენ ერთმანეთზე დადებულ სამ (ოთხ) გამოსახულებას ნაცრისფერ ტონალობაში.

ქვემოთ მოცემულ მაგალითზე ნათლად ჩანს, როგორ მუშაობს **შეღწევდობა**.

მარცხენა სურათზე **შეღწევადობა** დაყენებულია 16. ეს იმას ნიშნავს, რომ დაწკაპუნების წერტილიდან ინსტრუმენტმა მონიშნა 16 პიქსელი მარცხნივ და 16 პიქსელი მარჯვნივ. მეორე სურათზე შეღწევადობის მაჩვენებელი არის 32, რაც ნიშნავს, რომ ყოველი მხრიდან მან მონიშნა 32 ტონალობა.



სურ. 1.2-7. მარცხენა სურათზე შეღწევადობის მაჩვენებელია 16, ხოლო მარჯვენაზე შეღწევადობის მაჩვენებელია 32

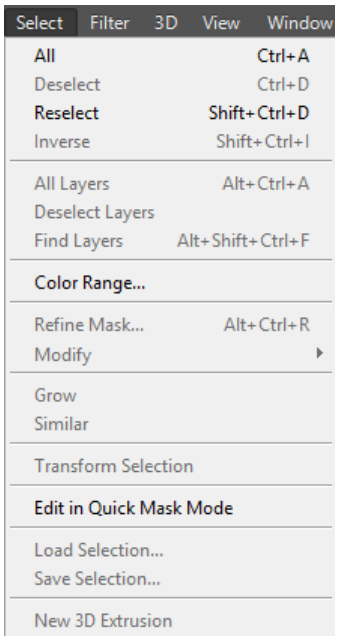
**მომიჯნავე პიქსელები** (Contiguous) - ეს პარამეტრი ავტომატურად ჩართულია და მიუთითებს ინსტრუმენტს, რომ უნდა მონიშნოს ის პიქსელები, რომლებიც ესაზღვრებიან იმ ადგილს, სადაც ინსტრუმენტით დააწკაპუნეთ. იმ შემთხვევაში თუ გსურთ მსგავსი ტონალობის პიქსელების მონიშვნა მთელი გამოსახულების მასშტაბით, მაშინ ეს პარამეტრი გამორთეთ.

**ყველა ფენის ნიმუში** (Sample All Layers) - ჩართეთ ეს პარამეტრი იმ შემთხვევაში, თუ გსურთ რომ ინსტრუმენტის მოქმედება სხვა ფენებზეც გავრცელდეს.

**ნიმუშის ზომა** (Sample Size) - ამ პარამეტრებს ინსტრუმენტი იყენებს ტონალობის შესაფასებლად. **წერტილოვანი ნიმუში** (Point Sample) - ითვალისწინებს ზუსტად იმ წერტილის მახასიათებელს, სადაც თქვენ დააწკაპუნეთ ინსტრუმენტი; **3x3 საშუალო** (3x3 Avarage) - ითვალისწინებს 9 პიქსელის მახასიათებელს; 5x5 საშუალო (5x5 Avarage) - ითვალისწინებს 25 პიქსელის მახასიათებელს და ა.შ.

გამოსახულებაზე არეების, პიქსელების, ცალკეული ფერის ტონალობების მონიშვნას ძალიან დიდი მნიშვნელობა ენიჭება. პროგრამაში ცალკეა გამოტანილი **მონიშვნის** მენიუ (Select), რომლის დახმარებითაც თქვენ შეგიძლიათ როგორც მონიშვნა, ისე სხვადასხვა მოქმედების შესრულება.

## მონიშვნის მენიუ (Select)




სურ. 1.2-8. მონიშვნის მენიუ (Select)

**ყველაფრის მონიშვნა** (All – Ctrl + A) - გამოსახულებას მთლიანად მონიშვნა.

**მონიშვნის გაუქმება** (Deselect – Ctrl + D).

**ხელახლა მონიშვნა** (Reselect – Shift + Ctrl + D) - ბოლოს გაუქმებული მონიშვნის აღდგენა.

**მონიშვნის შებრუნება** (Inverse – Shift + Ctrl + I) - მოინიშნება გამოსახულების ის ნაწილი რაც არ არის მონიშნული.

**ყველა ფენა** (All Layers – Alt + Ctrl + A) - ფენების პანელში მოხდება გამოსახულების ყველა ფენის მონიშვნა.

**ფენების მონიშვნის გაუქმება** (Deselect Layers).

**მსგავსი ფენების მონიშვნა** (Similar Layers).

**ფერთა დიაპაზონი** (Color Range) - წარმოადგენს **ჯადოსნური ჯოხის** (Magic Wand) ინტელექტუალურ ვერსიას. საშუალებას იძლევა ფერისა და მისი ტონალობის საფუძველზე შექმნას მონიშვნა. ეს იმდენად მძლავრი ფუნქციაა, რომ ცალკე განხილვის იმსახურებს.

**საზღვრის სრულყოფა** (Refine Edge) - გამოიყენება მონიშვნის საზღვრის რედაქტირებისთვის. მისი დახმარებით შესაძლებელია ჩვენი მონიშნული არე უფრო ზუსტი გავხადოთ, რაც შემდგომ ეტაპზე მუშაობას გაცილებით გაგვიადვილებს.

**საზღვრების ცვლილება** (Modify), რომლის ქვემენიუში 5 ბრძანებაა მოთავსებული: **საზღვარი** (Border), **გასწორება** (Smooth), **გაფართოება** (Expand), **შეკუმშვა** (Contract), **დარბილება** (Feather).

ა. **საზღვარი** (Border) - აქ მითითებული მნიშვნელობის შემდეგ, პროგრამა ხელახლა გადახატავს მონიშნულ არეს ორმაგი საზღვრით, რომლის სიგანე იქნება თქვენს მიერ მითითებული მნიშვნელობის ტოლი. დიაპაზონი 1-დან 200 პიქსელამდე. მცირე მნიშვნელობების მითითება სასარგებლოა, როდესაც თქვენ გინდათ აკურატულად გააბუნდოვნოთ, ჩაამუქოთ ან სხვანაირად დაამუშაოთ მონიშნული ობიექტის საზღვრები.

ბ. **გასწორება** (Smooth) - გამოიყენება იმ შემთხვევაში, როდესაც თქვენს მიერ შექმნილ მონიშნულ არეს გააჩნია ბევრი კუთხოვანი ადგილები. ამ ბრძანებით თქვენ ახდენთ კუთხოვანი ადგილების გასწორებას. მისათითებელი დიაპაზონი მერყეობს 1-დან 100 პიქსელამდე.

გ. **გაფართოება** (Expand) - მონიშნულ არეს ზრდის მითითებული მნიშვნელობით. მისათითებელი დიაპაზონი მერყეობს 1-დან 100 პიქსელამდე.

დ. **შეკუმშვა** (Contract) - მონიშნულ არეს ამცირებს მითითებული მნიშვნელობით. მისათითებელი დიაპაზონი მერყეობს 1-დან 100 პიქსელამდე.

ე. **დარბილება** (Feather) - არბილებს მონიშნული არის საზღვრებს. მისათითებელი დიაპაზონი მერყეობს 0.2-დან 250 პიქსელამდე. (იხ. სურ. 1.2-3)

ბრძანებები **მომიჯნავე** (Grow) და **მსგავსი** (Similar), მონიშვნას ქმნიან წინასწარ განსაზღვრული ფერის ეტალონის საფუძველზე. ორივე ბრძანების მოქმედებას განსაზღვრავს **ჯადოსნური ჯოხის** (Magic Wand) ინსტრუმენტის **შეღწევადობის** (Tolerance) სიდიდე (იხ. გვ. 28). გამოსახულებაზე უნდა შევქმნათ მონიშვნა იმ ფერებზე, რომლის გამოყოფაც გვინდა და ამოვირჩიოთ ორი ბრძანებიდან ერთ-ერთი:

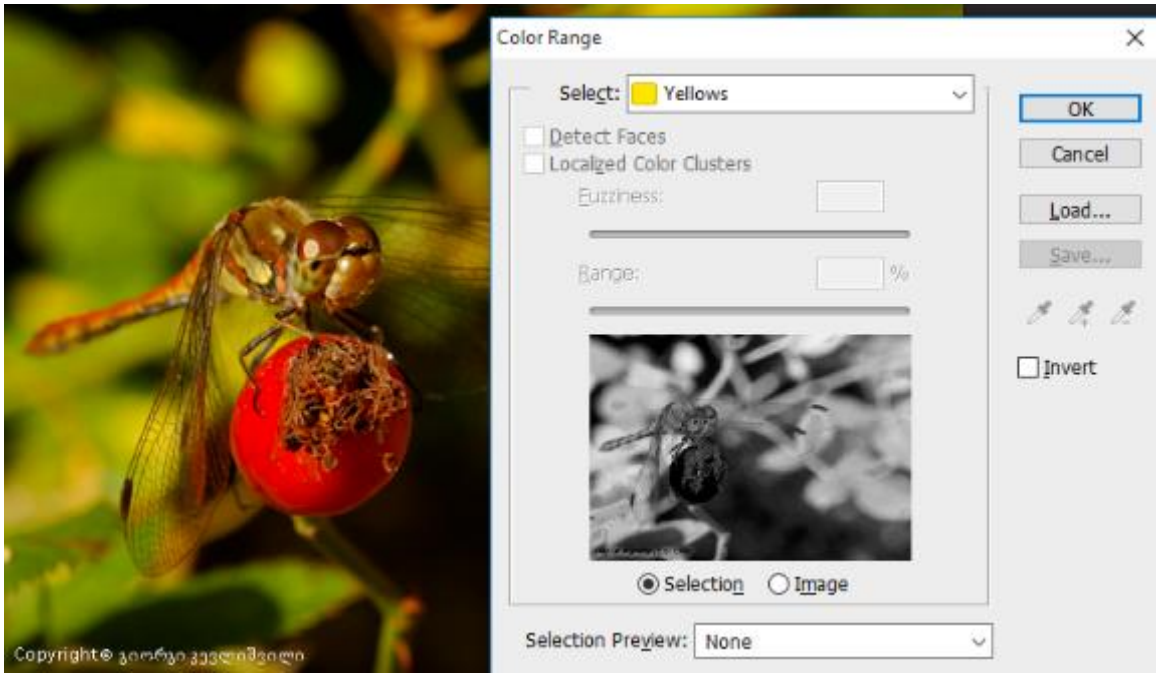
**მომიჯნავე** (Grow) მონიშნავს მხოლოდ იმ პიქსელებს, რომლებიც საწყის მონიშნულ არეს ემიჯნებიან.

**მსგავსი** (Similar) ახდენს მთლიანი გამოსახულების ანალიზს და ყველა იმ პიქსელს მონიშნავს, რომელიც **შეღწევადობაში** (Tolerance) მითითებული პარამეტრის კრიტერიუმებს აკმაყოფილებს.

ცალკე ყურადღებას იმსახურებს ორი მნიშვნელოვანი ფუნქცია: მონიშვნა **ფერის დიაპაზონით** (Color Range) და მონიშნული არის **საზღვრის სრულყოფა** (Refine Edge).

### ფერის დიაპაზონი (Color Range)

ეს ფუნქცია წარმოადგენს **ჯადოსნური ჯოხის** (Magic Wand) „ინტელექტუალურ“ ვერსიას, რადგან ის ახდენს მონიშვნას ფერისა და ფერთა ტონალობის მიხედვით, გაძლევთ წინასწარ ხედს და შესაბამისად მეტი კონტროლის საშუალებას. **ჯადოსნური ჯოხი** (Magic Wand) ან ნიშნავს პიქსელს ან არა. **ფერთა დიაპაზონი** (Color Range) კი საშუალებას იძლევა პიქსელი ნაწილობრივ მონიშნოთ.

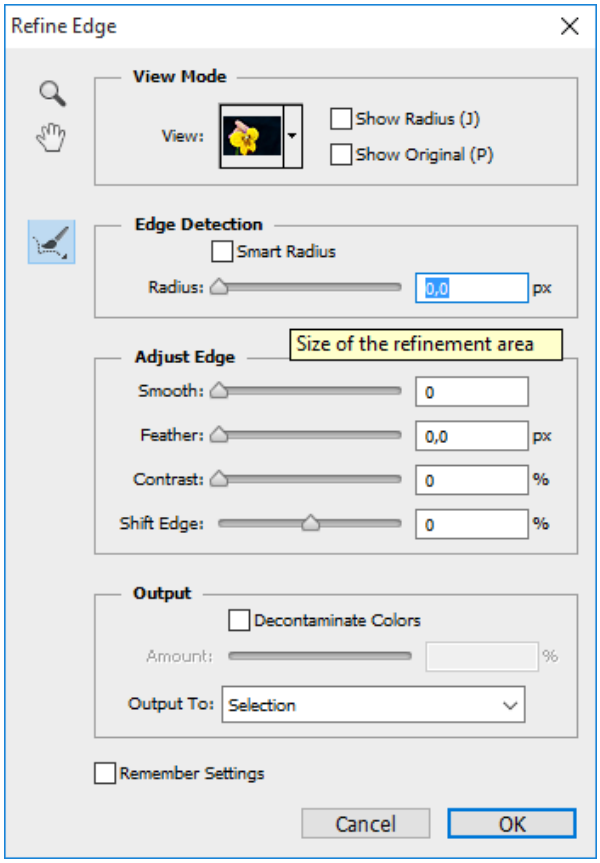



სურ. 1.2-9. ფერთა დიაპაზონის (Color Range) დიალოგური ფანჯარა. ამ შემთხვევაში მითითებულია, რომ გამოსახულებაზე მოინიშნოს ყვითელი ფერი. ფოტო: გიორგი კევლიშვილი

დიალოგური ფანჯრის დახმარებით შესაძლებელია გამოსახულებაზე რომელიმე კონკრეტული ფერის მონიშვნა ან ფერის მონიშვნა მოხდეს ჩვენს მიერ მითითებული ნიმუშის მიხედვით. ნიმუშის მითითებისას ვიყენებთ პიპეტის ინსტრუმენტს, რომელის დახმარებითაც ჩვენ შეგვიძლია შევარჩიოთ, დავამატოთ ან გამოვაკლოთ საჭირო ფერი ან მისი ტონალობა.

### საზღვრის სრულყოფა (Refine Edge)

ზემოთ განხილული მონიშვნის ყველა ინსტრუმენტს ფორმატირების პანელზე და მონიშვნის (Select) მენიუში **საზღვრების სრულყოფის** (Refine Edge) ფუნქცია.




სურ. 1.2-10. საზღვრის დაზუსტების (Refine Edge) დიალოგური ფანჯარა.

მისი დახმარებით ჩვენ შეგვიძლია მონიშნული არე მაქსიმალურად დახვეწილი და სამუშაოდ გამოყენებადი გავხადოთ. ერთ ფანჯარაში არის მოთავსებული ისეთი ინსტრუმენტები, როგორებიცაა **გასწორება** (Smooth), **შერბილება** (Feather), **სიმკვეთრე** (Contrast) და სხვა. ყველაფერი ეს საშუალებას გაძლევთ თქვენი მონიშნული არე მაქსიმალურად მორგებული გახადოთ სამუშაოდ.

მონიშვნის ინსტრუმენტების და ბრძანებების კარგად ფლობა ძალიან მნიშვნელოვანია შემდეგი ეტაპისთვის, იქნება ეს ფერთა კორექცია თუ ობიექტის განცალკევება ფონისგან, მისი წაშლა თუ სხვა ფენაზე ან ფაილში გადატანა. ზემოთ ჩამოთვლილი ინსტრუმენტების დახმარებით თქვენ შეგიძლიათ შექმნათ მონიშნული არე, დააზუსტოთ მისი საზღვრები და გადახვიდეთ გამოსახულებასთან მუშაობის შემდეგ ეტაპზე:

### მოქმედები მონიშნულ არეზე

როდესაც გამოსახულებაზე გვაქვს მონიშნული არე, მასთან მიმართებით შეგვიძლია სხვადასხვა მოქმედებები შევასრულოთ:

**გადატანა** - **გადაადგილების** (Move) ინსტრუმენტის დახმარებით შეგვიძლია გამოსახულებაზე მონიშნულ ფრაგმენტს ადგილმდებარეობა შეუცვალოთ;

**კოპირება** - Edit -> Copy (Ctrl + C) მონიშნული ფრაგმენტის ასლის შექმნა;

**ამოჭრა** -> Edit -> Cut (Ctrl + X) მონიშნული ფრაგმენტის ამოჭრა;

**ჩასმა** -> Edit -> Paste (Crl + V) კოპირებული ან ამოჭრილი ფრაგმენტის ჩასმა (სხვა ფენაზე, სხვა ფაილში).

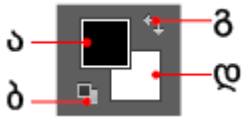
## დაფერვის მეთოდები

გამოსახულებასთან მუშაობა რასაკვირველია არ შემოიფარგლება მხოლოდ მისი მონიშვნით და სტანდარტული მოქმედებების შესრულებით (ამოჭრა, კოპირება და სხვა). ჩვენ ასევე შეგვიძლია მოვახდინოთ გამოსახულების დაფერვა. ილუსტრატორები საერთოდ სუფთა ტილოდან იწყებენ მუშაობას და საოცარ შედეგებს აღწევენ. ბუნებრივია აქ მხოლოდ პროგრამის კარგად ცოდნა არ არის საკმარისი. გარდა ამისა, დაფერვა შეგვიძლია გამოვიყენოთ როგორც მთლიანად გამოსახულებასთან, ისე მონიშნულ არესთან.

პროგრამაში დაფერვის უამრავი ხერხი არსებობს. სანამ ამ მეთოდებს განვიხილავთ, მინდა დავიწყო ჯერ ფერების განსაზღვრით.

## ფერების განსაზღვრა პროგრამაში

ფოტოშოპში მუშაობისას გამოიყენება ორი ფერი: მთავარი ანუ წინა ფონის ფერი (Foreground Color) და უკანა ფონის (Background Color) ფერი. ინსტრუმენტების პანელის ქვედა ნაწილში მდებარეობს ფერის მართვის ელემენტები (ორი ერთმანეთზე დადებული კვადრატი):

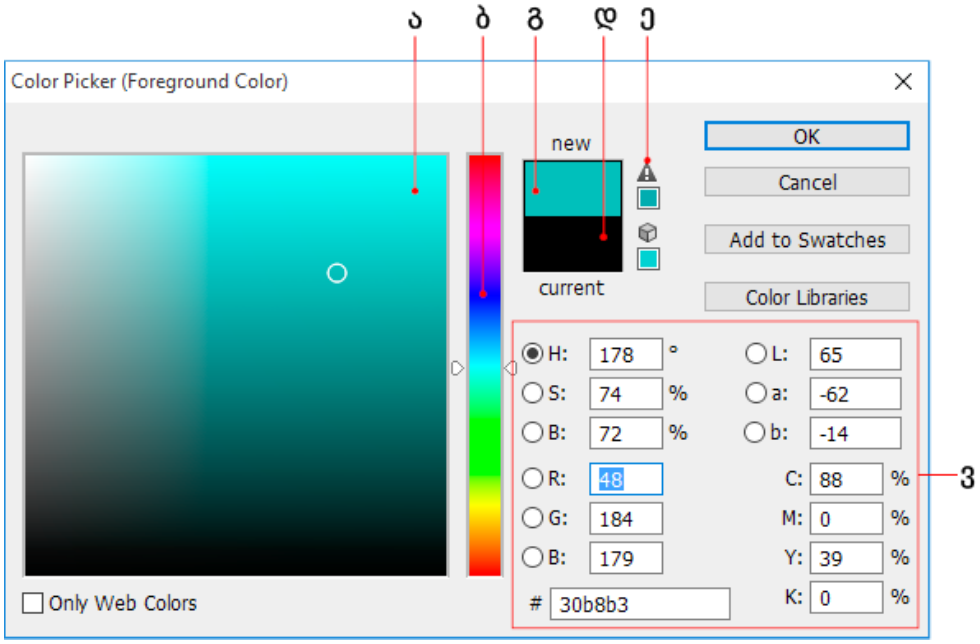



სურ. 1.2-11. ა - წინა ფონის ფერი (Foreground Color); ბ - საწყის მდგომარეობაში დაბრუნება (წინა ფონი შავი, უკანა - თეთრი); გ - წინა და უკანა ფონების ადგილების გადანაცვლება; დ - უკანა ფონის ფერი (Background Color).

რას ნიშნავს წინა და უკანა ფერები?

ინსტრუმენტები, როგორებიცაა ფუნჯი, ფანქარი, ფერის ჩასხმა და სხვა მსგავსი ინსტრუმენტები იყენებენ წინა ფონის ფერს. როდესაც ხდება სურათზე მონაკვეთის წაშლა, ამოჭრა და სხვა მსგავსი მოქმედება, ამ მონაკვეთების ჩანაცვლება ხდება უკანა ფონის ფერით.

რომელიმე კვადრატზე დაჭერისას იხსნება ფერების პალიტრა, საიდანაც ვირჩევთ სამუშაო ფერს.

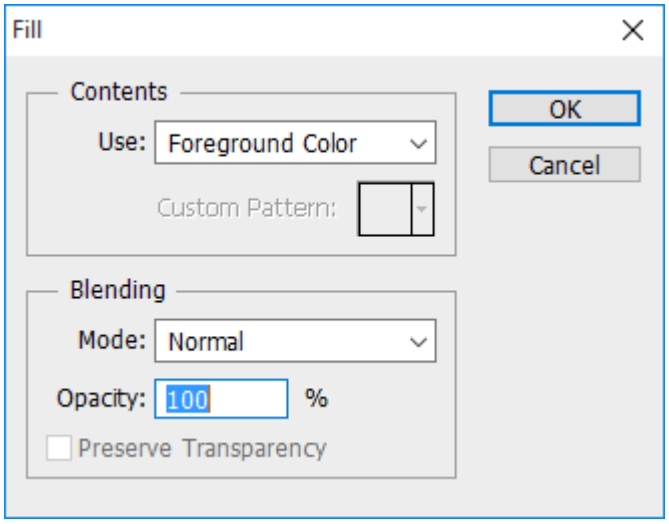



სურ. 1.2-12. ა - ფერთა ველი, რომელზეც პატარა წრით მონიშნულია შერჩეული ფერი; ბ - ფერთა შკალა, სადაც ვირჩევთ ფერს მოცემული სპექტრიდან; გ - ახალი შერჩეული ფერი; დ - მიმდინარე ფერი; ე - ნიშანი მიუთითებს, რომ ეს ფერი ვერ თავსდება CMYK ფერთა მოდელის სპექტრში; ვ - ფერთა მნიშვნელობები (ფერის შერჩევისას აისახება მისი რიცხობრივი მნიშვნელობები HSB, Lab, RGB, CMYK ფერთა მოდელებისთვის. ასევე მისი მნიშვნელობა თექვსმეტობით ფორმატში).

### *შევსება (Fill)*

**შევსება** შეიძლება გამოყენებული იყოს ცარიელი ადგილების შესავსებად ან გამოსახულებიდან არასასურველი ელემენტების მოსაცილებლად.

ამისათვის Fill ბრძანება უნდა გამოიძახოთ: Edit -> Fill (Shift + F5).




სურ. 1.2-13. დიალოგური ფანჯარა Fill

**გამოყენების** (Use) ჩამოსაშლელი მენიუდან ირჩევთ შევსების მეთოდს:

Foreground Color, Background Color, Color... - ვიყენებთ მაშინ, როდესაც არე გვინდა ფერით შევავსოთ;

Content Aware, Patern, History - ვიყენებთ მაშინ, როდესაც გვინდა არე გამოსახულებით ან რაიმე ტექსტურით შევავსოთ. ცალკე ყურადღებას იმსახურებს Content Aware.


ვიდეო 1.2-1. Content Aware-ის გამოყენება Photoshop CS6-ში

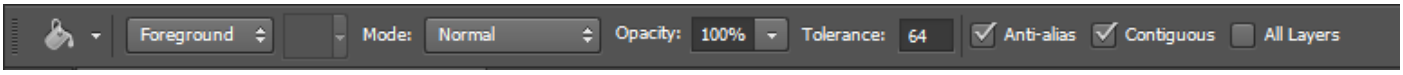
**შენიშვნა:** *Content Aware აქტიურია მხოლოდ მაშინ, როდესაც გამოსახულებაზე რამე ფრაგმენტია მონიშნული.*

Black, 50% Gray, White - სტანდარტული ფერებით შევსება.

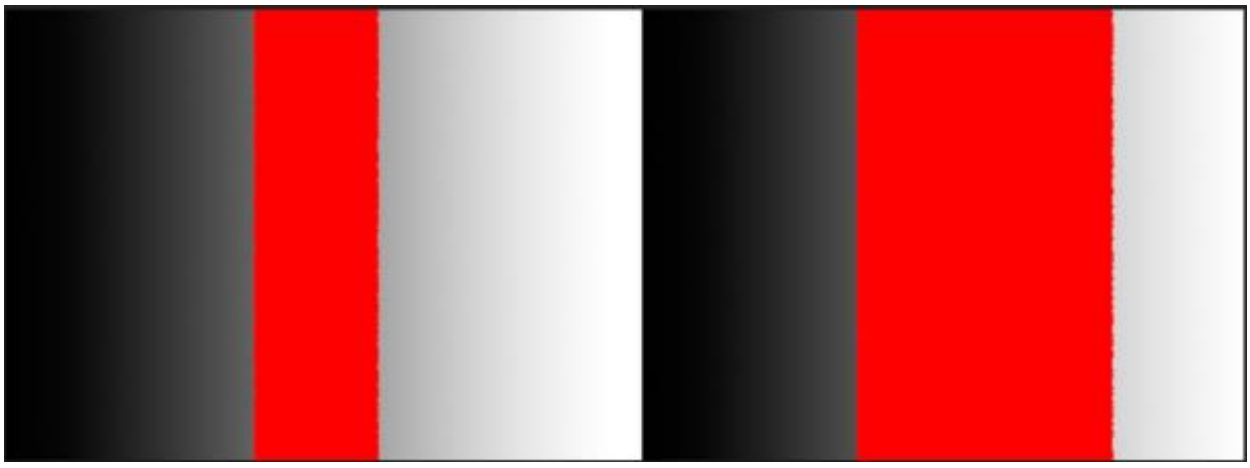
**შერევისა** (Blending) და **გამჭვირვალობის** (Opacity) დახმარებით შეგიძლიათ მართოთ **შევსების** პროცესი.

### *ფერის ჩასხმის ინსტრუმენტი (Paint Backet Tool)*

მოქმედების პრინციპით ეს ინსტრუმენტი ძალიან გავს **ჯადოსნურ ჯოხს** (Magic Wand).




ფორმატირების ზოლზე თქვენ შეხვდებით ისეთივე ფუნქციებს როგორიცაა **შეღწევადობა** (Tolerance), **შერბილება** (Anti-alias), **მომიჯნავე პიქსელები** (Contiguous) და **ყველა ფენა** (All Layers). დანარჩენი ფუნქციები კი დამახასიათებელია ამ ინსტრუმენტისთვის: შეგიძლიათ აირჩიოთ ინსტრუმენტმა **ფერი** გამოიყენოს თუ **ტექსტურა** (ჩამოსაშლელი მენიუ - Foreground / Patern), აირჩიოთ **შერევის** (Mode) მეთოდები და **გამჭვირვალობა** (Opacity).



სურ. 1.2-14. სურათზე ნაჩვენებია ინსტრუმენტის მოქმედება შეღწევადობის (Tolerance) სხვადასხვა მაჩვენებლებით. მარცხენა სურათზე მაჩვენებელი მითითებულია 32, ხოლო მარჯვენაზე 64.

### *გრადიენტის ინსტრუმენტი (Gradient Tool)*

გრადიენტის შექმნაში მონაწილეობს მინიმუმ ორი ფერი. ამ დროს პირველი ფერი ნელ-ნელა გადადის მეორეში.

გრადიენტის ინსტრუმენტის არჩევისას ფორმატირების პანელზე მისთვის დამახასიათებელი ფუნქციები ჩნდება.


აქ შეგიძლიათ აირჩიოთ გრადიენტის შეფერილობა და მისი ნაირსახეობები, დააყენოთ შერევის რეჟიმი, გამჭვირვალობა და სხვა.

გრადიენტი არის ხუთი სახის:

სურ. 1.2-15. მარცხნიდან მარჯვნივ: ხაზოვანი (Linear), წრიული (Radial), კუთხოვანი (Angular), არეკლილი (Reflected) და რომბისებრი (Dimond).

პროგრამაში მოცემული გამზადებული მაგალითების გარდა თქვენ თვითონაც შეგიძლიათ შექმნათ სასურველი მიმართულების გრადიენტი ფერების სასურველი რაოდენობით.

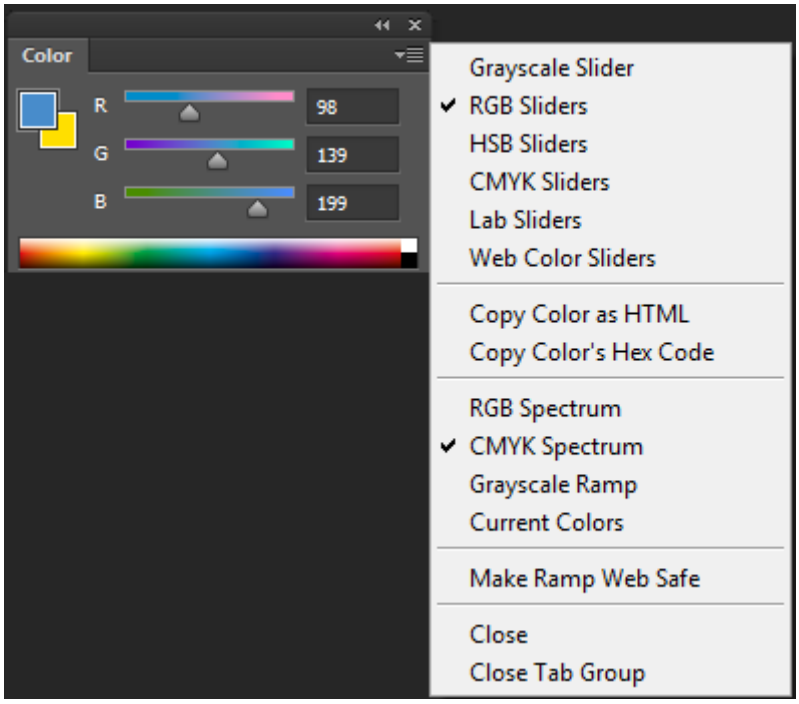

სურ. 1.2-16. გრადიენტის გამოყენების ერთ-ერთი მაგალითი

### *ფერთა ნიმუშები (Swatches)*


მისი გამოძახება ხდება Windows -> Swatches. აქ მოცემული ძირითადი სამუშაო ფერები. გარდა ამისა მუშაობის პროცესში თქვენ შესაძლოა შექმნათ საკუთარი ფერების პალიტრა. რომ არ იზეპიროთ მათი რიცხვითი მნიშვნელობები, შეგიძლიათ შეინახოთ ნიმუშად და შემდეგში გამოიყენოთ.

ფერი დასამატებლად რამდენიმე გზა არსებობს. ყველაზე მარტივია გამოიძახოთ ფერთა პალიტრა (იხ. სურ. 1.2-12) და ფერის შერჩევის შემდეგ დააწვეთ ღილაკს **ნიმუშებში დამატება** (Add to Swathes).

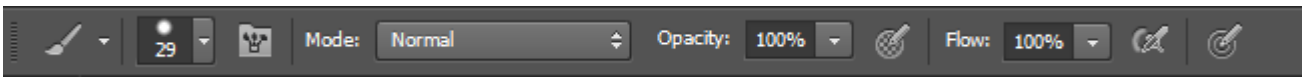
როდესაც **ფერთა ნიმუშები პანელიდან** აირჩევთ ფერს ის ჯდება როგორც წინა ფონის ფერი (Foreground Color). გარდა ამისა ის ასევე გამოისახება **ფერების** (Color) პანელზე. ამ პანელის (ისევე როგორც ფერთა პალიტრის) დანიშნულებაა შექმნათ ახალი ფერი. პანელის დამატებით მენიუში შეგიძლიათ აირჩიოთ, რომელი ფერთა მოდელისთვის ქმნით ფერს და შემდეგ მცოცავების გადაადგილებით ან გრაფებში მნიშვნელობების მითითებით, ადგენთ ახალ ფერს.



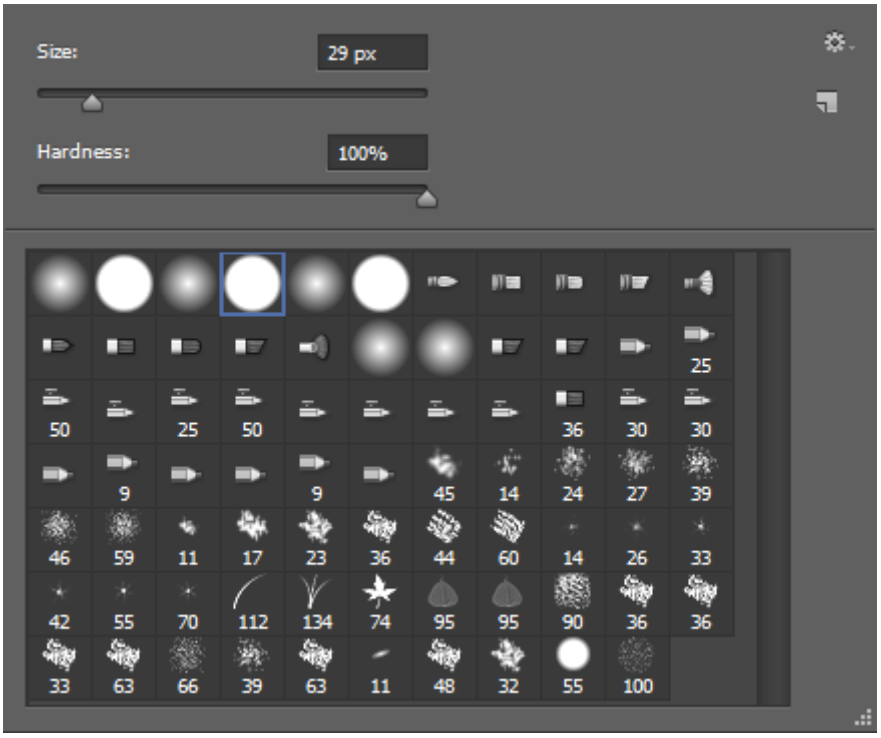

სურ. 1.2-17. პანელი **ფერი** (Color) და მისი დამატებითი მენიუ. ფერის შექმნა შეგიძლიათ როგორც წინა ფონისთვის (Foreground), ისე უკან ფონისთვის (Background)

### ფუნჯი (Brush)

ფუნჯი საკმაოდ ხშირად გამოყენებადი ინსტრუმენტია. შეიძლება ითქვას უნივერსალურიც კი: შეგიძლიათ უცვალოთ ფორმა, ფერი, ზომა, შექმნათ საკუთარი ფუნჯი. გარდა ამისა ის შეიძლება გამოიყენოთ ობიექტისთვის ნიღბისა (სწრაფი ნიღბის რეჟიმში) და კონტურის შესაქმნელად.




ფორმატირების ზოლიდან თქვენ შეგიძლიათ დააყენოთ მისი ზომა, აირჩიოთ ფუნჯის ტიპი, დააყენოთ ფერის შერევის რეჟიმები, გამჭვირვალობა და სხვა.



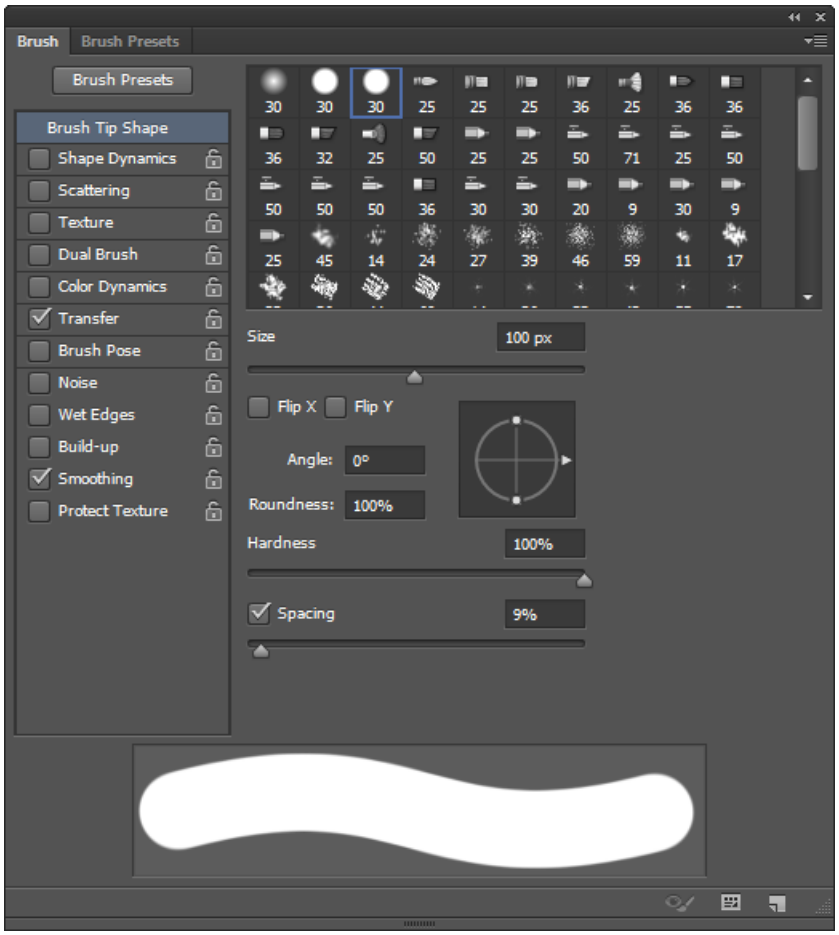

სურ. 1.2-18. აქ შეგიძლიათ აირჩიოთ ფუნჯის ტიპი. ასევე დააყენოთ მისი ზომა და სიმკვეთრე

შერჩეული ფუნჯისთვის შეგიძლიათ ზომა (px) და კიდეების სიმკვეთრე/სიმსუბუქე განსაზღვროთ: 100% - მკვეთრი, 0% - მსუბუქი.

სურ. 1.2-19. ერთნაირი დიამეტრი, მაგრამ სხვადასხვა სიმკვეთრის კიდეებით. პირველი ხაზი - 100%, მეორე ხაზი - 50%, მესამე ხაზი - 0%.

**შენიშვნა:** *ზოგი ფუნჯი ეფექტურია გამოიყენოთ სახატავი დაფითა და ფუნჯით სარგებლობისას.*

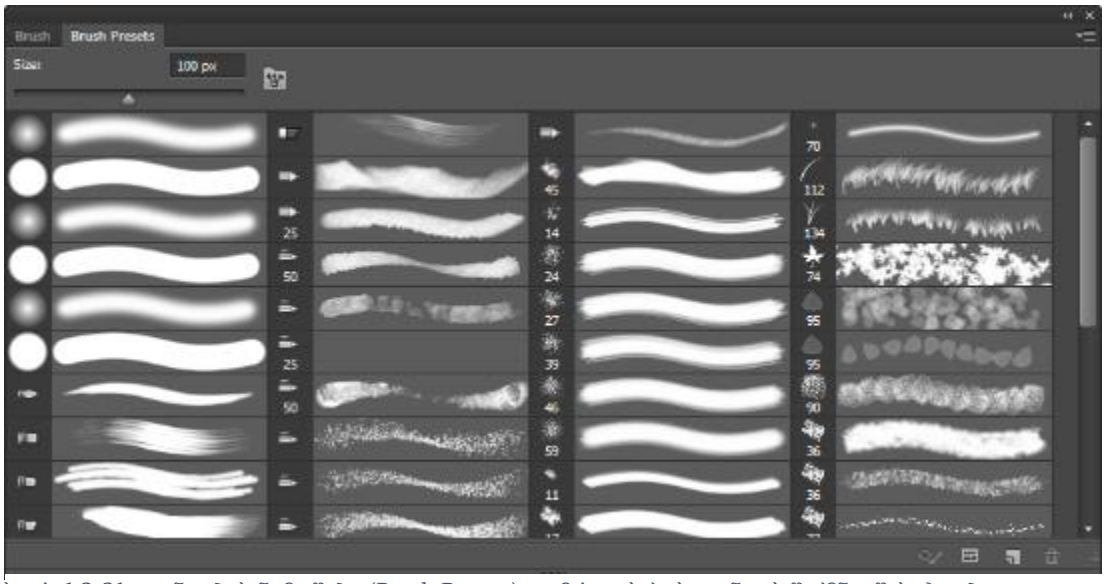
ძირითადი მახასიათებლების გარდა ასევე შეგიძლიათ ფუნჯის სხვა პარამეტრებიც ცვალოთ. ამისათვის გამოიძახეთ Window -> Brushes (F5).




სურ. 1.2-20. ფანჯარაში Brush შეგიძლიათ როგორც ფუნჯის არჩევა, ისე მისი მახასიათებლების დაყენება: სიმკვეთრე, დახრის კუთხე, ინტენსივობა და სხვა.

არჩეული ფუნჯისთვის უამრავი მახასიათებლის დაყენება შეგიძლიათ, ხოლო ფანჯრის ქვემოთ მოცემულია წინასწარი ხედის ფანჯარა, სადაც დაინახავთ, როგორ იცვლება ფუნჯი. სამომავლოდ ახალი ფუნჯის სახით შეგიძლიათ შეინახოთ.

ამავე ფანჯარაზე არის ჩანართი Brush Presets სადაც მოცემულია ფუნჯის მზა მაგალითები. ამავე ფანჯრის გამოძახება შეგიძლიათ Windows -> Brush Presets




სურ. 1.2-21. ფუნჯების ნიმუშები (Brush Presets). უამრავი სახის ფუნჯის შექმნაა შესაძლებელი.

გარდა არსებული ფუნჯებისა, რაც პროგრამას ახლავს თან, თქვენ შეგიძლიათ ჩატვირთოთ სხვა ფუნჯებიც, რომლებიც ინტერნეტში სხვადასხვა გრაფიკულ რესურსებზე მოიპოვება. ამისათვის ფორმატირების ზოლზე, ფუნჯების (Brushes) ნაირსახეობების ფანჯარაში (სურ. 1.2-18), დამატებით მენიუში აირჩიოთ ფუნჯების ჩატვირთვა (Load Brushes…) და გამოსულ ფანჯარაში მიუთითეთ ფუნჯის ადგილმდებარეობა (ფუნჯის ფაილის ფორმატია *.ABR).

თუ დაგიგროვდათ ძალიან ბევრი ფუნჯი და უკვე რთულია მათ შორის ორიენტირება, იმავე მენიუში მიუთითეთ ფუნჯების ჩამოყრა (Reset Brushes…) და ფუნჯების რაოდენობა და ტიპები დაუბრუნდება საწყის მდგომარეობას.

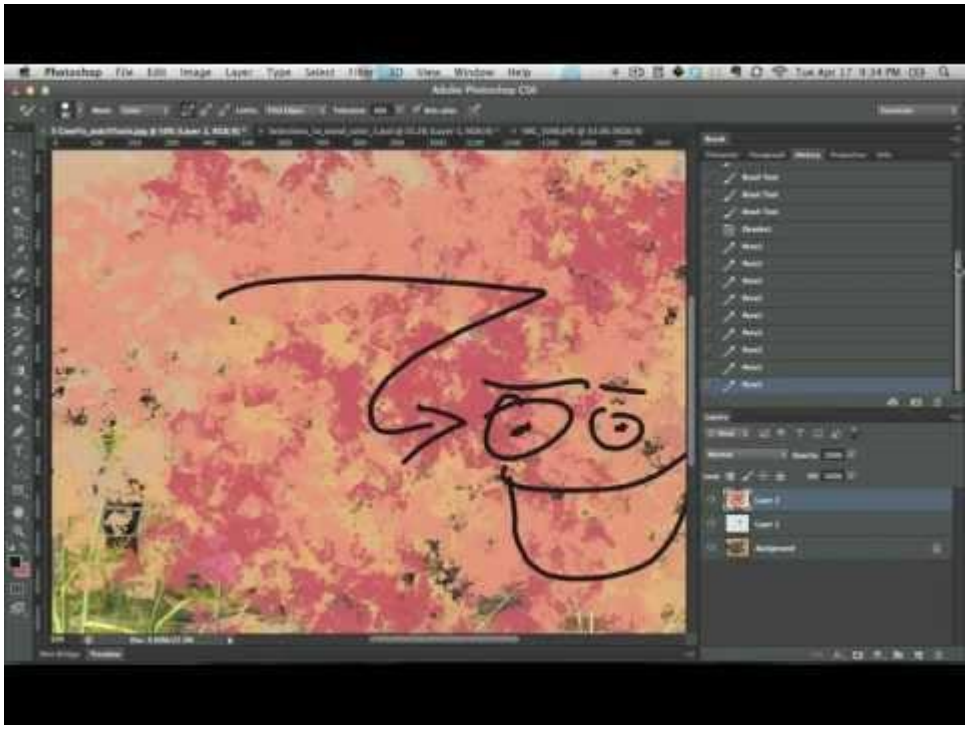
თქვენს მიერ შექმნილი ფუნჯების ნაკრები თუ გინდათ, რომ გაუგზავნოთ მეგობარს ან გადაიტანოთ სხვა კომპიუტერში, მენიუში აირჩიეთ ფუნჯების შენახვა (Save Brushes…). მიუთითეთ მისამართი, ფაილის დასახელება და მიღებული ფაილი (რომელშიც მოთავსებული ფუნჯები), შეგიძლიათ გაუზიაროთ სხვას.

ფუნჯის შექმნა ნებისმიერი გამოსახულებიდან შეიძლება, თუ ის აკმაყოფილებს ორ პირობას: გამოსახულების ზომა არ უნდა აღემატებოდეს 2500 X 2500 პიქსელს და უნდა ჰქონდეს გამჭვირვალე ან თეთრი ფონი.

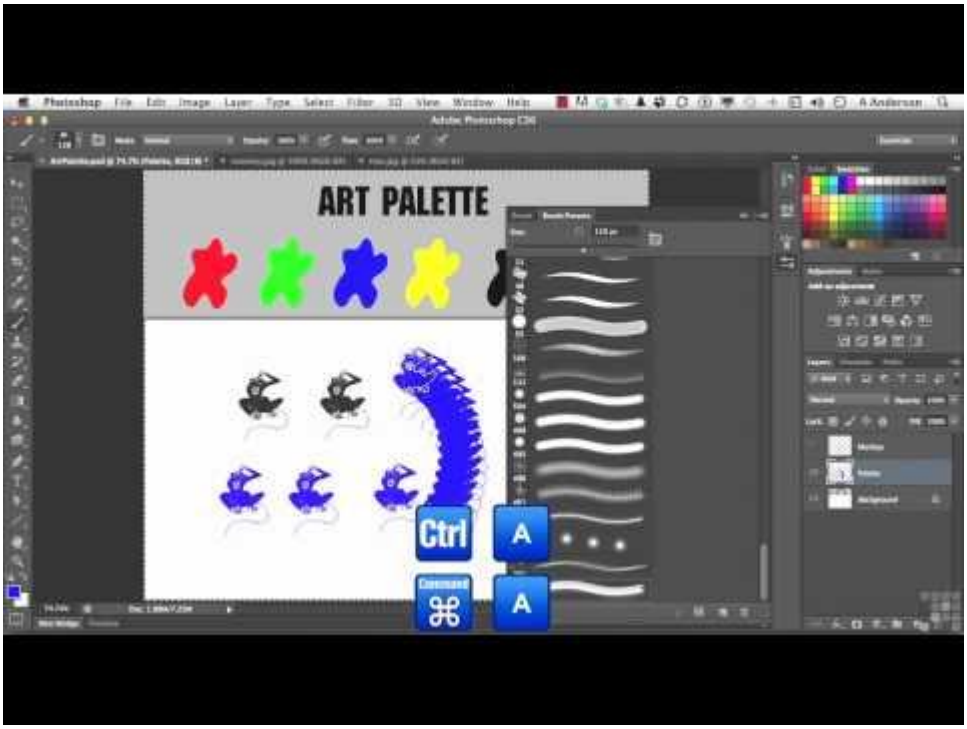
ფუნჯისათვის გამოსახულების მომზადების შემდეგ, Edit მენიუდან უნდა გავააქტიუროთ Define Brush Preset ბრძანება. ეკრანზე გამოვა სადიალოგო ფანჯარა Brush Name, რომელშიც იწერება შექმნილი ფუნჯის სახელი და ვადასტურებთ OK საბრძანებო ღილაკით.

ჩვენს მიერ შექმნილი ფუნჯი დაიკავებს ადგილს ფუნჯთა მენიუში, ფუნჯთა ნაკრების ბოლოს.

ფუნჯის მსგავსად სხვა ინსტრუმენტებიც იყენებენ ისეთ პარამეტრებს, როგორებიცაა ფუნჯის ზომა, სიმკვეთრე, დახრილობა და სხვა. ესენია: რეტუშირებისა და ხატვის ინსტრუმენტები (სურ. 1.1-9).



ვიდეო 1.2-2. ფუნჯების გამოყენება Photoshop CS6-ში



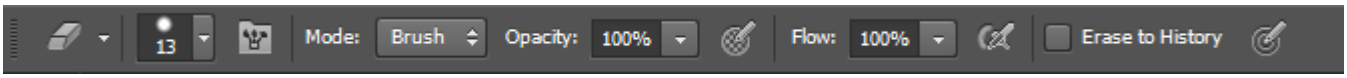

ვიდეო 1.2-3. ფუნჯის შექმნა პროგრამა Photoshop CS6-ში

## საშლელი (Eraser Tools)

გამოსახულების რედაქტირებისას დაფერვის მსგავსად ჩვენ დაგვჭირდება რაიმე ფრაგმენტის წაშლა. ინსტრუმენტთა პანელზე არის სამი სახის საშლელი: ჩვეულებრივი საშლელი (Eraser Tool), ფონის საშლელი (Background Eraser Tool) და ჯადოსნური საშლელი (Magic Eraser Tool).

სამივე ინსტრუმენტს ფორმატირების ზოლზე თავისთვის დამახასიათებელი ფუნქციები გააჩნია:

### *ჩვეულებრივი საშლელი (Eraser Tool)*




ფორმატირების ზოლიდან თქვენ შეგიძლიათ სხვადასხვა მახასიათებლების დაყენება: ზომა, საშლელის ტიპი, გამჭვირვალობა, დაჭერის ინტენსივობა და სხვა. ისევე, როგორც ფუნჯის შემთხვევაში, შეგიძლიათ დააყენოთ ფუნჯის ზომა და კიდეების სიმკვეთრე.

საშლელის მოქმედება დამოკიდებულია თუ რომელ ფენაზე (Layer – სამუშაო ფენა; Background - ფონის ფენა) ხდება მისი გამოყენება. გამოსახულების უკანა ფონზე (Background) მუშაობისას, წაშლილ პიქსელებს ჩაანაცვლებს უკანა ფონის ფერის (Background Color) პიქსელებით. თუ ფონის ფენას სამუშაო ფენად (Layer) გადავაქცევთ ( Layer + New + Layer From Background), საშლელი დაიწყებს პიქსელების წაშლას სრულ გამჭვირვალობამდე (დარჩება ჭადრაკისებრი ფონი). აღნიშნულ ინსტრუმენტს ფერის შეგრძნება არ გააჩნია და შლის ყველაფერს, რაც კურსორის ქვეშ მოხვდება.

საშლელი სამ რეჟიმში მუშაობს: Brush, Pencil და Block, რომელთა არჩევა ფორმატირების ზოლზე Mode ჩამოსაშლელი მენიუდან შეიძლება. Brush ან Pencil რეჟიმების არჩევის შემთხვევაში, საშლელისთვის ხელმისაწვდომი ხდება აღნიშნული ინსტრუმენტის პარამერტები. Block რეჟიმში ხელსაწყო კვადრატის ფორმას ღებულობს, მისი ზომა გამოსახულების ზომაზე არის დამოკიდებული და პარამეტრების პანელზე ყველა პარამეტრი დაბლოკილია.

### *ფონის საშლელი (Background Eraser)*


ფონის საშლელის პარამეტრების პანელი ძალიან ჰგავს ფერის ჩანაცვლების პარამეტრების პანელს და მოხმარების წესიც ანალოგიურია. განსხვავებული მხოლოდ შედეგია: ფერის ჩანაცვლების ინსტრუმენტი ფერს უცვლის პიქსელებს, ხოლო ფონის საშლელი შლის პიქსელებს.

ფუნქცია **წინა ფონის ფერის დაცვა** (Protect Foreground Color) ჩამრთველი იცავს წაშლისაგან გამოსახულების იმ ფერს, რომელიც წინა ფონის ფრად (Foreground Color) არის დაფიქსირებული.

სურ. 1.2-22. ფონის საშლელის (Background Eraser Tool) გამოყენების

### *ჯადოსნური საშლელი (Magic Eraser Tool)*


ჯადოსნური საშლელი გამოიყენება გამოსახულებაზე ერთფეროვანი არეების მარტივად და სწრაფად წასაშლელად. მისი მოქმედების პრინციპი ისეთივეა, როგორიც ჯადოსნურ ჯოხს (Magic Wand) გააჩნია. და ფორმატირების ზოლზე მსგავსი ფუნქციები აქვს. თუმცა მონიშნული არის შექმნის ნაცვლად, საშლელი შლის გამოსახულების ფრაგმენტებს.

სურ. 1.2-23. ჯადოსნური საშლელის (Magic Eraser) გამოყენების მაგალითი.

## 1.3. მარტივი გამოსახულების რედაქტირება

პარაგრაფის შესაბამისი თემატიკა

- ტრანსფორმაციის ინსტრუმენტები და ბრძანებები
- გამოსახულების ზომის რედაქტირება - გამოსახულებისა და ტილოს ზომა, შემოჭრის ინსტრუმენტი
- ფენების შექმნა და მათი მართვა. ფენების ნაირსახეობები
- ფენების სტილების მიმოხილვა

### ტრანსფორმაციის ინსტრუმენტები

მუშაობის პროცესში აუცილებლად დაგჭირდებათ ისეთი მოქმედებების შესრულება, როგორიცაა ობიექტის შემოტრიალება, ზომებისა თუ ფორმის შეცვლა. ამისათვის გამოიყენება ტრანსფორმაციის ფუნქცია.

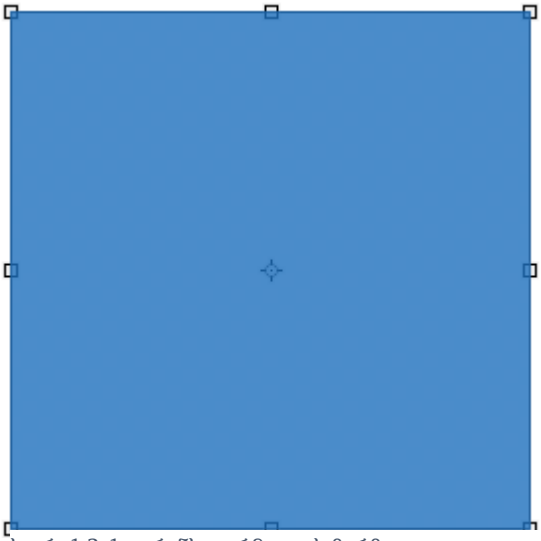
#### *თავისუფალი ტრანსფორმაცია*

თუ ობიექტი ძირითად ფონზეა განთავსებული (Background) ის უნდა მონიშნოთ (იხ. მონიშვნის მეთოდები, გვ. 23). ასევე ტრანსფორმაცია შეგიძლიათ გაუკეთოთ ცალკე ფენაზე მოთავსებულ ობიექტს, ობიექტთა ჯგუფს და ა.შ.

**თავისუფალი ტრანსფორმაციის** გამოძახება ხდება Edit -> Free Transform (Ctrl +T).


ფორმატირების ზოლზე რიცხობრივი მნიშვნელობების მითითებით შეგიძლიათ ცვალოთ ობიექტის მდებარეობა, მისი ზომები, მასშტაბურობა, დახრილობის კუთხე.

მონიშნული ობიექტი (რა ფორმისაც არ უნდა იყოს) თავსდება მართკუთხედში ე.წ. **ტრანსფორმაციის ჩარჩოში**, რომელსაც ნაპირებზე და კუთხეებში გააჩნია წერტილები. მათი საშუალებით ხდება ობიექტზე ზემოქმედება. სულ ცხრა ასეთი წერტილია. ცენტრში (ნაგულისხმევად) არის კიდევ ერთი წერტილი. მას **საკონტროლო წერტილს** უწოდებენ. ის განსაზღვრავს თუ რის მიმართ უნდა მოხდეს ტრანსფორმაცია, მაგალითად ობიექტის დახრა, სარკისებურად შებრუნება ან შემოტრიალება. სხვა წერტილებისგან განსხვავებით მისი გადაადგილება თავისუფლად შეგიძლიათ როგორც ჩარჩოს შიგნით, ისე მის გარეთ.



სურ. 1.3-1. ტრანსფორმაციის ჩარჩო.

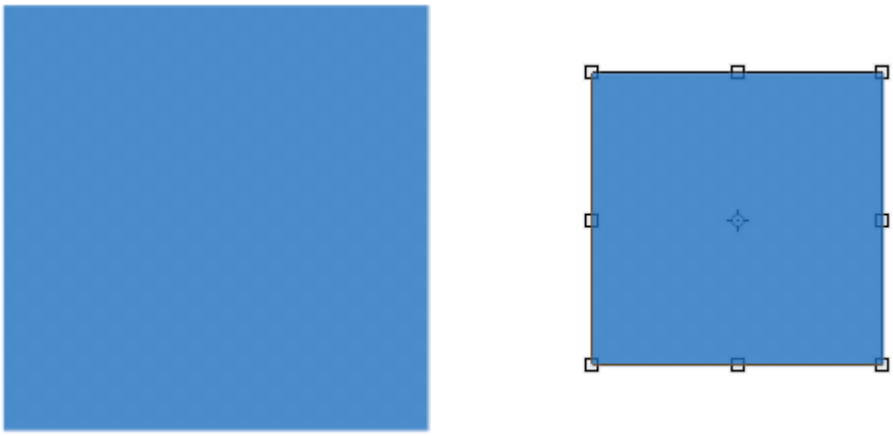
ობიექტზე ზემოქმედებისთვის კურსორი უნდა მიიტანოთ ერთ-ერთ წერტილთან ან კიდესთან. ის შეიცვლის ფორმას და ეს ფორმა მიგითითებთ მოქმედებაზე (ზომის, ფორმის შეცვლა ან შემოტრიალება).

თუ გამოიყენებთ Ctrl კლავიშს, მაშინ წერტილზე მოქმედებთ ინდივიდუალურად. Shift კლავიშის დახმარებით კი, ხდება ობიექტის ზომის ცვლილება პროპორციულად.

გარდა **თავისუფალი ტრანსფორმაციისა**, პროგრამაში ტრანსფორმაციის შესაძლებლობები ცალ-ცალკეა მოცემული. ისინი შეგიძლიათ იხილოთ Edit -> Transform მენიუში; ან, როდესაც ობიექტი მოთავსებულია **ტრანსფორმაციის ჩარჩოში**, ნებისმიერ ადგილას მაუსის მარჯვენა ღილაკზე დაჭერისას გამოსულ მენიუში:

*მასშტაბირება (Scale):*

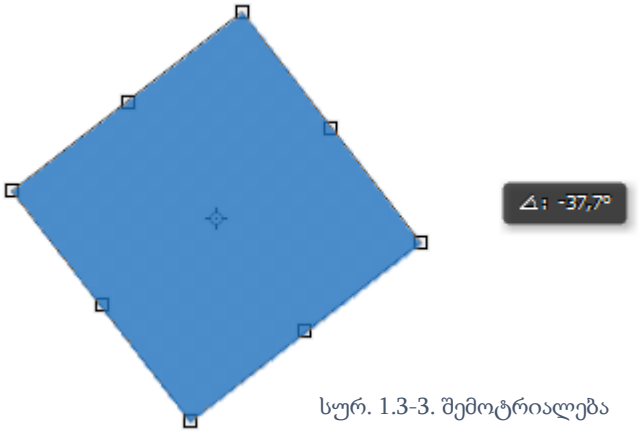
ობიექტის მასშტაბირება საკონტროლო წერილის მიმართ. მასშტაბირება შეიძლება როგორც ჰორიზონტალურად, ისე ვერტიკალურად ან ორივე მიმართულებით ერთდროულად;



სურ. 1.3-2. მასშტაბირება

*შემოტრიალება (Rotate):*

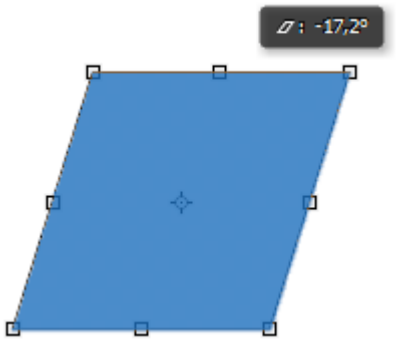
ობიექტის შემოტრიალება საკონტროლო წერტილის გარშემო. ნაგულისხმევად ეს წერტილი ცენტრშია მოთავსებული (სურ. 1.3-1);




სურ. 1.3-3. შემოტრიალება

*დახრა (Skew):*

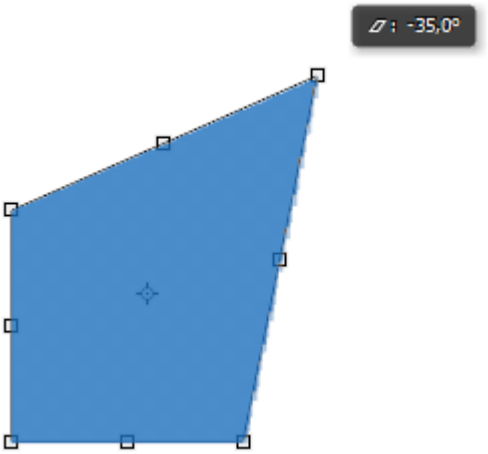
ობიექტის დახრა ჰორიზონტალურად ან ვერტიკალურად;




სურ. 1.3-4. დახრა

*დამახინჯება (Distort):*

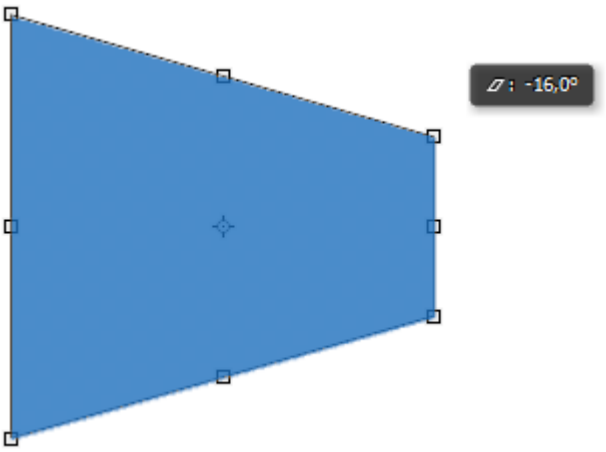
ობიექტის გაწელვა ყველა მიმართულებით;




სურ. 1.3-5. დამახინჯება

*პერსპექტივა (Perspective):*

ობიექტთან მიმართებაში გამოიყენება ერთ წერტილში შეკრების პერსპექტივა (ინფორმაცია პერსპექტივის შესახებ);

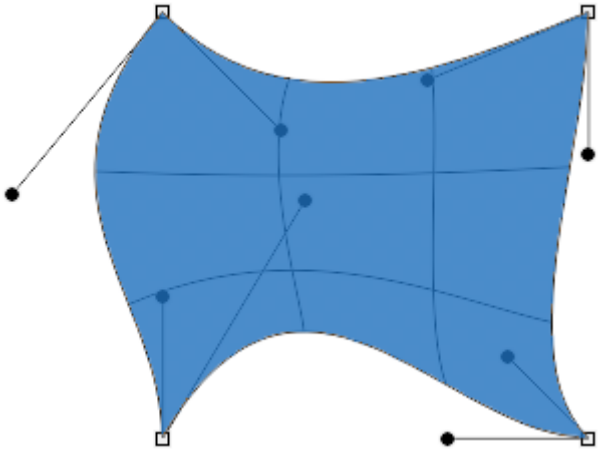



სურ. 1.3-6. პერსპექტივა

*დეფორმაცია (Warp):*

ობიექტის ფორმის შეცვლა; შეიცავს დამატებით მზა ფორმებს, როგორებიცაა თაღი, თევზი, დროშა და სხვა. ფორმატირების ზოლზე **დეფორმაცია** (Warp) ჩამოსაშლელი მენიუდან აირჩიეთ გამზადებული ფორმა. ამ დროს გამოჩენილ ბადეს გააჩნია ერთი საკვანძო წერტილი, რომლის დახმარებითაც თქვენ შეგიძლიათ ცვალოთ მისი ფორმა.

თუ მენიუდან აირჩევთ ფუნქციას **მორგებული** (Custom), გამოსახულებაზე დადებულ ბადეს ოთხი საკვანძო წერილი ექნება (კუთხეებში), რომელთა დახმარებითაც შეგიძლიათ თავისუფლად ცვალოთ ობიექტის ფორმა და მიანიჭოთ მას ნებისმიერი ფორმა.



სურ. 1.3-7. დეფორმაცია

*შემოტრიალება 180°, 90° საათის ისრის ან მის საწინააღმდეგო მიმართულებით:*

ობიექტი შემობრუნდება მოცემული კუთხით საათის ისრის ან მის საწინააღმდეგო მიმართულებით;

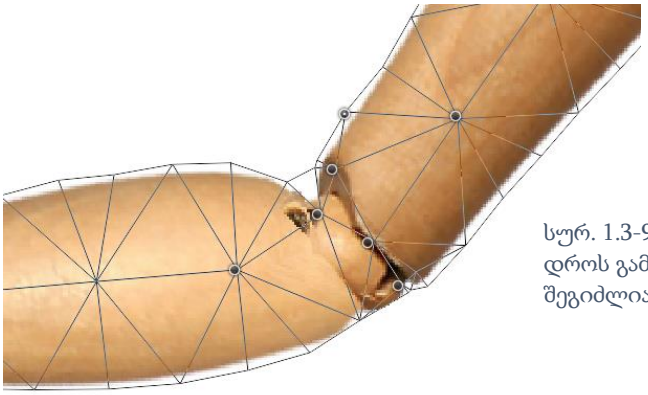
***ანარეკლი** (Flip Horizontal / Vertical):*

ობიექტის ანარეკლი ჰორიზონტალურ ან ვერტიკალურ სიბრტყეში.

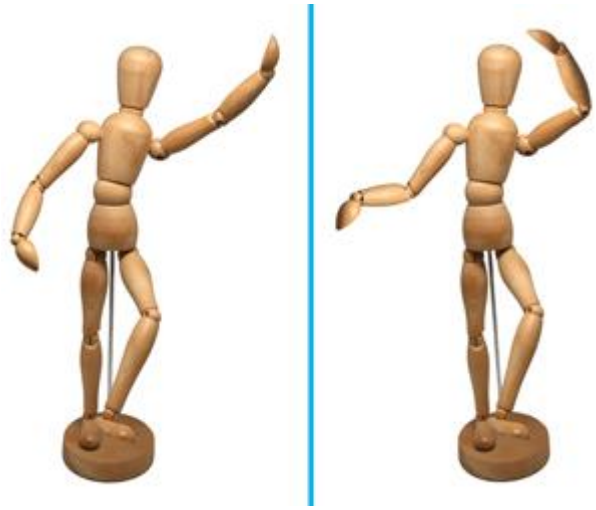
სურ. 1.3-8. ანარეკლი (ვერტიკალურად)

*მარიონეტული დეფორმაცია (Puppet Warp)*

ეს ობიექტის ტრანსფორმირების ერთ-ერთი ინსტრუმენტია. მისი გამოძახება ხდება Edit -> Puppet Warp. გამოსახულება ამ დროს იფარება დეტალური ბადით. ხაზების გადაკვეთის წერტილში კი შეგიძლიათ დეფორმაციის წერტილების დამატება (). წერტილების დახმარებით კი შეგიძლიათ ცვალოთ ბადის სტრუქტურა (წანაცვლება, შემობრუნება), რაც შესაბამისად გამოიწვევ ობიექტის ფორმის შეცვლას.



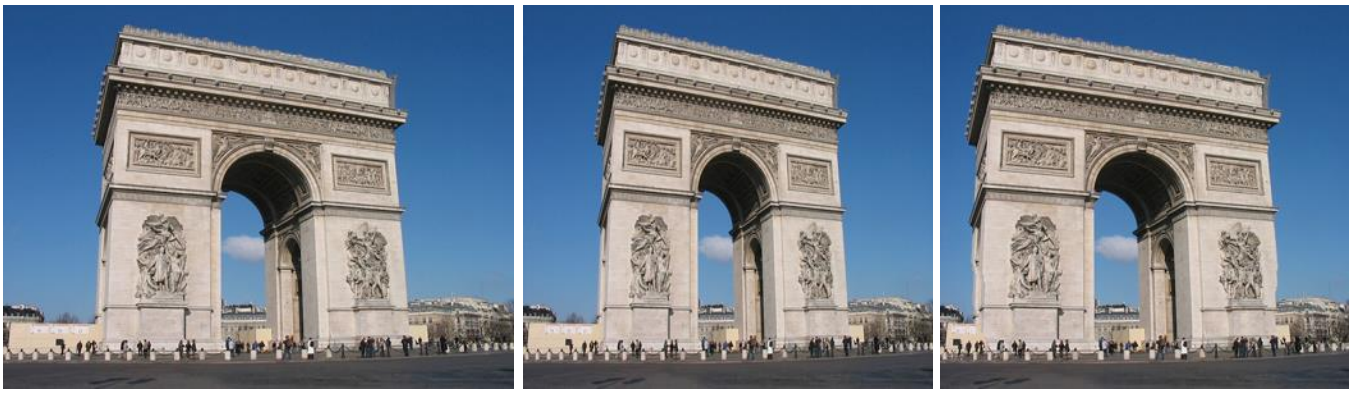
სურ. 1.3-9. მარიონეტული დეფორმაციის (Puppet Warp) დროს გამოსახულება იფარება ბადით, რომელზეც შეგიძლიათ დეფორმაციის წერტილების დამატება.



სურ. 1.3-10. ობიექტი ტრანსფორმაციამდე და ტრანსფორმაციის შემდეგ

### *მასშტაბირება შიგთავსის გათვალისწინებით (Content-Aware Scale)*

პროგრამა გამოსახულებას აანალიზებს და მისი ზომისა და/ან პროპორციების შეცვლას ამ ინფორმაციის საფუძველზე ახდენს. თუ ჩვეულებრივი ტრანსფორმაციის დროს ხდება ერთბაშად ყველა პიქსელების შემჭიდროვება, ასეთ შემთხვევაში ხდება მხოლოდ იმ პიქსელების შემჭიდროვება სადაც მინიმალური ან მსგავსი ინფორმაცია ინახება.



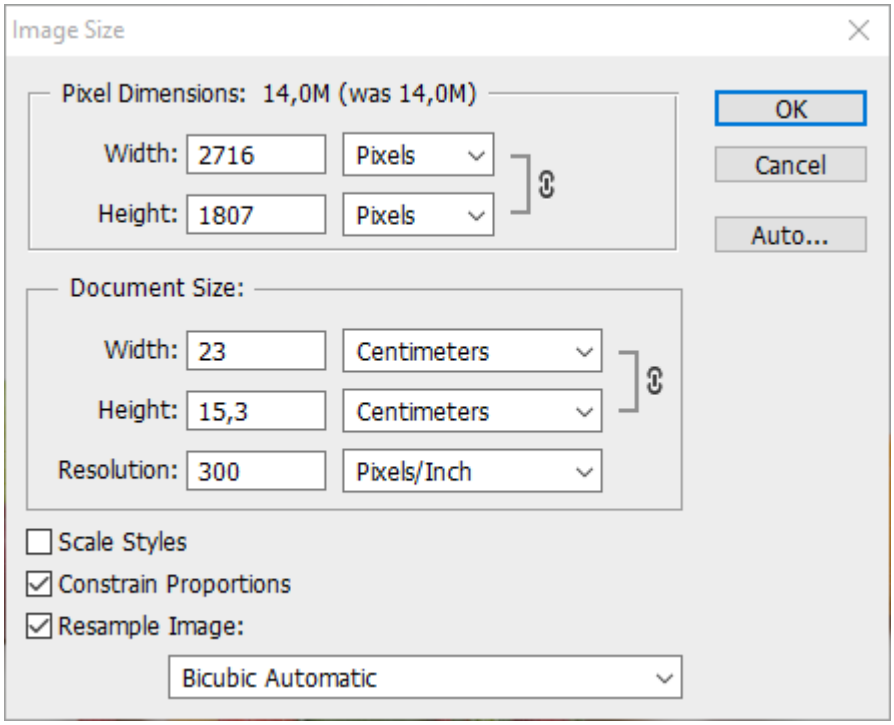
სურ. 1.3-11. პირველი საწყისი გამოსახულებაა. მეორე გამოსახულება შემჭიდროვებულია 20%-ით თავისუფალი ტრანსფორმაციის (Free Transform) გამოყენებით. მესამე გამოსახულება შემჭიდროვებულია 20%-ით მასშტაბირების შიგთავსის გათვალისწინების მეთოდით (Content-Aware Scale).

## გამოსახულების ზომების რედაქტირება

### *გამოსახულების ზომა (Image Size)*

გამოსახულების ელემენტებისა და ცალკეული ფრაგმენტების რედაქტირების გარდა, ჩვენ ასევე შეგვიძლია მისი ზომებისა და გარჩევადობის შეცვლა/რედაქტირება.

ნებისმიერ ფაილს, რომელსაც ჩვენ შექმნით თუ გავხსნით პროგრამაში გააჩნია ზომები და გარჩევადობის ხარისხი. მუშაობის პროცესში რომ ვაკონტროლოთ ზომები, ობიექტების განლაგება და სხვა მსგავსი პარამეტრები, შეგვიძლია ვისარგებლოთ სახაზავით. მისი გამოძახება/გათიშვა ხდება მენიუდან View -> Rulers (Ctrl + R). მაგრამ ეს არ არის საკმარისი იმისათის, რომ გავიგოთ დეტალური ინფორმაცია გამოძახებული ფაილის ზომების შესახებ. ამისათვის კი ვიძახებთ Image -> Image Size (Alt + Ctrl + I).

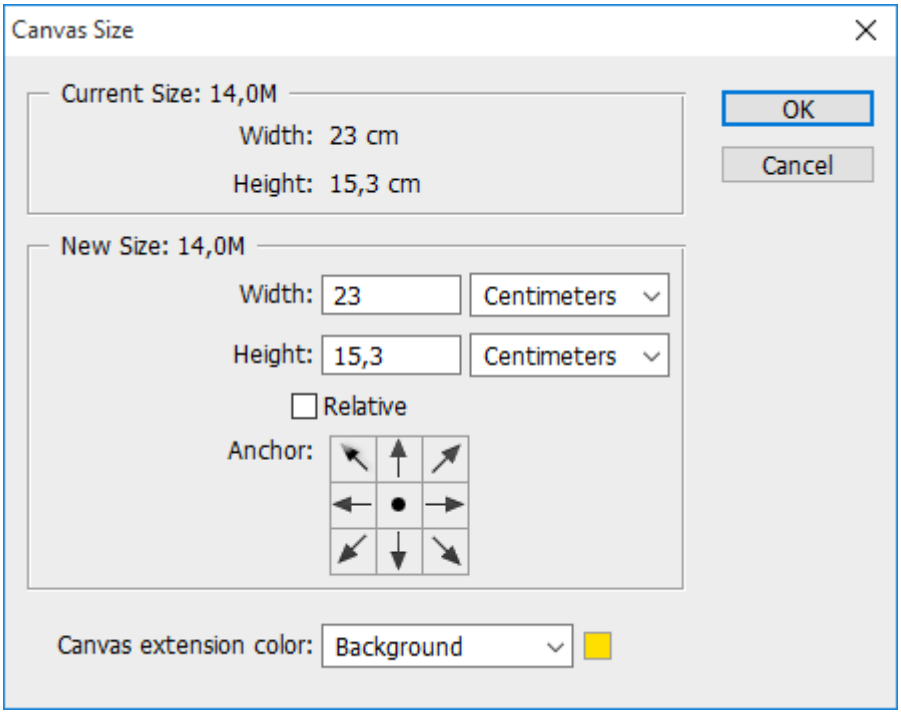



სურ. 1.3-12. გამოსახულების ზომა

აქ ჩვენ შეგვიძლია უკვე დეტალური ინფორმაცია ვიხილოთ ობიექტის შესახებ: მისი ზომა პიქსელებში და რომელიმე სხვა საზომ ერთეულში (სმ, მმ, პქს და სხვა), გარჩევადობის ხარისხი. ამავე ფანჯრიდან შეგვიძლია ვცვალოთ სიგანე, სიმაღლე ან ორივე მახასიათებელი ერთად (ასეთ შემთხვევაში უნდა იყოს მონიშნული **პროპორციების დაცვა** (Constrain Proportion).

### *ტილოს ზომა (Canvas Size)*

მუშაობის პროცესში ასევე დაგვჭირდება არამხოლოდ ფაილის ზომების ცვლილება, არამედ სამუშაო ტილოს გაზრდა. ამისათვის ვიძახებთ Image -> Canvas Size (Alt + Ctrl + C):




სურ. 1.3-13. ტილოს ზომის ცვლილება (Canvas Size)

ამ ფანჯრის დახმარებით ჩვენ ვცვლით ტილოს ზომას - ვუმატებთ ან ვაკლებთ სამუშაო სივრცეს სიმაღლეში, სიგანეში ან ორივე მიმართულებით.

### *შემოჭრის ინსტრუმენტი (Crop Tool)*

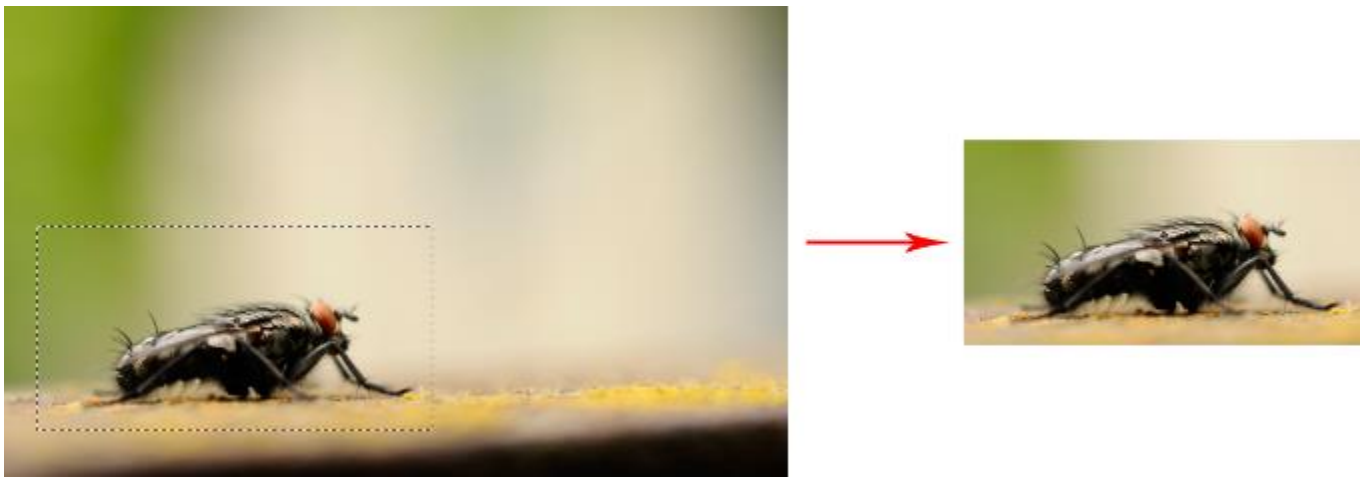
შემოჭრის ინსტრუმენტი გვეხმარება მოვაჭრათ გამოსახულებას ზედმეტი, არასასურველი არეები. როდესაც ამ ინსტრუმენტს ავირჩევთ, გამოსახულება მოთავსდება ჩარჩოში, რომელიც წააგავს ტრანსფორმაციისას. კიდეებზე ან კუთხეებზე თუ მოვკიდებთ კურსორს, შეგვიძლია მისი გადაადგილება (როგორც წესი ცენტრისკენ). ჩარჩოს გარეთ დარჩენილი გამოსახულება ჩამოიჭრება.

სურ. 1.3-14. გამოსახულების შემოჭრა. ჩარჩოს გარეთ მოხვედრილი სივრცე მოიჭრება, ხოლო ჩარჩოს შიგნით კი დარჩება.

გარდა ამისა, ფორმატირების ზოლზე მოცემული პარამეტრების დახმარებით ჩვენ შეგვიძლია ვმართოთ ჩამოჭრა: ზომები, დახრილობა და სხვა. საინტერესოა ჩამრთველი **ჩამოჭრილი პიქსელების წაშლა** (Delete Cropped Pixels). თუ ეს ფუნქცია ჩართულია, მაშინ ჩარჩოს გარეთ დარჩენილი პიქსელები წაიშლება, ხოლო თუ გამორთულია, მაშინ ისინი უბრალოდ დაიმალებიან. ხოლო მათი გამოჩენა შესაძლებელი იქნება თუ გამოიძახებთ ფუნქციას Image -> Reveal All.

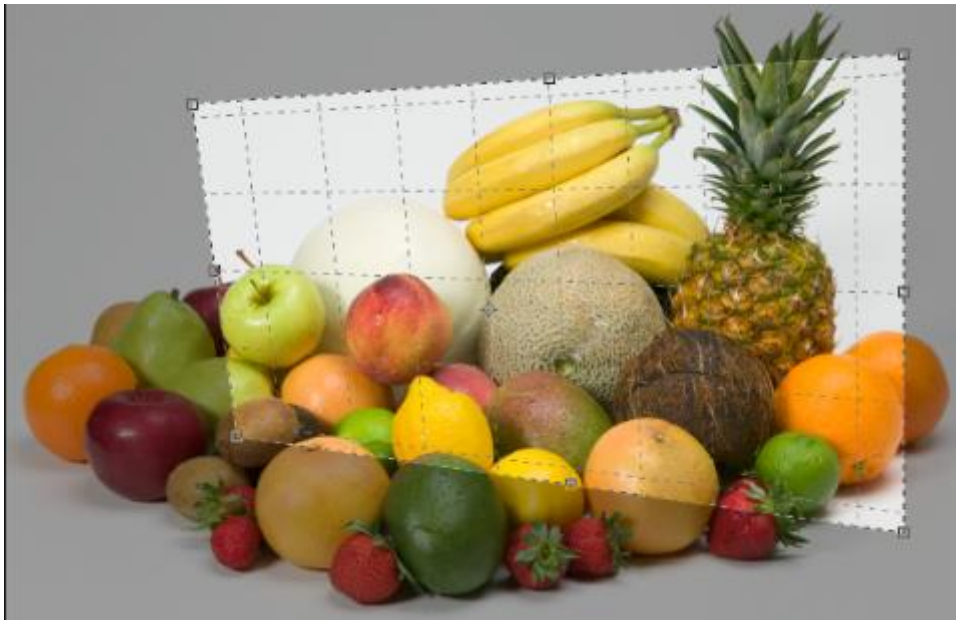
**შენიშვნა:** *გაითვალისწინეთ, რომ თუ პიქსელები დაიფარება და არა წაიშლება, ფაილის მოცულობა (დისკზე) არ შეიცვლება.*

**შემოჭრის** ინსტრუმენტის გარდა, ზედმეტი არეების მოსაჭრელად ასევე შეგიძლიათ გამოიყენოთ მართკუთხა მონიშვნის ინსტრუმენტი. მონიშნეთ სასურველი არე (რაც უნდა დარჩეს), შემდეგ გამოიძახეთ Image – Crop.



სურ. 1.3-15. მართკითხა მონიშვნის ინსტრუმენტის (Rectangular Marque Tool) გამოყენება გამოსახულების შემოსაჭრელად. ფოტო: გიორგი კევლიშვილი

პროგრამაში არსებობს ასევე მეორე სახის შემოჭრის ინსტრუმენტი, რომელსაც **პერსპექტივაში შემოჭრა** ეწოდება. გამოიყენება დეფორმირებული სურათების შემოსაჭრელად. მოქმედებით ის ძალიან წააგავს ჩვეულებრივ შემოჭრის ინსტრუმენტს. განსხვავება კი ისაა, რომ აქ შეგვიძლია ცალკეული (საკონტროლო) წერტილების გადაადგილება და შემოჭრის არისთვის ფორმის მიცემა



სურ. 1.3-16. პერსპექტივაში შემოჭრის ინსტრუმენტი (Perspective Crop Tool)

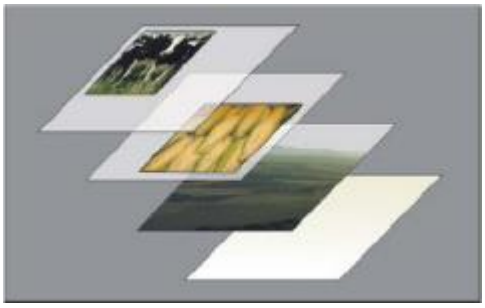
გარდა ამ მოქმედებისა, ჩვენ ასევე შეგვიძლია გამოსახულება დავატრიალოთ სხვადასვა კუთხით ან სულაც სარკულად შევაბრუნოთ. ამისათვის შედით Image -> Rotation. თქვენ შეგიძლიათ შემოაბრუნოთ გამოსახულება 180°, 90° საათის ისრის ან მის საწინააღმდეგო მიმართულებით, შეგიძლიათ შეაბრუნოთ გამოსახულება სარკულად ჰორიზონტალურ ან ვერტიკალურ სიბრტყეში, ხოლო ფუნქცია Arbitrary

დაგეხმარებათ მოაბრუნოთ გამოსახულება ნებისმიერი კუთხით - უბრალოდ მიუთითეთ კუთხის რიცხვითი მნიშვნელობა და მიმართულება (საათის ისრის ან მის საწინააღმდეგო მიმართულებით).

ფუნქცია Image -> Dublicate… გაძლევთ საშუალებას შექმნათ გამოსახულების ზუსტი ასლი, როგორც თავიდანვე, ისე მუშაობის პროცესში. კარგია გამოიყენოთ მაშინ, როდესაც არ გინდათ, რომ დაზიანდეს გამოსახულების ორიგინალი; ან მუშაობის რაღაც ეტაპზე გინდათ მოსინჯოთ რაიმე ფილტრი ან ფუნქცია, მაგრამ შედეგში არ ხართ დარწმუნებული - შექმენით ასლი და გააგრძელეთ მუშაობა.

## ფენები და მათთან მუშაობა

ფენები გამჭვირვალე ქაღალდების დასტას გავს. ზემოთ მოთავსებული ფენების გამჭვირვალე არეების წყალობით ჩვენ შეგვიძლია დავინახოთ ქვედა ფენები. ჩვენ შეგვიძლია ვცვალოთ ფენების მიმდევრობა ისევე, როგორც დასტაში ფურცლები.



სურ. 1.3-17. ფენების ვიზუალური წარმოდგენა

ფენები შეგვიძლია გამოვიყენოთ ტექსტის, ვექტორული ობიექტის ან სხვა გამოსახულების დადებისას ძირითად გამოსახულებაზე. შესაძლებელია ამ ფენებზე სპეციალური ეფექტებიც გამოვიყენოთ, როგორიცაა ჩრდილი, განათება, ფერის შეცვლა და სხვა.

ფენებთან სამუშაოდ ვიყენებთ მენიუ Layer (ფენა) და ფენების პანელს - Window -> Layers (F7).

მენიუდან ჩვენ შეგვიძლია შევქმნათ სხვადასხვა სახის ფენები, დავაჯგუფოთ ან დავაკავშიროთ ისინი ერთმანეთთან, გამოვიყენოთ ეფექტები და ა.შ. ფენების პანელის დახმარებით კი ჩვენ ვიზუალურად ვხედავთ ფენებს, შეგვიძლია მათ შევუცვალოთ მიმდევრობა, გამოვიყენოთ ეფექტები, წავშალოთ, გავთიშოთ/გამოვაჩინოთ და ა.შ. ფუნქციების ნაწილი შესაძლებელია გამოიყენოთ როგორც ერთი ფენისთვის, ისე ფენათა ჯგუფისთვის.

**შენიშვნა:** *გაითვალისწინეთ, რომ ზოგი ფუნქცია, რაც არის ხელმისაწვდომი მენიუში, არ არის ფენების (Layers) პანელზე და პირიქით.*

### *ფენების ნაირსახეობები*

პროგრამაში რამდენიმე ნაირსახეობის ფენა არსებობს. ყველა ფენას თავისი დანიშნულება აქვს.

**ფენების ჯგუფი** (Layer Group): ეს ერთგვაროვანი საქაღალდეა, რომელში შეგიძლიათ მოათავსოთ ან ამოიღოთ ფენები (იხ. ფენების დაჯგუფება/დაშლა);

**ტექსტური ფენა** (Type Layer): მახასიათებლებით გაქვს გამოსახულების ფენას, მაგრამ მასზე შესაძლებელია ტექსტის რედაქტირება (მაგ. ფერი, ზომა, შრიფტი) (იხ. ტექსტი და მისი რედაქტირება);

**მაკორექტირებელი ფენა** (Adjustment Layer): ამ ფენის დახმარებით შესაძლებელი ქვემდებარე ფენების ფერის ან/და ტონალობის შეცვლა;

**ფენის ნიღაბი** (Layer Mask): ნიღაბზე ფუნჯით (იხ. ფუნჯი (Brush)) ან საშლელით (იხ. საშლელი (Eraser Tools)) მოქმედების დროს თქვენ შეგიძლიათ გამოაჩინოთ ან პირიქით დამალოთ გამოსახულების ცალკეული ნაწილები;

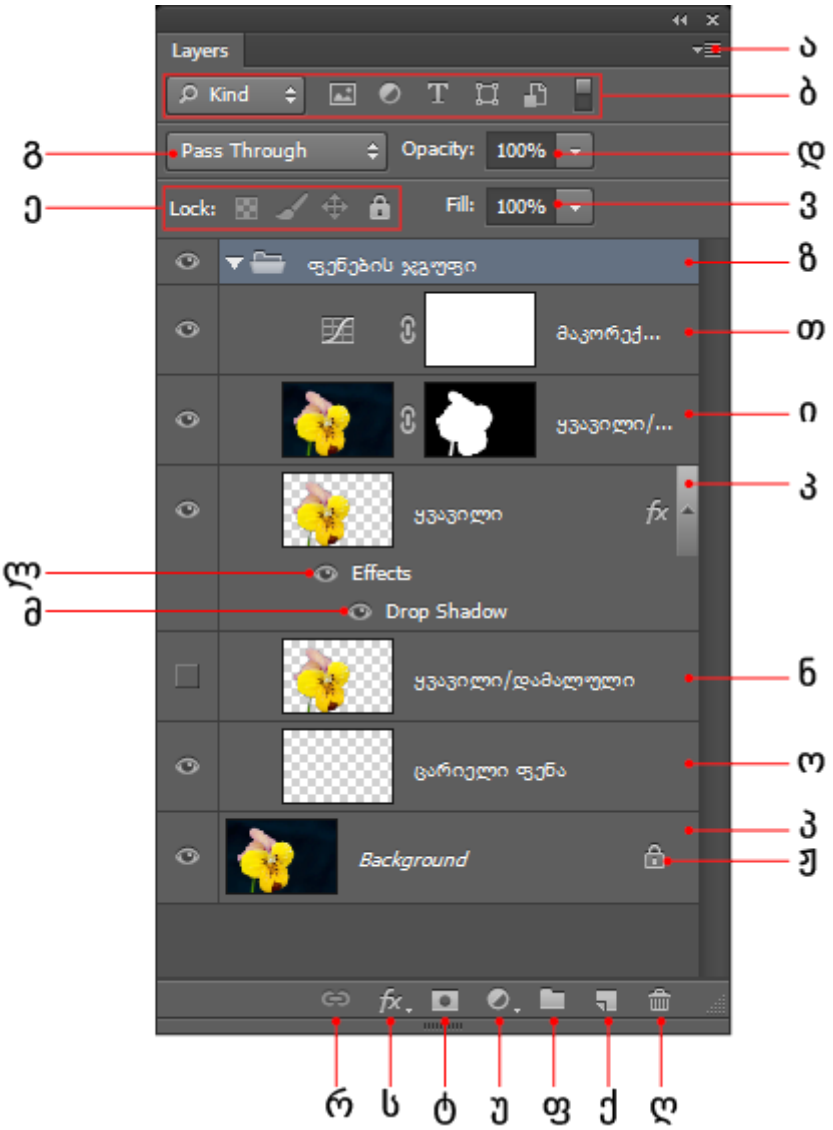
**სმარტ-ობიექტი** (Smart Object): ეს ფენა ერთგვარ კონტეინერს წარმოადგენს. კონტეინერში შეიძლება განვათავსოთ როგორც ერთი, ისე რამდენიმე ობიექტი (ისინი თავის მხრივ შეიძლება ფენებზე იყვნენ განლაგებული): ვექტორული ობიექტები, RAW ფაილები, ვიდეო, 3D და კიდევ უამრავი სხვა ტიპის ობიექტი;

**მხატვრული ფენა** (Art Layer): ძირითადი სამუშაო ფენა;

**ძირითადი ფენა** (Background): ფენა ყოველთვის ჩაკეტილი, მასზე განთავსებული გამოსახულების გადაადგილება შეუძლებელი და მის მიმართ ვერ გამოვიყენებთ ფენების სტილებს (იხ. ფენების სტილები).

**ვიდეოს ფენა** (Video Layer): ამ ფენაზე ვიდეო გამოსახულება თავსდება. შეგიძლიათ მისი ფერების კორექცია, გამოიყენოთ ფილტრები, დაჭრათ, მოახდინოთ მისი მარტივი მონტაჟი.

**3D-ს ფენა** (3D Layer): პროგრამაში შესაძლებელია როგორც 3D გამოსახულების იმპორტირება, ისე მათი შექმნა და მუშაობა. ბოლო ვერსიებში გაუმჯობესებულია 3D ბეჭდვის ტექნოლოგია.




სურ. 1.3-18. ფენების პანელი (Layers).
ა - პანელის მენიუ;
ბ - ფილტრაციის ინსტრუმენტები;
გ - ფენების შერევის რეჟიმები;
დ - ფენის გამჭვირვალობა;
ე - ფენის ჩაკეტვის ინსტრუმენტები;
ვ - შევსების გამჭვირვალობა;
ზ - ფენათა ჯგუფი;
თ - მაკორექტირებელი ფენა;
ი - შენიღბული ფენა;
კ - ეფექტების დამალვა/გამოჩენა;
ლ - ყველა ეფექტის დროებით გათიშვა;
მ - კონკრეტული ეფექტის დროებით გათიშვა;
ნ - დამალული ფენა;
ო - ცარიელი ფენა;
პ - ძირითადი ფენა;
ჟ - ფენის ჩაკეტვის ნიშნული
რ - ფენებს შორის ბმა/ბმის დაშლა;
ს - ფენაზე ეფექტების გამოყენება;
ტ - ფენაზე ნიღბის დამატება;
უ - მაკორექტირებელი ფენის შექმნა;
ფ - ახალი ჯგუფის შექმნა (ფენების);
ქ - ახალი ფენის შექმნა;
ღ - ფენის წაშლა.

### *შერევის (ზედდების) რეჟიმები*

შერევის რეჟიმები დაყოფილია მცირე ჯგუფებად. სულ ექვსი ესეთი ჯგუფია.

**საბაზო რეჟიმები:**

**Normal** - ნორმალურ რეჟიმში ზედა და ქვედა ფენის პიქსელების შერევა არ ხდება. ზედა ფენის პიქსელები ავტომატურად ჩაანაცვლებენ ქვედა ფენის პიქსელებს, თუ ზედა ფენის გამჭვირვალობა 100%-ის ტოლია. პიქსელების ფერების შერევის ეფექტი, ფენის გამჭვირვალობის შემცირებით მიიღწევა, ისიც იმ შემთხვევაში, თუ ფენებზე სხვადასხვა სახის გამოსახულება არის მოთავსებული. ავტომატურ მდგომარეობაში, ყველა ფენა Normal რეჟიმში იმყოფება.

**Dissolve** - გახსნის რეჟიმში პიქსელების ფერი შემთხვევითი შერჩევის გზით მიიღება და მიღებულ გამოსახულებას აქვს მარცვლოვანი, ფოროვანი სახე. ეფექტი შესამჩნევია თუ ზედა ფენის გამჭვირვალობა 100%-ზე ნაკლებია.

**დამუქების რეჟიმები:**

**Darken** - მუქი ფერით ჩანაცვლება. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო ღია ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელი ფერი, ხოლო პიქსელი, რომელიც უფრო მუქი ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Multiply** - გამრავლება. ხდება ზედა და ქვედა ფენების ფერების გადამრავლება და შედეგად მიღებული გამოსახულების ფერი უფრო მუქია, ვიდრე ცალკე აღებული თითოეული ფენის გამოსახულების ფერი. პროცესში მონაწილეობს ყველა ფერის პიქსელი, თეთრი ფერის გარდა.

**Color Burn** - ფუძის დამუქება. ხდება მუქი ფერის პიქსელების დამუქება, ხოლო ღია ფერის პიქსელები უცვლელი რჩება.

**Linear Burn** - ხაზოვანი დამუქება. ისევე მოქმედებს, როგორც Color Burn რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Darker Color** - შედარებით მუქი ფერი. ისევე მოქმედებს, როგორც Darken რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**გაღიავების რეჟიმები:**

**Lighten** - ღია ფერით ჩანაცვლება. Darken რეჟიმის საპირისპირო რეჟიმია. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო მუქი ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელის ფერი, ხოლო პიქსელი, რომელიც უფრო ღია ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Screen** - განათება. Multiply რეჟიმის საპირისპირო რეჟიმია. ხდება გამოსახულების ყველა ფერის პიქსელის გაღიავება, შავი ფერის გარდა.

**Color Dodge** - ფუძის განათება. Color Burn რეჟიმის საპირისპირო რეჟიმია. აღიავებს გამოსახულების ღია ფერის პიქსელებს და უცვლეს ტოვებს მუქი ფერის პიქსელებს.

**Linear Dodge** - ხაზოვანი განათება. იგივენაირად მოქმედებს, როგორც Color Dodge რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Lighter Color** - ღია ფერი. ორივე ფენის პიქსელების ფერის შედარების შემდეგ, რჩება უფრო ღია ფერის პიქსელი.

**კონტრასტის რეჟიმები:**

**Overlay** - გადაფარვა. ეს არის Multiply და Screen რჟიმების კომბინირებული რეჟიმი. ხდება მუქი პიქსელების დამუქება და ღია პიქსელების გაღიავება. ეფექტი არ მოქმედებს შავ და თეთრ ფერებზე. ამ რეჟიმში 50%-იანი ნაცრისფერი გაუმჭვირვალე ხდება.

**Soft Light** - რბილი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**Hard Light** - ძლიერი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Vivid Light** - მკვეთრი განათება. იგივენაირად მოქმედებს, როგორც Overlay და Hard Light რეჟიმები, მაგრამ ეფექტი უფრო ძლიერია.

**Linear Light** - ხაზოვანი განათება. Vivid Light რეჟიმივით მოქმედებს, მაგრამ ეფექტი უფრო ძლიერია.

**Pin Light** - წერტილოვანი განათება. ღია ფერის პიქსელების შერევის დროს იყენებს Lighten რეჟიმს, ხოლო მუქი ფერის პიქსელების შერევისას, Darken რეჟიმს.

**Hard Mix** - მკვეთრი შერევა. RGB არხების მიხედვით ხდება ზედა და ქვედა პიქსელების სიმკვეთრის მაჩვენებლების გადათვლა. დაჯამების შემდეგ პიქსელი იღებს ზღვრულ მნიშვნელობას: თუ ჯამური მაჩვენებელი 255-ზე ნაკლებია, არხის რიცხვითი მაჩვენებელი 0-ს უტოლდება, ხოლო თუ ჯამური მაჩვენებელი 255 -ია ან 255-ზე მეტია - 255-ს. ამის შედეგად, გამოსახულებაზე 8 ძირითადი ფერის პიქსელი მიიღება ანუ ხდება გამოსახულების პოსტერიზაცია.

**შედარების რეჟიმები**:

**Difference** - განსხვავება. ხდება პიქსელების სიმკვეთრის მაჩვენებლების შედარება, დიდი მაჩვენებლებიდან პატერა მაჩვენებლების გამოკლება და სხვაობის შებრუნება. ერთნაირი მაჩვენებლების პიქსელები შავი ფერით აისახება.

**Exclusion** - გამორიცხვა. ისევე მოქმედებს, როგორც Difference რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**გამოსახულების კომპონენტების რეჟიმები:**

**Hue** - ტონი. მიიღება გამოსახულება ზედა ფენის ტონით და ქვედა ფენის გაჯერებულობით და სიმკვეთრით.

**Saturation** - გაჯერებულობა. მიიღება გამოსახულება ზედა ფენის გაჯერებულობით და ქვედა ფენის ტონით და სიმკვეთრით.

**Color** - ფერი. მიიღება გამოსახულება ზედა ფენის ტონით და გაჯერებულობით და ქვედა ფენის სიმკვეთრით.

**Luminosity** - სიმკვეთრე. მიიღება გამოსახულება ზედა ფენის სიმკვეთრით და ქვედა ფენის ტონით და გაჯერებულო- ბით.

*ახალი ფენის შექმნა*

Layer -> New -> New Layer (Ctrl + Shift + N) - ახალი ფენი შექმნა;

Layer -> New -> New Layer from Background - ახალი ფენის შექმნა ძირითადი ფენისგან;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> New Layer (Ctrl + Shift + N);

მართვის პანელზე ახალი ფენის შექმნის ღილაკი (სურ. 1.3-18, ქ).

*ფენის ასლის/დუბლიკატის შექმნა*

Layer -> New -> Layer via Copy (Ctrl + J) - შექმნის მიმდინარე ფენის ასლს. თუ გამოსახულებაზე მონიშნული გაქვთ არე, მაშინ მოხდება მისი ასლის შექმნა და მისი განთავსება ახალ ფენაზე;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Dublicate Layer…;

ფენების პანელზე ფენის გადათრევით ახალი ფენის შექმნის ღილაკზე (სურ. 1.3-18, ქ).

### *ფენის დაბლოკვა/გააქტიურება*

ფენების პანელზე ჩამკეტი ღილაკების (სურ. 1.3-18, ე) დახმარებით. ჩაკეტვის ოთხნაირი საშუალება არსებობს - მხოლოდ გამჭვირვალე პიქსელები, გამოსახულების პიქსელები, პოზიციის დაბლოკვა, ყველაფრის დაბლოკვა. მაგალითად ძირითადი ფენა (სურ. 1.3-18, პ) ნაგულისხმევად სრულად არის ჩაკეტილი და გვერდით შესაბამისი ნიშნული აქვს (სურ. 1.3-18, ჟ).

დასაბლოკად, მონიშნეთ სასურველი ფენა და დააწექით შესაბამის ღილაკს (სურ. 1.3-18, ე).

ბლოკის მოსახსნელად ხელმეორედ დააწექით იმავე ღილაკს.

Layer -> Lock All Layers in Group - ჯგუფში (სურ. 1.3-18, ზ) არსებულ ყველა ფენის დაბლოკვა

### *ფენის დამალვა/გამოჩენა*

ფენა ნაგულისხმევად (არსებული თუ ახალი შექმნილი) ყოველთვის ჩანს. ამას მიუთითებს ყოველი ფენის გვერდით (მარცხნიდან) არსებული თვალის ნიშნული. ფენის დასამალად დააწექით ამ ნიშნულს - ის გაქრება, რაც მიუთითებს, რომ ფენა დამალულია ანუ მასზე არსებული გამოსახულება არ გამოჩნდება სამუშაო ტილოზე (სურ. 1.3-18, ნ). გამოსაჩენად დააწექით ცარიელ კვადრატს.

ასევე შეგიძლიათ ისარგებლოთ ფუნქციით Layer -> Hide/Show Layers.

### *ფენის მონიშვნა (არჩევა)*

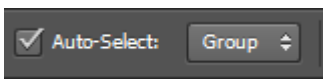
მუშაობის პროცესში გარკვეული მოქმედებები შესაძლოა არა ერთ ფენაზე, არამედ რამდენიმე ფენაზე მოგიწიოთ (დამალვა, ეფექტების გამოჩენა, ფენაზე განთავსებული ობიექტების ერთმანეთთან გასწორება და სხვა).

ფენა რომ მონიშნოთ, დააწექით შესაბამის ფენას. ფენათა ჯგუფის მოსანიშნად გამოიყენება კლავიშები Shift და Ctrl.

Shift კლავიშის გამოყენება: როდესაც ფენები გინდათ მონიშნოთ მიმდევრობით, მონიშნეთ ერთი ფენა და შემდეგ კლავიშის დახმარებით მონიშნეთ სხვა ფენა. მოინიშნება ეს ორივე ფენა და მათ შორის მდებარე ყველა ფენა.

Ctrl კლავიშის გამოყენება: მონიშნეთ ერთი ფენა და კლავიშის გამოყენებით (უნდა გეჭიროთ) შეგიძლიათ მონიშნოთ ფენები არჩევითად.

ფენების მონიშვნა შესაძლებელია ასევე გადაადგილების (Move) ინსტრუმენტის დახმარებით. ფორმატირების ზოლზე უნდა ჩართოთ ავტომატური მონიშვნა (Auto-Select), ხოლო მენიუდან აირჩიოთ რისი მონიშვნა გინდათ: ჯგუფის (სურ. 1.3-18, ზ) თუ ცალკეული ფენის.




სურ. 1.3-19. გადაადგილების (Move) ინსტრუმენტის ფუნქცია ავტომატური მონიშვნა (Auto-Select) ფორმატირების ზოლზე

შემდეგში, მუშაობის პროცესში, გამოსახულებაზე ცალკეულ ელემენტზე დაჭერისას მოინიშნება ის ფენა, რომელზეც ეს ელემენტი მდებარეობს.

იმ შემთხვევაში, თუ ფენას გააჩნია ბმა (იხ. ფენებს შორის ბმა) სხვა ფენებთან, მაშინ გამოიძახეთ ფუნქცია Layer -> Select Linked Layers. ამ დროს მოინიშნება ყველა ფენა, რომელსაც ამ ფენასთან ბმა გააჩნია.

### *ფენების დაჯგუფება/დაშლა*

Layers -> Group Layers (Ctrl + G) - ფენების დაჯგუფება;

Layers -> Ungroup Layers (Shift + Ctrl + G) - ფენების დაჯგუფება;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> New Group - შეიქმნება ცარიელი ჯგუფი, სადაც შემდეგში გადაიტანთ სასურველ ფენებს;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> New Group from Layers - ჯგუფის შექმნა მონიშნული ფენებისგან;

ფენების პანელზე ახალი ჯგუფის შექმნის ღილაკი - ქმნის ახალ ცარიელ ჯგუფს (სურ. 1.3-18, ფ).

### *ფენებს შორის ბმა*

გამოიყენება იმ დროს, როდესაც გსურთ რაიმე საერთო მოქმედების შესრულება (გადაადგილება, ტრანსფორმაცია და სხვა), მაგრამ არ არის საჭირო მათი დაჯგუფება. მონიშნული უნდა იყოს ორი ან მეტი ფენა. ბმის შექმნის შემდეგ ფენაზე (დასახელების გვერდით) ჩნდება ჯაჭვის ნიშანი.

Layer - > Link/Unlink Layers - ფენებს შორის ბმა/ბმის წყვეტა;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) - > Link/Unlink Layers;

ფენების პანელზე ბმა/ბმის წყვეტის ღილაკი (სურ. 1.3-18, რ).

### *ფენების სწორება ერთმანეთის მიმართ*

ხანდახან საჭიროა და აუცილებელიც კი ხდება ფენებზე განთავსებული ობიექტები ერთმანეთს გაუსწოროთ. სწორება შესაძლებელია მოხდეს კიდეზე, ცენტრის მიმართ ან განისაზღვროს თანაბარი მანძილი ობიექტებს შორის. ამისათვის უნდა მონიშნოთ ის ფენები, რომლებიც უნდა გაუსწოროთ ერთმანეთს, შემდეგ გამოიძახეთ Layer -> Align. ამ მენიუში მოთავსებული ფუნქციების დახმარებით თქვენ შეგიძლიათ ობიექტები ექვსი მახასიათებლის მიხედვით გაასწოროთ.

იმისათვის, რომ მოახდინოთ ობიექტების თანაბრად განაწილება, უნდა მონიშნოთ ფენები (მინიმუმ სამი) და გამოიძახოთ Layer -> Distribute. აქაც ექვსი მახასიათებლის მიხედვით შეგიძლიათ გაანაწილოთ ობიექტები.

თუ არჩეული გაქვთ გადაადგილების (Move) ინსტრუმენტი, მაშინ შეგიძლიათ ისარგებლოთ ფორმატირების ზოლზე განლაგებული სწორებისა და განაწილების შესაბამისი ღილაკებით.

სურ. 1.3-20. სწორებისა და განაწილები ღილაკები ფორმატირების ზოლზე.

### *ფენების გადაადგილება*

ფენების გადაადგილება გადათრევით - მონიშნეთ ფენა, მოკიდეთ კურსორით და გადაათრიეთ იმ ფენებს შორის, სადაც გინდათ, რომ იყოს ეს ფენა განთავსებული;

ფენების გადასაადგილებლად ზემოთ ან ქვემოთ:

Layers -> Arrange -> Bring to Front (Shift + Ctrl + ))

Layers -> Arrange -> Bring Forward (Ctrl + ))

Layers -> Arrange -> Bring Backward (Ctrl + ()

Layers -> Arrange -> Bring to Back (Shift + Ctrl + ()

**შენიშვნა:** *ძირითადი ფენა (Background) ყველაზე ქვედა ფენაა. მასზე ქვემოთ ფენის გადაადგილება შეუძლებელი მანამ, სანამ ის არ გადაიქცევა სამუშაო ფენად (იხ.*

*შერევის (ზედდების)* რეჟიმები

შერევის რეჟიმები დაყოფილია მცირე ჯგუფებად. სულ ექვსი ესეთი ჯგუფია.

**საბაზო რეჟიმები:**

**Normal** - ნორმალურ რეჟიმში ზედა და ქვედა ფენის პიქსელების შერევა არ ხდება. ზედა ფენის პიქსელები ავტომატურად ჩაანაცვლებენ ქვედა ფენის პიქსელებს, თუ ზედა ფენის გამჭვირვალობა 100%-ის ტოლია. პიქსელების ფერების შერევის ეფექტი, ფენის გამჭვირვალობის შემცირებით მიიღწევა, ისიც იმ შემთხვევაში, თუ ფენებზე სხვადასხვა სახის გამოსახულება არის მოთავსებული. ავტომატურ მდგომარეობაში, ყველა ფენა Normal რეჟიმში იმყოფება.

**Dissolve** - გახსნის რეჟიმში პიქსელების ფერი შემთხვევითი შერჩევის გზით მიიღება და მიღებულ გამოსახულებას აქვს მარცვლოვანი, ფოროვანი სახე. ეფექტი შესამჩნევია თუ ზედა ფენის გამჭვირვალობა 100%-ზე ნაკლებია.

**დამუქების რეჟიმები:**

**Darken** - მუქი ფერით ჩანაცვლება. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო ღია ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელი ფერი, ხოლო პიქსელი, რომელიც უფრო მუქი ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Multiply** - გამრავლება. ხდება ზედა და ქვედა ფენების ფერების გადამრავლება და შედეგად მიღებული გამოსახულების ფერი უფრო მუქია, ვიდრე ცალკე აღებული თითოეული ფენის გამოსახულების ფერი. პროცესში მონაწილეობს ყველა ფერის პიქსელი, თეთრი ფერის გარდა.

**Color Burn** - ფუძის დამუქება. ხდება მუქი ფერის პიქსელების დამუქება, ხოლო ღია ფერის პიქსელები უცვლელი რჩება.

**Linear Burn** - ხაზოვანი დამუქება. ისევე მოქმედებს, როგორც Color Burn რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Darker Color** - შედარებით მუქი ფერი. ისევე მოქმედებს, როგორც Darken რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**გაღიავების რეჟიმები:**

**Lighten** - ღია ფერით ჩანაცვლება. Darken რეჟიმის საპირისპირო რეჟიმია. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო მუქი ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელის ფერი, ხოლო პიქსელი, რომელიც უფრო ღია ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Screen** - განათება. Multiply რეჟიმის საპირისპირო რეჟიმია. ხდება გამოსახულების ყველა ფერის პიქსელის გაღიავება, შავი ფერის გარდა.

**Color Dodge** - ფუძის განათება. Color Burn რეჟიმის საპირისპირო რეჟიმია. აღიავებს გამოსახულების ღია ფერის პიქსელებს და უცვლეს ტოვებს მუქი ფერის პიქსელებს.

**Linear Dodge** - ხაზოვანი განათება. იგივენაირად მოქმედებს, როგორც Color Dodge რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Lighter Color** - ღია ფერი. ორივე ფენის პიქსელების ფერის შედარების შემდეგ, რჩება უფრო ღია ფერის პიქსელი.

**კონტრასტის რეჟიმები:**

**Overlay** - გადაფარვა. ეს არის Multiply და Screen რჟიმების კომბინირებული რეჟიმი. ხდება მუქი პიქსელების დამუქება და ღია პიქსელების გაღიავება. ეფექტი არ მოქმედებს შავ და თეთრ ფერებზე. ამ რეჟიმში 50%-იანი ნაცრისფერი გაუმჭვირვალე ხდება.

**Soft Light** - რბილი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**Hard Light** - ძლიერი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**Vivid Light** - მკვეთრი განათება. იგივენაირად მოქმედებს, როგორც Overlay და Hard Light რეჟიმები, მაგრამ ეფექტი უფრო ძლიერია.

**Linear Light** - ხაზოვანი განათება. Vivid Light რეჟიმივით მოქმედებს, მაგრამ ეფექტი უფრო ძლიერია.

**Pin Light** - წერტილოვანი განათება. ღია ფერის პიქსელების შერევის დროს იყენებს Lighten რეჟიმს, ხოლო მუქი ფერის პიქსელების შერევისას, Darken რეჟიმს.

**Hard Mix** - მკვეთრი შერევა. RGB არხების მიხედვით ხდება ზედა და ქვედა პიქსელების სიმკვეთრის მაჩვენებლების გადათვლა. დაჯამების შემდეგ პიქსელი იღებს ზღვრულ მნიშვნელობას: თუ ჯამური მაჩვენებელი 255-ზე ნაკლებია, არხის რიცხვითი მაჩვენებელი 0-ს უტოლდება, ხოლო თუ ჯამური მაჩვენებელი 255 -ია ან 255-ზე მეტია - 255-ს. ამის შედეგად, გამოსახულებაზე 8 ძირითადი ფერის პიქსელი მიიღება ანუ ხდება გამოსახულების პოსტერიზაცია.

**შედარების რეჟიმები**:

**Difference** - განსხვავება. ხდება პიქსელების სიმკვეთრის მაჩვენებლების შედარება, დიდი მაჩვენებლებიდან პატერა მაჩვენებლების გამოკლება და სხვაობის შებრუნება. ერთნაირი მაჩვენებლების პიქსელები შავი ფერით აისახება.

**Exclusion** - გამორიცხვა. ისევე მოქმედებს, როგორც Difference რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**გამოსახულების კომპონენტების რეჟიმები:**

**Hue** - ტონი. მიიღება გამოსახულება ზედა ფენის ტონით და ქვედა ფენის გაჯერებულობით და სიმკვეთრით.

**Saturation** - გაჯერებულობა. მიიღება გამოსახულება ზედა ფენის გაჯერებულობით და ქვედა ფენის ტონით და სიმკვეთრით.

**Color** - ფერი. მიიღება გამოსახულება ზედა ფენის ტონით და გაჯერებულობით და ქვედა ფენის სიმკვეთრით.

**Luminosity** - სიმკვეთრე. მიიღება გამოსახულება ზედა ფენის სიმკვეთრით და ქვედა ფენის ტონით და გაჯერებულო- ბით.

ახალი ფენის შექმნა *ძირითადი ფენისგან).*

ფენების მიმდევრობა რომ შეაბრუნოთ, აირჩიეთ Layers -> Arrange -> Reverse.

### *ფენების წაშლა*

შეგიძლიათ წაშალოთ როგორც ფენა, ისე ფენათა ჯგუფი.

Layer -> Delete -> Layer - ფენ(ებ)ის წაშლა;

Layer -> Delete -> Hidden Layers - ყველა დამალული ფენის (იხ. ფენის დამალვა/გამოჩენა) წაშლა;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Delete Layer;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Delete Hidden Layers;

ფენების პანელზე ფენის წაშლის ღილაკი (სურ. 1.3-18, ღ)

### *ფენების შერწყმა/გაერთიანება*

Layer -> Merge Down (Ctrl + E) - ქვედა ფენასთან შერწყმა. აქტიურია ერთი მონიშნული ფენის დროს;

Layer -> Merge Layers (Ctrl + E) - ფენების შერწყმა. აქტიურია ერთზე მეტი მონიშნული ფენის დროს. მონიშვნის მიმდევრობას მნიშვნელობა არ აქვს;

Layer -> Merge Visible (Shift + Ctrl + E) - ხილული ფენების შერწყმა. დამალული ფენები ხელუხლებელი დარჩება;

Layer -> Flatten Image - ყველა ფენის შერწყმა. ამის შემდეგ დარჩება მხოლოდ ერთი ფენა, რომელიც იქნება ძირითადი (Background);

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Merge Down/Layers;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Merge Visible;

ფენების პანელის მენიუ (სურ. 1.3-18, ა) -> Flatten Image;

**შენიშვნა:** *დამატებითი კლავიშების კომბინაცია Shift + Ctrl + Alt + E აერთიანებს ყველა ფენას, მაგრამ ამ დროს ყველა ფენა რჩება როგორც იყო, ხოლო წარმოქმნილი ფენა თავსდება ყველაზე ზემოთ.*

### *სმარტ-ობიექტი*

სმარტ-ობიექტის შექმნა ორი გზით შეიძლება (წინასწარ ფენა უნდა მონიშნოთ):

Layer -> Smart Objects -> Convert to Smart Object;

ფენების პანელის მენიუ -> Convert to Smart Object;

სხვა ფაილის შემოტანისას მიმდინარე ფაილში - File -> Place.

ამ ფენაზე შესაძლებელია ჩვეულებრივი მოქმედებების შესრულება, როგორიცაა მისი ტრანსფორმაცია, ფენების სტილის გამოყენება ან ფენების ჯგუფში მოთავსება.

სმარტ-ობიექტის პირდაპირი რედაქტირება შეუძლებელია (მოჭრა, წაშლა და სხვა). ამისათვის უნდა გამოიძახოთ:

Layer -> Smart Objects -> Edit Content;

ფენების პანელის მენიუ -> Edit Content;

გამოსახულება გაიხსნება სხვა ფაილში, სადაც თქვენ მოახდენთ მის რედაქტირებას (წაშლა, ხატვა და ა.შ.), შეინახავთ, როგორც ჩვეულებრივ ფაილს და სამუშაო ფაილში დაბრუნებისთანავე, თქვენ დაგხვდებათ გამოსახულება რედაქტირებული.

სმარტ-ობიექტზე ფილტრის გამოყენებისას, ის იმუშავებს უკვე როგორც სმარტ-ფილტრი. ჩვეულებრივისგან ეს ფილტრი იმით განსხვავდება, რომ მისი მოქმედება შესაძლებელია შეცვლილი ან გაუქმებული იყოს მუშაობის ნებისმიერ ეტაპზე.

ჩვეულებრივ ფენაზე სმარტ-ფილტრი არ მუშაობს - ის ჯერ გარდაიქმნება სმარტ-ობიექტად და ამის შემდეგ მასზე შესაძლებელია სმარტ ფილტრით მოქმედება.

## ფენების სტილები

იმისათვის, რომ ფენას რაიმე მხატვრული ან თუნდაც რეალისტური ეფექტი შევძინოთ, ამისათვის შეგვიძლია სხვადასხვა სახის ეფექტები გამოვიყენოთ. ეფექტები შესაძლებელი გამოვიყენოთ როგორც ერთ ფენაზე, ისე ფენათა ჯგუფზე. ასევე ერთდროულად შესაძლებელია გამოვიყენოთ რამდენიმე ეფექტი ერთდროულად.

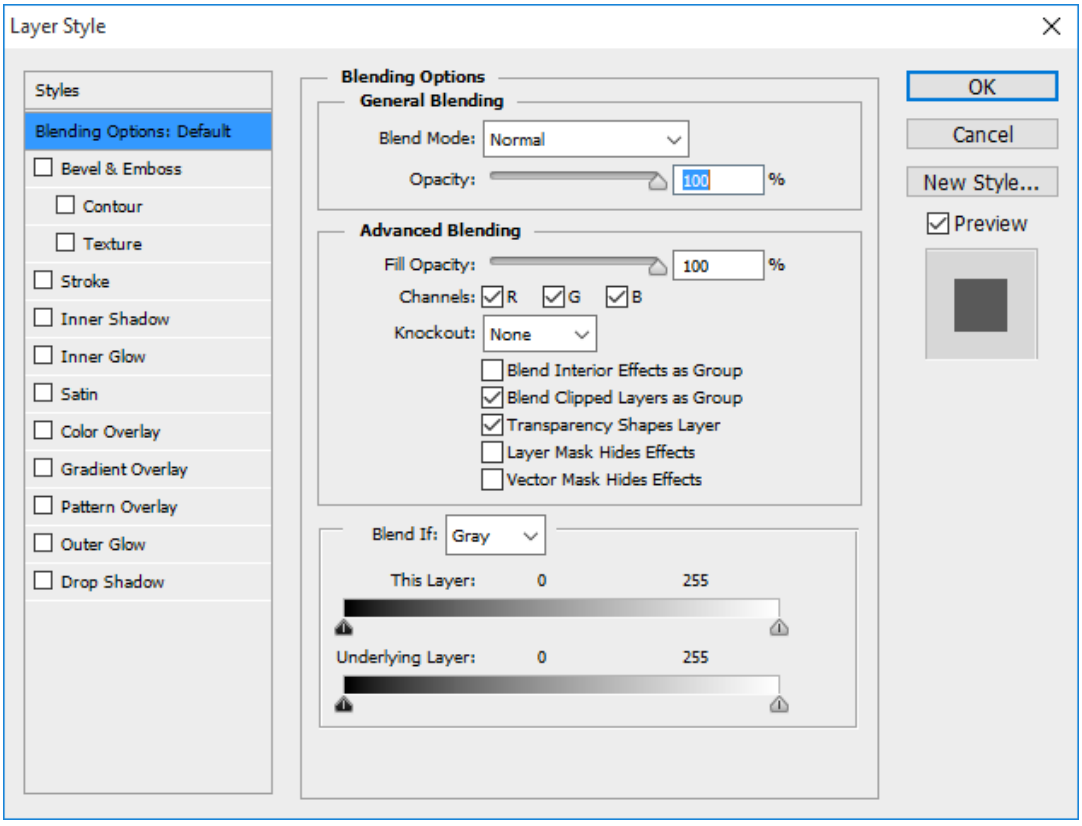
სტილი წარმოადგენს სხვადასხვა ეფექტის ერთობლიობას (ჩრდილი, ფაქტურა და ა.შ.)

სტილის მოქმედება ვრცელდება ფენაზე განთავსებულ ობიექტებზე მთლიანად. ამიტომ თუ ამავე ფენაზე დაემატება ახალი ელემენტი, მასზეც გავრცელდება ის სტილი, რაც ფენაზეა გამოყენებული.

თქვენ შეგიძლიათ გამოიყენოთ გამზადებული სტილები (Window -> Styles) ან შექმნათ საკუთარი. საკუთარი სტილები რომ შექმნათ, წინასწარ უნდა მონიშნოთ სასურველი ფენა და გამოიძახოთ:

Layer -> Layer Styles. შემდეგ მენიუდან აირჩიოთ თქვენთვის სასურველი სტილი.

ფენების მართვის პანელზე დააწვეთ ეფექტების ღილაკს (სურ. 1.3-18, ს) და გამოსული მენიუდან აირჩიეთ სასურველი სტილი.

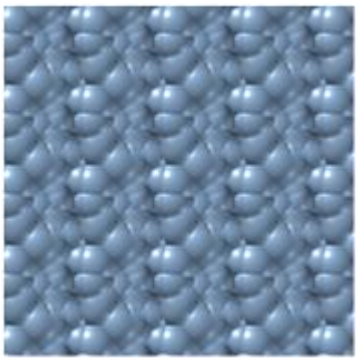



სურ. 1.3-21. ფენის სტილის (Layer Style) დიალოგური ფანჯარა. მისი დახმარებით თქვენ შეგიძლიათ ფენის გამჭვირვალობის, შერევის მეთოდებისა და ეფექტების მართვა/გამოყენება

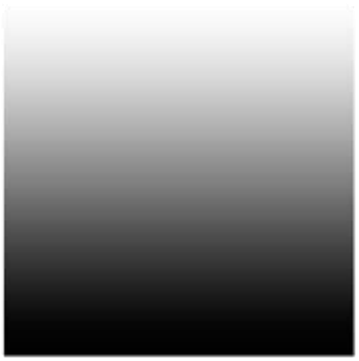
**შენიშვნა:** *წინა ვერსიებისგან განსხვავებით, ამ ვერსიაში (და მომდევნოებში) ეფექტების დალაგების მიმდევრობა შეცვლილია. პირველად გამოსახულებაზე გამოიყენება* ***დაცემული ჩრდილი*** *(Drop Shadow), ამიტომ ის ქვემოთ არის განთავსებული, ხოლო ეფექტი* ***დაცერება & რელიეფი*** *(Bevel & Emboss) გამოიყენება ბოლოს, ამიტომ ის ზემოთ არის განთავსებული.*

სურ. 1.3-22. დაცემული ჩრდილი (Drop Shadow)

სურ. 1.3-23. გარე მნათობი კონტური (Outer Glow)



სურ. 1.3-24. ფაქტურის გადაკვრა (Pattern Overlay)



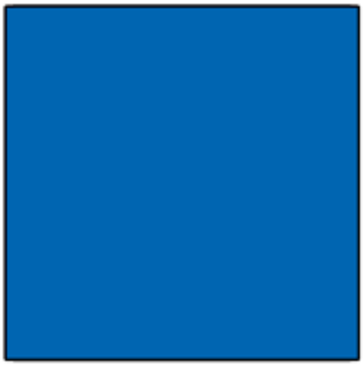
სურ. 1.3-25. გრადიენტის გადაკვრა (Gradient Overlay)

სურ. 1.3-26. ფერის გადაკვრა (Color Overlay)

სურ. 1.3-27. ატლასის გადაკვრა

სურ. 1.3-28. შიდა მნათობი კონტური (Inner Glow)

სურ. 1.3-29. შიდა ჩრდილი (Inner Shadow)



სურ. 1.3-30. კანტი (Stroke)

სურ. 1.3-31. დაცერება და რელიეფი (Bevel & Emboss)

## 1.2-სა და 1.3-ის შემაჯამებელი სავარჯიშოები

1. მოიძიეთ სხვადასხვა სურათი, რომელზეც არეების მოსანიშნად გამოიყენებთ ინსტრუმენტებს: **ლასო**, **სწორხაზოვანი** ან **მაგნიტური ლასო**, **სწრაფი მონიშვნა**, მონიშვნა **გეომეტრიული ფორმების** მიხედვით, **ფერის დიაპაზონის** მიხედვით; ინსტრუმენტი სწორად შეარჩიეთ ამოცანის მიხედვით; მოახდინეთ მონიშნული არის **საზღვრების სრულყოფა** (Refine Edge).
2. ჩამოთვალეთ შერევის რეჟიმები;
3. დაფერეთ მონიშნული არე ახალი ფენის გამოყენებით; გამოიყენეთ **შერევის რეჟიმი** ფერის დასადებად გამოსახულებაზე.
4. შეარჩიეთ გამოსახულება, რომლის რედაქტირებასაც მოახდენთ შევსებით **შიგთავსის გათვალისწინებით** (Content Aware).
5. შექმენით ახალი **ფუნჯები**. შეინახეთ ისინი თქვენი საკუთარი ფუნჯების ნაკრების სახით.
6. მოიძიეთ კონტურული გამოსახულებები. მოახდინეთ მათი დაფერვა **ფუნჯის** დახმარებით და შესაბამისი **შერევის რეჟიმების** გამოყენებით; საჭიროების შემთხვევაში შექმენით ახალი ფუნჯი.
7. შეარჩიეთ რამდენიმე გამოსახულება, რომელზეც ფონს წაშლით **ჯადოსნური საშლელის** ან **ფონის საშლელის** დახმარებით.
8. ჩამოთვალეთ **ფენების სტილები**. შექმენით და შეინახეთ ახალი სტილი.
9. ჩამოთვალეთ ტრანსფორმაციის მეთოდები.
10. მოახდინეთ სხვადასხვა გამოსახულების კომბინირება. გამოიყენეთ ტრანსფორმაციის სხვადასხვა ინსტრუმენტები ფრაგმენტების მოსარგებად; გამოსახულების კომბინირებისას გამოიყენეთ **შერევის რეჟიმები;** საჭიროების შემთხვევაში გამოიყენეთ ფენების ეფექტები.
11. გამოსახულებას შეუცვალეთ ფორმა **დეფორმაციისა** და **მარიონეტული ტრანსფორმაციის** დახმარებით;
12. რა არის სმარტ-ობიექტი. მოახდინეთ გამოსახულებების კომბინირება ისე, რომ გამოიყენოთ მხოლოდ სმარტ-ობიეტები.
13. მოახდინეთ გამოსახულების ზომების შეცვლა შიგთავსის გათვალისწინებით (Conten Aware Scale).
14. შერჩეულ გამოსახულებებს შეუცვალეთ ზომები; შეცვალეთ **ტილოს** ზომა; **შემოჭრის** ინსტრუმენტის დახმარებით მოახდინეთ გამოსახულების შემოჭრა. ჩამოთვალეთ შემოჭრის მეთოდები.
15. შეარჩიეთ სურათი (გარჩევადობა 300 dpi) და მოახდინეთ მისი შემოჭრა ფბ-ს ან ტვიტერის მთავარი სურათის მახასიათებლების მიხედვით. შეინახეთ შესაბამის ფორმატში.

## 1.4. ტექსტი და მისი რედაქტირება

პარაგრაფის შესაბამისი თემატიკა

- ტექსტისთვის შრიფტის არჩევა, ზომების დაყენება, სიმბოლოებს შორის ინტერვალის რეგულირება, სტრიქონებს შორის მანძილის დაყენება, აბზაცების სწორების მართვა
- ტექსტისთვის სტილების შექმნა: აბზაცის სტილი (Paragraph Style) და სიმბოლოს სტილი (Character Style)
- ტექსტის გარდაქმნა ვექტორულ ან/და რასტრულ ობიექტად
- ტექსტის ტრანსფორმირების ინსტრუმენტები
- ტექსტური ფენისთვის ფენების სტილების (Layer Style) გამოყენება.

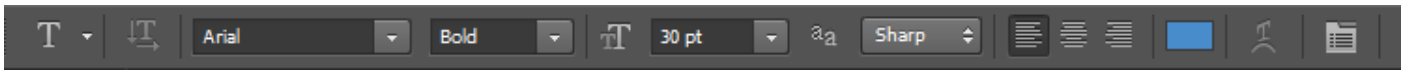
ტექსტთან სამუშაოდ გამოიყენება **ტექსტის ინსტრუმენტი** (სურ. 1.1-9, ვ).

ორი მიმართულებით შეგვიძლია ტექსტის ბეჭდვა - ჰორიზონტალურად და ვერტიკალურად.

ასევე შეგვიძია 2 ნაირსახეობის ტექსტის შექმნა - ჩვეულებრივი ვექტორული სახის ტექსტი და მონიშნული არის სახით.

პირველის შემთხვევაში ჩვენ გვაქვს ტექსტის ვიზუალური მხარე, ხოლო მეორე შემთხვევაში ჩვენ გვაქვს მხოლოდ მონიშნული არე და ამ შემთხვევაში შესაბამისი მოქმედებების შესრულება შეგვიძლია (იხ. მონიშვნის ხელსაწყოები და მონიშვნა).

მაგრამ, როგორი სახითაც ჩვენ არ უნდა ვქმნიდეთ ტექსტს - ჩვეულებრივი თუ მონიშნული არის - ორივე შემთხვევაში ჩვენ შეგვიძლია შევასრულოთ მსგავსი მოქმედებები: შრიფტის არჩევა, მისი ზომა, სწორება და სხვა. ტექსტის ინსტრუმენტის არჩევის შემდეგ, ფორმატირების ზოლზე ჩვენ შესაბამის ფუნქციებს ვნახავთ, რომელიც დაგვეხმარება ტექსტის შექმნაში და შემდეგ ფორმატირებაში.




სურ. 1.4-1. ტექსტის ფორმატირების ზოლი.

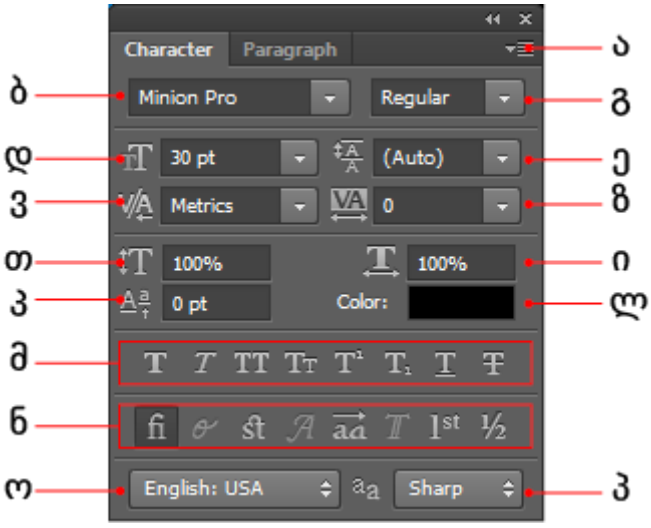
ფორმატირების ზოლიდან ჩვენ შეგვიძლია ავირჩიოთ შრიფტი, მისი ზომა, სწორება, ფერი და სხვა.

**შენიშვნა:** *ტექსტის ფერი ნაგულისხმევად არის წინა ფონის ფერი.*

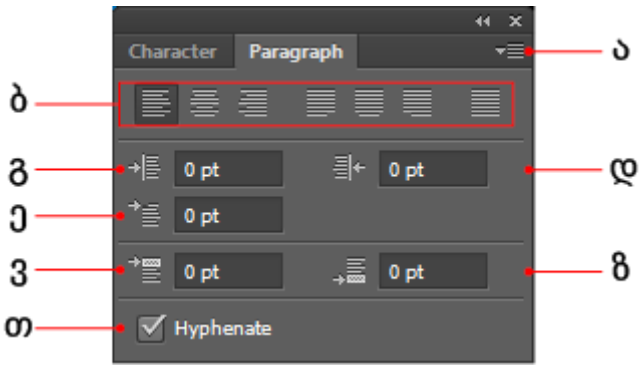
გარდა ფორმატირების ზოლისა, რომელზეც მხოლოდ ძირითადი ფუნქციებია გამოტანილი, არის კიდევ ორი პანელი, რომელიც ტექსტის ფორმატირებისთვის გამოიყენება:

Window -> Character - სიმბოლოს ფორმატირების პანელი;

Window -> Paragraph - აბზაცის ფორმატირების პანელი.




სურ. 1.4-2. ა - სიმბოლოს ფორმატირების პანელის მენიუ; ბ - შრიფტის არჩევა; გ - შრიფტის სტილის დაყენება (მუქი, დახრილი და ა.შ.); დ - შრიფტის ზომის დაყენება; ე - სტრიქონებს შორის მანძილი; ვ - კერნინგი; ზ - ინტერვალი; თ - სიმბოლოს მასშტაბირება (ვერტიკალურად); ი - თ - სიმბოლოს მასშტაბირება (ჰორიზონტალურად); კ - საბაზისო ხაზის მართვა; ლ - ტექსტის ფერი; მ - ცალკეული სიმბოლოების ფორმატირების ინსტრუმენტები; ნ - Open Type ტიპის შრიფტების ფორმატირების ინსტრუმენტები; ო - ენის არჩევა; პ - შრიფტის მოხაზულობის გასწორების მეთოდები




სურ. 1.4-3. ა - პარაგრაფის პანელის მენიუ; ბ - აბზაცის სწორების მეთოდები (მარჯვნივ, მარცხნივ, ცენტრზე და ა.შ.; გ - დაცილება მარცხენა კიდიდან; დ - დაცილება მარჯვენა კიდიდან; ე - პირველი სტრიქონის დაცილება მარცხნიდან; ვ - დაცილება აბზაცამდე; ზ - დაცილება აბზაცის შემდეგ; თ - გადატანები ტექსტისთვის.

ზემოთ ჩამოთვლილი პანელების დახმარებით თქვენ შეგიძლიათ მაქსიმალურად კარგად მოახდინოთ ნებისმიერი ზომის ტექსტის ფორმატირება.

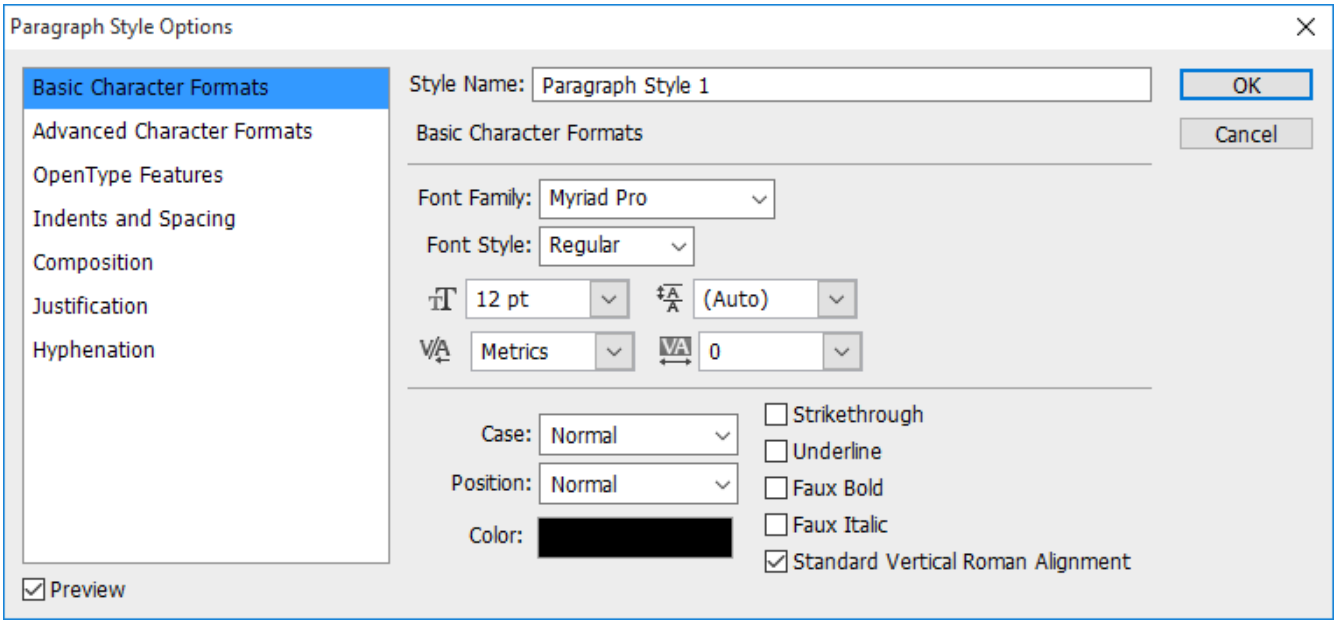
მუშაობის პროცესში არის ისეთი მომენტები, როდესაც თქვენ ხშირად გჭირდებათ ერთი და იმავე ფორმატირების გამოყენება ტექსტისთვის, მაგალითად ბანერზე ტექსტური წარწერები ან სათაურები, ცალკეული სიმბოლოს ფორმატი და სხვა.

ამისათვის თქვენ შეგიძლიათ ისარგებლოთ აბზაცისა და სიმბოლოს სტილებით. სტილები, ისევე როგორც ინდიზაინში, გაძლევენ საშუალებას სწრაფად მოახდინოთ ტექსტის ფორმატირება. მასში უკვე ჩადებულია ფორმატირების ის მახასიათებლები, რომლებიც თქვენ გამოიყენეთ აბზაცისთვის თუ ცალკეული სიმბოლოსთვის.

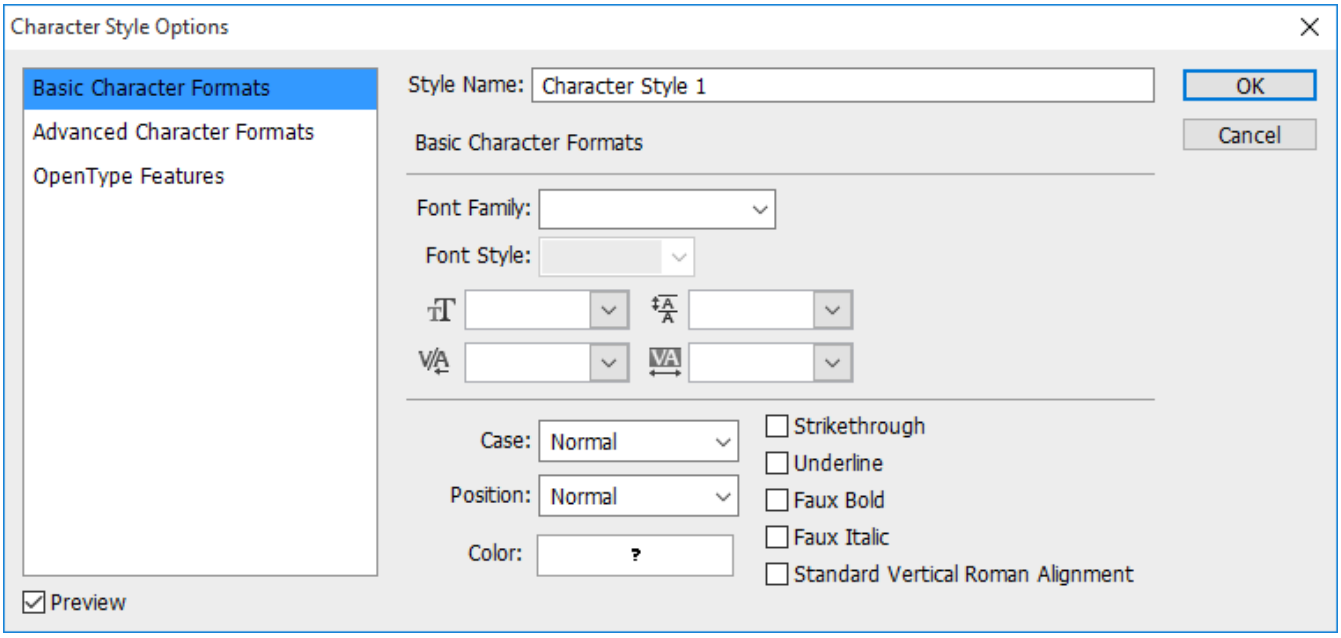
Window -> Paragraph Style - აბზაცის სტილების პანელი;

Window -> Character Style - სიმბოლოების სტილების პანელი.

უნდა გაითვალისწინოთ, რომ აბზაცის სტილი მოქმედებს მთლიანად აბზაცზე, ხოლო სიმბოლოს სტილი ვრცელდება მხოლოდ ერთ კონკრეტულ სიმბოლოზე (ის წინასწარ უნდა მონიშნოთ).




სურ. 1.4-4. ახალი აბზაცის სტილის შექმნის ფანჯარა. აქ შეგიძლიათ სტილს მიანიჭოთ დასახელება, დააყენოთ შრიფტი, ზომა, ფერი და ყველა ის მახასიათებელი რაც სიმბოლოსა და აბზაცის ფორმატირების პანელებზე იყო მოცემული. შემდეგ საკმარისია მონიშნოთ ტექსტი აირჩიოთ სტილი და ის ავტომატურად მიიღებს შესაბამის სახეს.




სურ. 1.4-5. სიმბოლოს სტილის შექმნა. აქ თქვენ გამოიყენებთ მხოლოდ სიმბოლოს ფორმატირების პანელის მახასიათებლებს. შეგიძლიათ სტილს დაარქვათ სახელი, აირჩიოთ შრიფტი, დააყენოთ ზომა, ფერი და ა,შ. შემდეგში ეს სტილი გავრცელდება მხოლოდ იმ სიმბოლოზე ან სიტყვაზე, რომელსაც მონიშნავთ. მაგალითად, როდესაც გინდათ, რომ მთელი აბზაცის მასშტაბით ერთი კონკრეტული სიტყვა იყოს ყოველთვის წითელი, ასეთ შემთხვევაში სიმბოლოების სტილი შეუცვლელია.

ტექსტი ავტომატურად თავსდება ახალ ფენაზე და თავისი მახასიათებლებით ის ვექტორული ობიექტია. საჭიროების შემთხვევაში თქვენ შეგიძლიათ მისი რედაქტირება, შრიფტისა და ზომის შეცვლა. მაგრამ ზოგიერთი მოქმედება (მაგ. ფილტრების მოქმედება, მისი ფუნჯით ან სხვა მსგავსი ინსტრუმენტით დამუშავება) შეუძლებელია და საჭირო ხდება მისი გარდაქმნა რასტრულ გამოსახულებად:

Layer -> Rasterize -> Type;

Type -> Rasterize Type Layer;

გარდა ამისა შეგიძლიათ ტექსტი ვექტორულ ფორმად გარდაქმნათ. ამის შემდეგაც ტექსტის რედაქტირება შეუძლებელი იქნება, თუმცა ის ვექტორულ ობიექტად დარჩება: Type -> Convert to Shape

### *ტექსტის ტრანსფორმაცია*

ტექსტის აკრეფისას ფორმატირების ზოლზე არის ღილაკი  . მისი დახმარებით შეგიძლიათ ტექსტის ტრანსფორმირება და მისთვის სხვადასხვა ფორმის მინიჭება. ეს ფუნქცია ანალოგიურია ფუნქციისა **დეფორმაცია** (Warp), რომელიც ობიექტების ტრანსფორმაციისას გამოიყენება. განსხვავება მხოლოდ ისაა, რომ აქ მხოლოდ გამზადებული შაბლონების მიხედვით ახდენთ მასზე მოქმედებას.

ზოგადად ტექსტის (როგორც ობიექტის) ტრანსფორმირება შეგიძლიათ ჩვეულებრივი ობიექტის მსგავსად - შეცვალოთ მისი ზომა, ფორმა, შეატრიალოთ სარკულად ჰორიზონტალურად თუ ვერტიკალურად და ა.შ.

ტრანსფორმირების ფუნქციები იხ. ტრანსფორმაციის ინსტრუმენტები.

**ფენების სტილები:**

რადგან ტექსტი ცალკე ფენაზე თავსდება, მასზე შესაძლებელია ფენების სტილების გამოყენება ისევე, როგორც სხვა ობიექტებზე. ფენების სტილების შესახებ იხ. ფენების სტილები.

## სავარჯიშოები

1. ჩამოთვალეთ ტექსტისა და აბზაცის რედაქტირების პარამეტრები;
2. შექმენით აბზაცისა და სიმბოლოების სტილები.
3. შექმენით გამოსახულებისთვის წარწერა. მოახდინეთ მისი ფორმატირება. ვიზუალური გაფორმებისთვის გამოიყენეთ **ტექსტის ტრანსფორმაცია** და/ან **ფენის სტილები**.
4. შექმენით ტექსტი. მოახდინეთ მისი გადაყვანა რასტრში და გამოიყენეთ სხვადასხვა ინსტრუმენტები და ბრძანებები მის ვიზუალურად გასაფორმებლად.
5. გამოსახულებაზე შექმენით მონიშნული არე ტექსტის სახით. მოახდინეთ მისი **დაფერვა** ან სხვადასხვა ეფექტების გამოყენებით ვიზუალურად გააფორმეთ.
6. შეარჩიეთ სასურველი წიგნის დასახელება და შექმენით ყდის წარწერა (ქართულად). შექმნისას გამოიყენეთ შესაბამისი ინსტრუმენტები.

## 1.5. ვექტორული ობიექტების შექმნა და დამუშავება

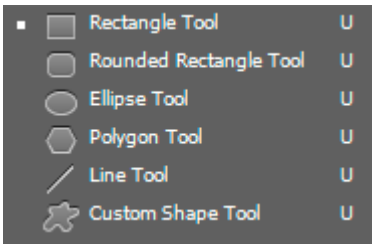
პარაგრაფის შესაბამისი თემატიკა

- მარტივი ვექტორული ობიექტების შექმნა და მათთან მუშაობა.
- ვექტორული ობიექტების შექმნა კალმის (Pen) გამოყენებით. თავისუფალი კალმის (Freeform Pen) გამოყენება, საკვანძო წერტილების დამატება წაშლა. მოქმედებები საკვანძო წერტილებზე.
- ვექტორული ობიექტის მთლიანი მონიშვნა. ცალკეული წერტილების მონიშვნა.

ვექტორული ობიექტი პროგრამაში ორი გზით შეიძლება შეიქმნას - გეომეტრიული ფორმებისა (Shape) და კალმის (Pen) გამოყენებით. ორივე ინსტრუმენტთა პანელზეა განთავსებული და მიეკუთვნება მოხაზულობის ინსტრუმენტებს (იხ. სურ. 1.1-9, ვ).

### *გეომეტრიული ფიგურები*

გეომეტრიული ფორმების შესაქმნელად მივმართავთ **მართკუთხედის ინსტრუმენტს** (Rectangle Tool) (U). ჯგუფში შედის კიდევ დამატებითი ინსტრუმენტები:




სურ. 1.5-1. ვექტორული ფორმები: მართკუთხედი (Rectangle), მართკუთხედი მომრგვალებული კუთხეებით (Rounded Rectangle), ელიფსი (Ellipse), მრავალკუთხედი (Polygon), ხაზი (Line) და მორგებული ფორმა (Custom Shape)

ინსტრუმენტის არჩევის შემდეგ, ფორმატირების ზოლზე გამოტანილი პარამეტრების დახმარებით ჩვენ შეგვიძლია განვსაზღვროთ ფორმის შევსებისა და კონტურის ფერები, მივუთითოთ კონტურის სისქე და ნაირსახეობა, ფორმის სიგანე ან/და სიმაღლე.

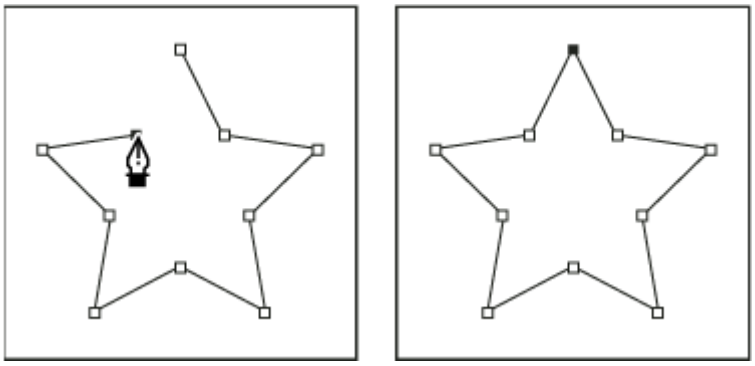

შევსებისთვის და კონტურისთვის შეგვიძლია სხვადასხვა გავაკეთოთ:

- შეგვიძლია შევსება ან კონტური საერთოდ მოვხსნათ;
- გამოვიყენოთ მხოლოდ ფერი (იხსნება ფერთა პალიტრა);
- გამოვიყენოთ გრადიენტი (იხსნება გრადიენტის პალიტრა. იხ. გრადიენტის ინსტრუმენტი (Gradient Tool));
- გამოვიყენოთ ფაქტურა (იხსნება ფაქტურების პალიტრა);
- ან ფერთა პალიტრიდან ავარჩიოთ ფერი, რომელის ფერთა ნიმუშებში არ არის მოცემული.

შექმნილი ობიექტი თავსდება ახალ ფენაზე. შეგიძლიათ გამოიყენოთ ტრანსფორმაციის ბრძანებები (იხ. ტრანსფორმაციის ინსტრუმენტები) ან გამოვიყენოთ სტილები (იხ. ფენების სტილები).

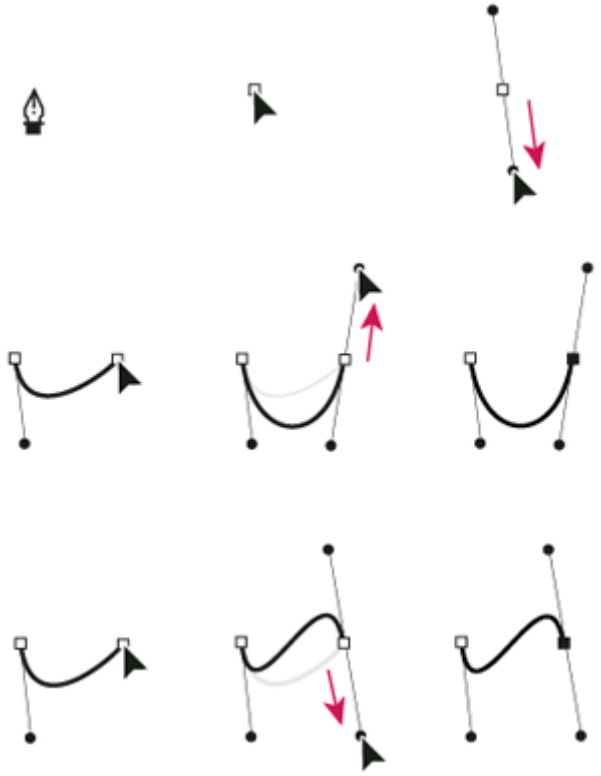
### *სასურველი ფორმების შექმნა (Pen Tool და Freeform Pen)*

კალმის (Pen) ინსტრუმენტით შესაძლებელია როგორ უბრალოდ ფორმების შექმნა, ის გამოსახულების კონტურებზე შემოტარება. ინსტრუმენტს იყენებთ ისევე, როგორც სწორხაზოვან ლასოს (Polygonal Lasso) - უნდა დასვათ წერტილები და ბოლო წერტილი შეაერთოთ პირველ წერტილთან (სურ. 1.5-2).



სურ. 1.5-2. კალმის (Pen) ინსტრუმენტის გამოყენება.

ინსტრუმენტის გამოყენება არ შემოიფარგლება მხოლოდ სწორი წირების გავლებით. შეგიძლიათ მრუდი წირების გავლებაც. ამისათვის მომდევნო წერტილის დასმისას ნუ აუშვებთ მაუსის ღილაკს, გეჭიროთ და ისე ამოძრავებთ. დაინახავთ, რომ საკვანძო წერტილს გაუჩნდება ორი, ერთმანეთის საპირისპირო ხაზები. ამ ხაზებზე მოქმედებით თქვენ მოქმედებთ საკვანძო წერტილზე და ცვლით წირის მოხაზულობას. წირს შეიძლება ნებისმიერი ფორმა მიანიჭოთ.



სურ. 1.5-3. მრუდი წირის შექმნა.

ინსტრუმენტი თავისუფალი კალამი (Freeform Pen) მუშაობს, როგორც ჩვეულებრივი ლასო. უნდა შემოავლოთ ის გამოსახულებაზე და ინსტრუმენტი თვითონ დასვამს საკვანძო წერტილებს.

ინსტრუმენტი **საკვანძო წერტილის დამატება** (Add Anchor Point) გაძლევთ საშუალებას დაამატოთ წირზე (სწორხაზოვანზე ან მრუდზე) საკვანძო წერტილები ნებისმიერი რაოდენობის.

ინსტრუმენტი **საკვანძო წერტილების წაშლა** (Delete Anchor Point) გაძლევთ საშუალებას წაშალოთ წირზე (სწორხაზოვანზე ან მრუდზე) არსებული საკვანძო წერტილები.

**დაიმახსოვრეთ:** *რაც უფრო ნაკლები წერტილისგან შედგება წირი, მით ნაკლებად ტეხილი გამოვა.*

წირის ხაზვისას თქვენ შეგიძლიათ შექმნათ მრუდი, მაგრამ როდესაც საჭირო გახდება უკვე არსებული სწორხაზოვანი მონაკვეთის გამრუდება, ამაში დაგეხმარებათ კუთხის ინსტრუმენტი (Convert Point). ის კალმის ინსტრუმენტთან ერთ ჯგუფში იმყოფება. მისი დახმარებით თქვენ მოქმედებთ საკვანძო წერტილზე, რის შედეგადაც თქვენ მოქმედებთ მთლიანად წირზე - ამრუდებთ, უცვლით ფორმას და ა.შ.

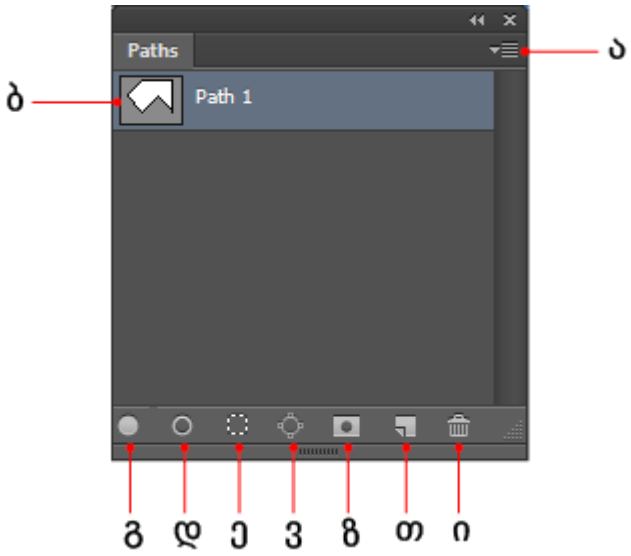
უკვე არსებულ წირის (თავისუფალი ფორმის წირი იქნება თუ გეომეტრიული ფორმა) მონიშვნა ჩვეულებრივი ინსტრუმენტებით ვერ ხერხდება. ამისათვის ინსტრუმენტთა პანელზე მოცემულია ორი ინსტრუმენტი: **კონტურის მონიშვნა** (Path Selection) და **ისარი** (Direct Selection).

პირველი ახდენს მთლიანი კონტურის მონიშვნას. მისი დახმარებით ასევე შესაძლებელი კონტურის გადაადგილება ტილოს ფარგლებში.

მეორე ინსტრუმენტის დახმარებით თქვენ ნიშნავთ საკვანძო წერტილს, რომელზეც გინდათ შემდეგ დამატებითი მოქმედებების შესრულება.

კონტურთან მუშაობისას ეს ორი ინსტრუმენტი შეუცვლელია.

კონტურთან სამუშაოდ შესაბამისი პანელის გამოძახება ხდება Window -> Path. მისი დახმარებით ჩვენ შეგვიძლია დამატებითი მოქმედებები შევასრულოთ.




სურ. 1.5-4. კონტურის პანელი (Path). ა - პანელის მენიუ; ბ - გამოსახულებაზე შექმნილი კონტური. რამდენი დამოუკიდებელო კონტური შეიქმნება, იმდენი აღმნიშვნელი იქნება აქ. კონტურებს შეგიძლიათ სახელებიც შეუცვალოთ; გ - კონტურის ფერით შევსება; დ - კონტურის მიხედვით კანტის შექმნა; ე - კონტურის გარდაქმნა მონიშნულ არედ; ვ - მონიშნული არის გარდაქმნა კონტურად; ზ - კონტურისგან გამოსახულების ნიღაბის მიღება; თ - ახალი სივრცის შექმნა, რომელზეც კონტური განთავსდება; ი - კონტურის წაშლა.

## სავარჯიშოები

1. შექმენით სხვადასხვა სახის გეომეტრიული ფიგურები. გამოიყენეთ მათთვის ტრანსფორმაციის ინსტრუმენტები და ფენის სტილები.
2. კალმის გამოყენებით გამოსახულების რომელიმე ობიექტის მიხედვით შექმენით ვექტორული ფორმა. შემდეგ ვექტორული ფორმა აქციეთ მონიშნულ არედ და მოახდინეთ მისი რედაქტირება.
3. ვექტორის შექმნისა და რედაქტირების ინსტრუმენტების დახმარებით, მარტივი გეომეტრიული ფიგურა აქციეთ რთულ ფორმად.

# 2. ვექტორული გამოსახულების შექმნა

## 2.1. დოკუმენტის საჭირო პარამეტრებით შექმნა

პარაგრაფის შესაბამისი თემატიკა

- ვექტორული გამოსახულება და მისი შედარება რასტრულ გამოსახულებასთან
- ფერთა მოდელების განხილვა
- ფერების პროფილების სინქრონიზაცია პროგრამებს შორის
- ვექტორული და რასტრული გამოსახულების გრაფიკული ფორმატები
- გრაფიკული რედაქტორის ინტერფეისის ტიპების გაცნობა
- ფანჯრების განლაგების რეჟიმების მიმოხილვა
- ილუსტრატორის პარამეტრების გაცნობა
- დოკუმენტის შესაბამისი ფორმატების გაცნობა

### ვექტორული გამოსახულება

ვექტორულ გრაფიკაში გამოსახულება წარმოდგენილია მათემატიკური ობიექტების, კონტურების სახით, რომლებშიც ჩასხმულია ფერი. თავის მხრივ, კონტური შედგება საკვანძო წერტილებისგან, რომლებიც ერთმანეთს უკავშირდებიან მრუდეებით. საკვანძო წერტილებზე მოქმედებისას ჩვენ შეგვიძლია ვცვალოთ მრუდის მოხაზულობა და საერთო ჯამში ფიგურის ან ფორმის იერსახე.

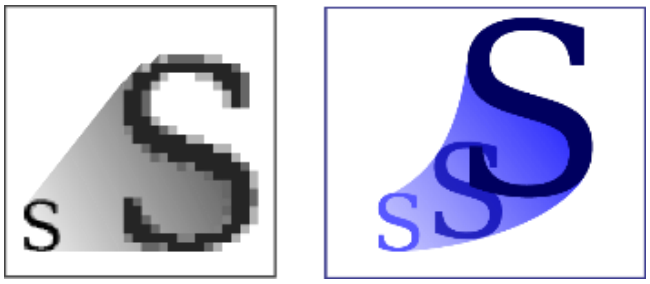
ვექტორულ გამოსახულებას თავისი დადებითი და უარყოფითი მხარეები გააჩნია.

დადებითი მხარეებიდან შეგვიძლია გამოვყოთ:

- ფაილის მცირე ზომა;
- შესაძლებელია გამოსახულების გადიდება უსასრულოდ. მაგალითად თუ გავზრდით წირის მრუდეს, ის ისეთივე სუფთა დარჩება, როგორიც იყო; თუ მრუდე წარმოდგენილია ტეხილი ხაზით (რაც მცირე ზომისას არ ჩანს), გადიდებისას ის აუცილებლად გამოჩნდება;

უარყოფითი მხარეები:

- შეუძლებელია ყველა ობიექტი მარტივად გამოისახოს ვექტორული სახით; იმისათვის, რომ ვექტორული გამოსახულება დაემსგავსოს ორიგინალს, შესაძლებელია დაგჭირდეთ ძალიან დიდი რაოდენობის ვექტორები;
- ვექტორული გამოსახულების გარდაქმნა რასტრულად ძალიან მარტივია. ხოლო უკან დაბრუნება (რასტრულიდან ვექტორისკენ) საკმაოდ რთული;



სურ. 2.1-1. რასტრული და ვექტორული გამოსახულებების შედარება

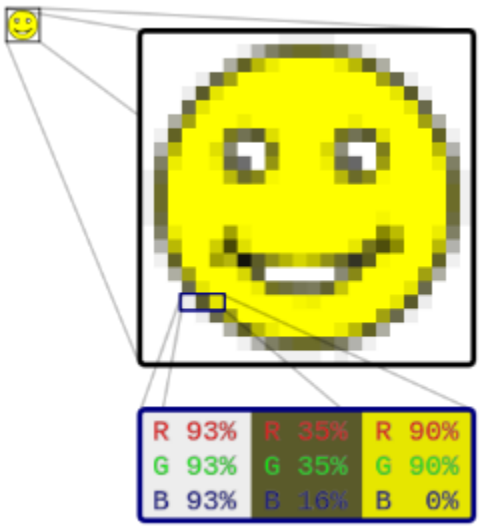
### რასტრული გამოსახულება

რასტრული გამოსახულება წერტილოვანი სტრუქტურის სახითაა წარმოდგენილი. როგორც წესი, წერტილს გააჩნია წრის ფორმა, თუმცა სხვა ფორმის წერტილებიც გამოიყენება: ელიფსური, რომბისებური და კვადრატული.

კომპიუტერულ გრაფიკაში რასტრული გამოსახულება იყოფა კვადრატულ ელემენტებად (ე. წ. პიქსელებად), რომელთა ერთობლიობა ქმნის გამოსახულებას ქაღალდზე, მონიტორზე ან სხვა რომელიმე ამსახველ მოწყობილობაზე. სიტყვა პიქსელი (pixel), ინგლისური Picture element – ის შემოკლებული ვარიანტია.

რასტრული გამოსახულების ძირითადი მახასიათებლებია:

- გამოსახულების ზომა პიქსელებში სიგანეზე და სიმაღლეზე (800x600px, 1024x768px და სხვა);
- გამოყენებული ფერების რაოდენობა ანუ ფერის სიღრმე;
- ფერთა მოდელები (RGB, CMYK, Lab და სხვა)
- გამოსახულების გარჩევადობა - სიდიდე, რომელიც განისაზღვრება წერტილების რაოდენობით მოცემულ ფართობზე, მაგალითად დუიმზე (1 დუიმი = 2.54 სმ).




სურ. 2.1-2. რასტრული გამოსახულების მაგალითი.

არსებობს გარჩევადობის ორი სახის საზომი ერთეული - ppi და dpi.

**PPI (pixel per inch)** - პიქსელების რაოდენობა დუიმზე. გამოიყენება ამსახველი მოწყობილობების გარჩევადობის შესაფასებლად: მონიტორი, ციფრული კამერის ეკრანი და სხვა;

**DPI (dot per inch)** - წერტილების რაოდენობა დუიმზე. გამოიყენება საბეჭდი მოწყობილობების ან სკანერების გარჩევადობის შესაფასებლად.

საბოლო ჯამში გარჩევადობა ნიშნავს ინფორმაციას. რაც უფრო მეტია გარჩევადობა, მით მეტია ინფორმაცია. რაც უფრო მეტი პიქსელია გამოსახულებაში, მით უფრო დიდ ფორმატზე შეიძლება იყოს ის აღბეჭდილი; რაც უფრო მჭიდროა პიქსელების განლაგება გამოსახულებაში, მით უფრო დეტალური და ხარისხიანია გამოსახულება.

ძირითადად გვხვდება შემდეგი ხარისხები:

72-150 pixel/inch - ამ ხარისხის გამოსახულებები ძირითადად ინტერნეტისთვის ან ელექტრონული პუბლიკაციებისთვის გამოიყენება;

150-250 pixel/inch - გამოიყენება ელექტრონული პუბლიკაციებში და პრინტერზე ბეჭდვისათვის;

300 pixel/inch და მეტი - გამოიყენება პოლიგრაფიაში და ბეჭდვით ტექნოლოგიებში. *შენიშვნა:* გამონაკლის შემთხვევაში, შესაძლებელია გამოყენებული იყოს მინიმუმ 250 pixel/inch ხარისხის გამოსახულება.

რასტრულ გრაფიკას გააჩნია როგორც დადებითი ისე უარყოფითი მხარეებიც.

დადებითი მხარეებია:

- რასტრულ გრაფიკაში შეიძლება ნებისმიერი სირთულის გამოსახულების შექმნა;
- პოპულარობა - რასტრული გრაფიკა პრაქტიკულად ყველგან გამოიყენება;
- რთული გამოსახულებების სწრაფად დამუშავება;

უარყოფითი მხარეები:

- მიღებული ფაილის დიდი მოცულობა;
- გამოსახულების ზომების გაზრდისას მისი ხარისხი მკვეთრად ეცემა.

### ფერთა მოდელები

ცნობილია, რომ მზის სხივი შეიცავს ბუნებაში არსებულ ყველა ფერს. მზის სხივის უწყვეტ სპექტრში არჩევენ 130-მდე ფერის ტონალობას. რეალურ ცხოვრებაში ამ ფერების ნახვა ცისარტყელაზე შეიძლება. მზის სხივის უწყვეტ სპექტრში ნიუტონმა გამოყო 7 ძირითადი ფერი: წითელი, ნარინჯისფერი, ყვითელი, მწვანე, ცისფერი, ლურჯი და იისფერი. ყველა სხვა ფერი, სწორედ ამ 7 ფერის სხვადასხვა რაოდენობით შერევის გზით მიიღება.

მრავალი ცდისა და დაკვირვების შედეგად, საერთაშორისო კოლორიმეტრული სისტემის მთავარი ფერები გამოავლინეს: წითელი (Red), მწვანე (Green) და ლურჯი (Blue). ინგლისური დასახელების პირველი ასოების მიხედვით, ფერთა ამ სისტემას დაერქვა - RGB.

RGB ფერთა სისტემა საფუძვლად უდევს მონიტორებს, სკანერების ერთ ნაწილს და კომპიუტერული პროგრამების უმრავლესობას. კომპიუტერის ეკრანზე ყველა ფერის მიღება, სწორედ ამ სამი ფერის სხვადასხვა პროპორციის შერევით არის შესაძლებელი. RGB მოდელის გარდა, არის ფერთა სხვა მოდელებიც: HSB; Lab; CMY; CMKY; YIQ და YCC.

მათ შორის ძირითადი მოდელები გამოიყენება: RGB – კომპიუტერულ გრაფიკაში; CMYK – პოლიგრაფიაში; Lab – ტექნიკური მიზნებისთვის; Grayscale – შავ-თეთრი ფოტო-სურათების რეჟიმი; Bitmap (Monochrome) – ორფეროვანი გამოსახულების რეჟიმი.

ფერთა მოდელის გარდა ასევე უნდა განისაზღვროს ფერის სიღრმე, რომელიც იზომება ბიტებში. რაც უფრო დიდია ფერის სიღრმე ე. ი. რაც უფრო მეტ ბიტიანია გამოსახულება, მით უკეთესად გადმოსცემს პიქსელი ფერს. პიქსელი გრაფიკული გამოსახულების უმცირესი ელემენტია და მისი დანიშნულება არის შეინახოს და გადმოსცეს ინფორმაცია ფერის შესახებ.

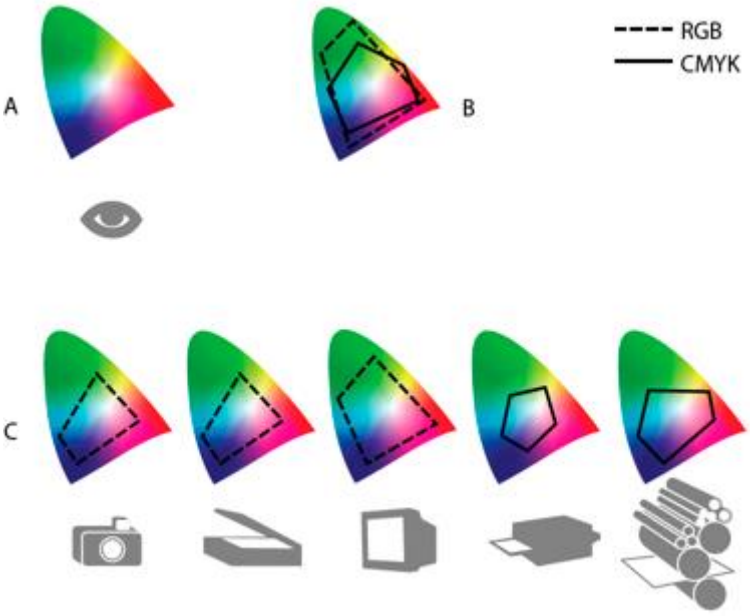
გამოსახულებაზე, რომლის ფერის სიღრმე 1 ბიტია ($2^1=2$), არის მხოლოდ ორი ფერი: თეთრი და შავი. 2 ბიტიან ($2^2=4$) გამოსახულებაზე ოთხი ფერია - თეთრი, შავი და ორი ნაცრისფერი. შავ-თეთრი ფოტოსურათების ფერის სიღრმე, უმეტეს შემთხვევაში, 8 ბიტია ($2^8=256$): შავი, თეთრი და 254 ნაცრისფრის სხვადასხვა ტონალობა. ე. ი. ამ შემთხვევაში პიქსელს შეუძლია გადმოსცეს ფერის 256 შესაძლო ვარიანტიდან ერთი. ნულოვან მნიშვნელობას შეესაბამება შავი ფერი, ხოლო 255-ს – თეთრი.

ფერად ციფრულ ფოტოსურათს აქვს 3 არხი (წითელი, მწვანე, ლურჯი) და 8 ბიტიანი ფერის სიღრმე თითო არხზე. ამიტომ, ასეთ გამოსახულებას 24 ბიტიანს ($2^{24}=16777216$) უწოდებენ და ამ შემთხვევაში, პიქსელს ფერის გადმოცემის 16,7 მილიონი შესაძლო ვარიანტი არსებობს. ფერის ეს რაოდენობა სავსებით საკმარისია გამოსახულების ფერის რეალურად გადმოცემისათვის, თუმცა არსებობს 48 ბიტიანი გამოსახულებები (16 ბიტი თითო არხზე), რაც 281 ტრილიონი ფერის გადმოცემის შესაძლო ვარიანტს იძლევა. ადამიანის თვალი ამდენ ფერს ვერ არჩევს, მაგრამ გამოსახულების რედაქტირებისას, ბიტების დიდი რაოდენობა შედეგზე დადებითად აისახება. 16 ბიტიანი გამოსახულებით პროფესიონალი ფოტოგრაფები მუშაობენ.

არც ერთ მოწყობილობას, რომელიც საგამომცემლო პროცესში მონაწილეობს, არ შეუძლია იმ ფერების სრული დიაპაზონის ასახვა, რომელსაც ადამიანის თვალი აღიქვამს. ყოველი მოწყობილობა იყენებს

განსაზღვრულ **ფერთა სივრცეს**. მის ფარგლებში მოცემულია ფერთა მხოლოდ კონკრეტული დიაპაზონი. ამას **მოცვა** ეწოდება.

ზოგ ფერთა მოდელში (მაგ. CIE L*a*b) ფერთა სივრცე ფიქსირებულია. მათში გათვალისწინებულია ადამიანის თვალის მიერ ფერის აღქმის შესაძლებლობა. ასეთ მოდელებს **მოწყობილობისგან დამოუკიდებელს** უწოდებენ. სხვა ფერთა მოდელებში (RGB, HSL, HSB, CMYK და ა.შ.) შესაძლოა ძალიან ბევრი ფერთა სივრცე არსებობდეს და რადგანაც სხვადასხვაა მოწყობილობებისთვის ისინი განსხვავდებიან, მათ **მოწყობილობაზე დამოკიდებულს** უწოდებენ.




სურ. 2.1-3. სხვადასხვა მოწყობილობისა და დოკუმენტის ფერთა მოცვა.
A – Lab ფერთა სივრცე; B - დოკუმენტები (სამუშაო სივრცე); C - მოწყობილობები.

ფერების შეთანხმების პრობლემები იმითაა გამოწვეული, რომ სხვადასხვა მოწყობილობა და პროგრამები იყენებენ სხვადასხვა ფერთა სივრცეს. ამ პრობლემის გადაწყვეტა შეიძლება იყოს მექანიზმი, რომელიც ზუსტად განსაზღვრავს ფერს და ცვლილებების გარეშე გადასცემს მას ერთი მოწყობილობიდან მეორეზე. ეს მექანიზმი **ფერთა მართვის სისტემაა**. **ფერთა მართვის სისტემა** ადარებს ფერთა სივრცეს, სადაც შეიქმნა ფერი, იმ ფერთა სივრცეს, სადაც ეს ფერი უნდა იყოს ასახული და შეაქვს კორექტივები, რომ სხვადასხვა მოწყობილობაზე ფერების ასახვა მაქსიმალურად იყოს შეთანხმებული.

ფერთა მართვის სისტემა ფერების გარდაქმნას ფერების პროფილის დახმარებით ახდენს. პროფილი არის მოწყობილობის ფერთა სივრცის მათემატიკური აღწერა. მაგალითად, სკანერის პროფილის მიხედვით ფერთა მართვის სისტემა განსაზღვრავს, როგორ „ხედავს" სკანერი ფერებს. Adobe-ის ფერთა მართვის სისტემა იყენებს ICC პროფილებს. ამ ფორმატს ფერების საერთაშორისო კონსორციუმი (International Color Consortium, ICC) განსაზღვრავს და ის კროსპლატფორმულია.

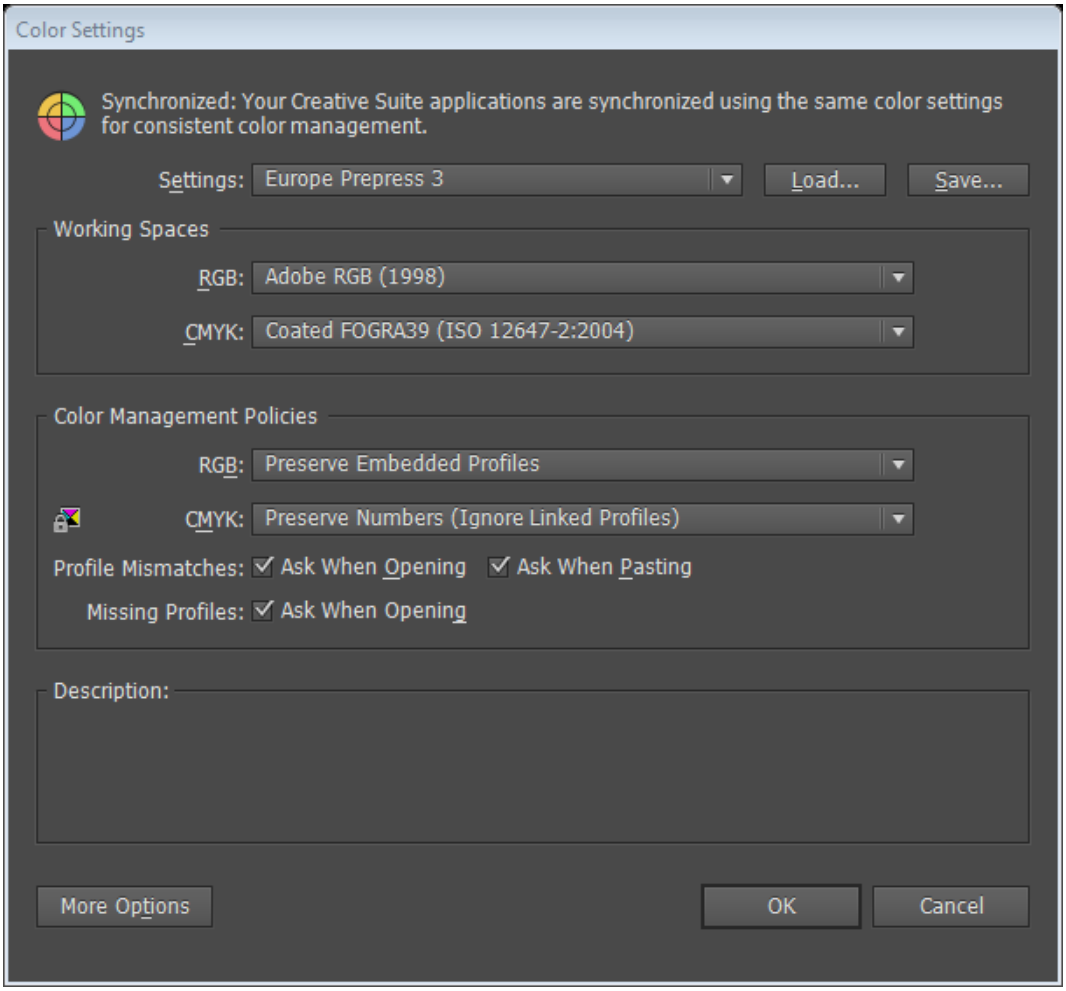
**შენიშვნა:** *ფერების მართვა და ფერთა კორექცია განსხვავდება ერთმანეთისგან. ფერთა მართვის სისტემა ვერ გაასწორებს გამოსახულებას, რომელიც ტონალური ან ფერთა ბალანსის დარღვევით იყო შენახული.*

### ფერების ერთიანი პროფილის გამოყენება

ტექნიკური დიზაინერი მუშაობის პროცესში იყენებს ოთხ პროგრამას: ფოტოშოპს, ილუსტრატორს, ინდიზაინს და აკრობატ-პროს. ყველა ამ პროგრამებს შორის უნდა მოხდეს ერთიანი ფერთა სისტემის გამოყენება, რადგან არ მოხდეს ფერთა ცდომილება პროგრამებს შორის. ამას ჩვენ ფერების პროფილის დახმარებით ვაკეთებთ.

თითოეულ პროგრამას გააჩნია ფერების მოწყობის ფუნქცია.

ფოტოშოპს, ინდიზაინს და ილუსტრატორს ეს ფუნქცია მენიუ Edit-ში აქვთ განთავსებული: Edit -> Color Settings. აკრობატში უნდა გამოვიძახოთ პროგრამის პარამეტრები (Edit -> Preferances) და მარცხნივ ჩამონათვალში არის Color Management.



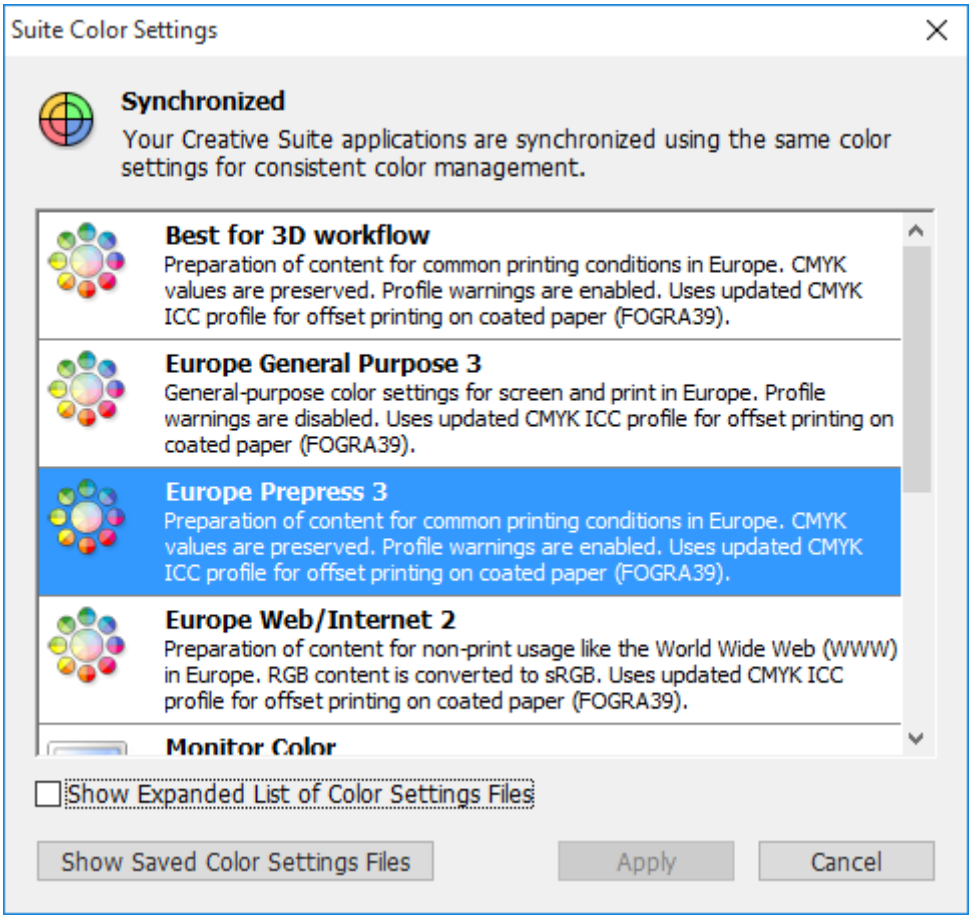

სურ. 2.1-4. ფერების პროფილის დაყენება პროგრამა ილუსტრატორში.

მიუთითებს, რომ პროგრამებს შორის ფერების პროფილები სინქრონიზებულია.

მიუთითებს, რომ პროგრამებს შორის ფერები პროფილები არაა სინქრონიზებული.

ერთიანად და ადვილად რომ მოვახდინოთ ფერების სინქრონიზაცია პროგრამებს შორის, უნდა გამოვიყენოთ პროგრამა ბრიჯი (Bridge). პროგრამა მხოლოდ ფაილების მენეჯერს არ წარმოადგენს. მასში ბევრი საინტერესო ფუნქციაა მოთავსებული და ამ პროგრამის გამოყენება საერთოდ ცალკე განხილვის თემაა.

ამ შემთხვევაში ჩვენ განვიხილავთ, თუ როგორ ვმართოთ ფერთა პროფილები პროგრამებს შორის. ამისათვის გამოიძახეთ Edit -> Creative Suite Color Settings (Ctrl + Shift + K).



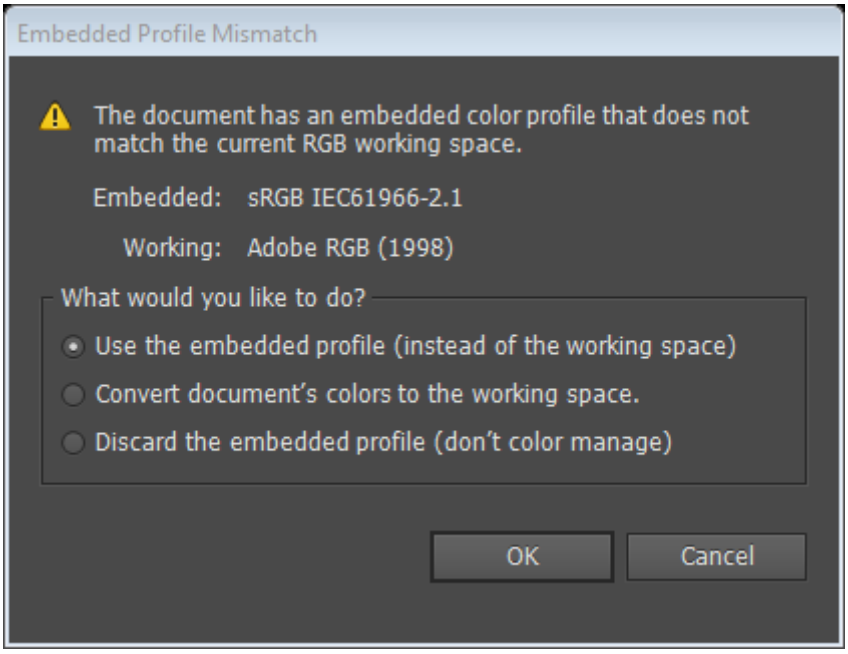

სურ. 2.1-5. ფერების ერთიანი პროფილის დაყენება პროგრამებისთვის.

ჩამონათვალში შეგიძლიათ აირჩიოთ ფერების პროფილები. ყველა პროფილს თავისი დასახელება აქვს და მცირე ზომის აღწერაც ერთვის მას. ჩვენს შემთხვევაში ყველსა გამოსადეგი არის Europe Prepress 3 პროფილი. აირჩიეთ და დააწკაპუნეთ ღილაკს Apply. ყველა შესაბამის პროგრამაში ეს პროფილი ავტომატურად დაყენდება. ამის შემდეგ, როდესაც გამოსახულების ფერთა კორექციას გააკეთებთ ფოტოშოპში, დარწმუნებული იყავით, რომ მას სწორად აღიქვამს ილუსტრატორიც, ინდიზაინიც და აკრობატიც.

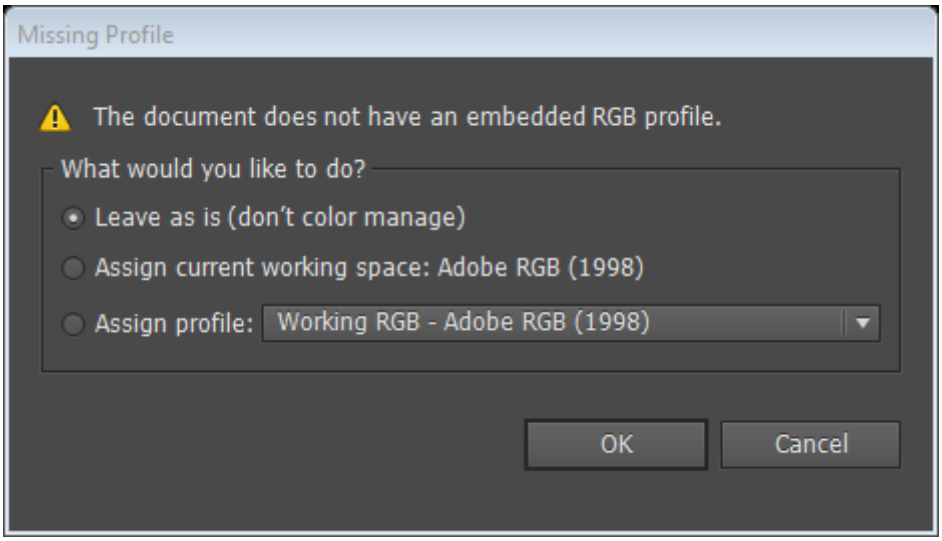
უფრო ვრცლად სინქრონიზაციის შესახებ შეგიძლიათ იხილოთ სტატია [ფერების სინქრონიზაცია Adobe-ის პროგრამებს შორის](#).

გამოსახულების შენახვის დროს მასში შეიძლება ჩანერგილი იყოს ფერების პროფილი. ფაილში ამ ინფორმაციას ცალკე ადგილი უკავია. გამოსახულების გახსნისას, პროგრამა კითხულობს ამ ინფორმაციას.

თუ გამოსახულებაში ჩანერგილი ფერთა პროფილი ემთხვევა პროგრამაში დაყენებულ ფერთა პროფილს, მაშინ ის პირდაპირ იხსნება. მაგრამ თუ მოხდა ისე, რომ ჩანერგილი პროფილი არ ემთხვევა პროგრამისას ან საერთოდ არ გააჩნია გამოსახულებას პროფილი, მაშინ ამის შესახებ გამოდის შესაბამისი გაფრთხილება:




სურ. 2.1-6. მსგავსი ფანჯარა გამოდის იმ შემთხვევაში, როდესაც გამოსახულებაში ჩანერგილი პროფილი განსხვავდება პროგრამაში დაყენებული პროფილისგან. გამოსახულებაში ჩანერგილია sRGB პროფილი, ხოლო მიმდინარე (სამუშაო) პროფილია Adobe RGB. აქ გვაქვს სამი ვარიანტი: გამოვიყენოთ გამოსახულების ფერის პროფილი, მოხდეს ფერების კონვერტაცია ჩვენს სამუშაო პროფილზე, წაიშალოს ინფორმაცია პროფილების შესახებ.




სურ. 2.1-7. მსგავსი ფანჯარა გამოდის იმ შემთხვევაში, თუ ფაილში ფერის პროფილი არ არის ჩანერგილი. ამ შემთხვევაშიც სამი ვარიანტი გვაქვს ასარჩევად: დავტოვოთ, როგორც არის (პროფილის გარეშე), მივანიჭოთ მიმდინარე (სამუშაო) პროფილი, მოვარგოთ რომელიმე პროფილი პროფილების სიიდან.

თუ თქვენ აპირებთ გამოსახულებასთან მუშაობას, მაშინ აუცილებლად მოარგეთ მას მიმდინარე (სამუშაო) ფერთა პროფილი, რომ მომავალში ფერთა კორექცია იქნება თუ რაიმე სხვა მსგავსი მოქმედება, ფერები დაჯდეს და აისახოს სწორად.

### ვექტორული ფაილის ფორმატები

**AI (Adobe Illustrator)** - გრაფიკული ინფორმაციის შენახვის ვექტორული ფორმატი. შექმნილია Adobe Systems-ის მიერ. თავისი არსით ეს PGF (Portble Graphic Format - პორტატული გრაფიკული ფორმატი) ფორმატის ფაილია და შეიძლება განვიხილოთ, როგორც შუალედური პროდუქტი, საიდანაც შეგვიძლია მივიღოთ როგორც EPS, ისე PDF ფაილები.

**EPS (Encapsulate PostScipt)** - ფაილების ფორმატი, რომელიც დაფუძნებულია PostSript[1]-ის ენაზე და განკუთვნილია პროგრამებს შორის გრაფიკული ინფორმაციის გასაცვლელად. ფორმატი ფართოდ გამოიყენება პოლიგრაფიაში და შეიძლება შეიცავდეს როგორც რასტრულ, ისე ვექტორულ გამოსახულებებს ან ორივეს ერთად. EPS ფორმატს ასევე გააჩნია Grayscale, RGB, CMYK და Lab ფერთა მოდელების მხარდაჭერა.

**PDF (Portable Document Format)** - ელექტრონული დოკუმენტების ფორმატი, რომელიც კომპანია Adobe Systems-მა შეიმუშავა. მისი პირველადი დანიშნულებაა პოლიგრაფიული პროდუქციის წარმოდგენა ელექტრონული სახით. PDF საშუალებას იძლევა ფაილში ჩაშენებული იყოს საჭირო შრიფტები, ვექტორული და რასტრული გამოსახულებები, ბმულები, ფორმები, მულტიმედია (აუდიო/ვიდეო) კონტენტი. გააჩნია Grayscale, RGB, CMYK და Lab ფერთა მოდელების მხარდაჭერა და პოლიგრაფიისათვის საჭირო საკუთარი ტექნიკური ფორმატები, მაგალითად PDF/X-1a, PDF/X-3.

### რასტრული გამოსახულების ფაილების ფორმატები

ილუსტრატორი რასტრული ფაილების მხოლოდ იმპორტი ან ექსპორტი შეუძლია, მაგრამ არა მათი რედაქტირება. რასტრული ფაილის საფუძველზე თქვენ ვექტორულ გამოსახულებას ქმნით ან ვექტორული გამოსახულება დააექსპორტოთ რასტრულ ფაილში (მაგ., ინტერნეტში განსათავსებლად).

**JPEG (Join Photographic Expert Group)** - ამ ფორმატის უპირატესობა არის მისი მოქნილობა. გამოსახულების რედაქტირების ყველა პროგრამა მუშაობს ამ ფორმატთან, შენახული ფაილების მოცულობა მცირეა, რადგან შენახვისას გამოიყენება *შეკუმშვა*. შეკუმშვისას გარკვეული ინფორმაცია *იკარგება* - ხდება ცალკეული პიქსელების ფერის გაშუალება, რის შედეგადაც მცირდება დეტალიზაცია და იცვლება ფერის ტონალობები. რაც უფრო მაღალია შეკუმშვის ხარისხი, მით უფრო მცირეა ფაილის მოცულობა, მაგრამ ამავდროულად გამოსახულების ხარისხი მკვეთრად ფუჭდება; და პირიქით, რაც უფრო მცირეა შეკუმშვის ხარისხი, ფაილის ზომა არ მცირდება და გამოსახულების ხარისხის მეტნაკლებად მისაღები ხდება. ერთი და იმავე ფაილზე მუშაობისას, ყოველ ახალ შენახვაზე ხდება ინფორმაციის მორიგი პორციის დაკარგვა. ამის გამო ამ ფორმატის გამოყენება ბეჭდვითი პროექტებისთვის მიუღებელია. მისი გამოყენება მისაღებია ელექტრონული პუბლიკაციებისთვის, ელფოსტით გასაგზავნად ან ინტერნეტში განსათავსებლად.

**PNG (Portable Network Graphics)** - GIF ფორმატის ჩასანაცვლებლად შეიქმნა. მისგან განსხვავებით, მუშაობს ფერების დიდ რაოდენობასთან და ასვე შეუძლია შეინახოს ფენის გამჭვირვალობა. მასში გამოყენებულია ისეთი მძლავრი უდანაკარგო შეკუმშვის ალგორითმი, როგორიცაა LZW. GIF და PNG ფორმატში შენახული გამოსახულებები ძირითადად გამოიყენება ინტერნეტში განსათავსებლად და ელექტრონულ პუბლიკაციებში.

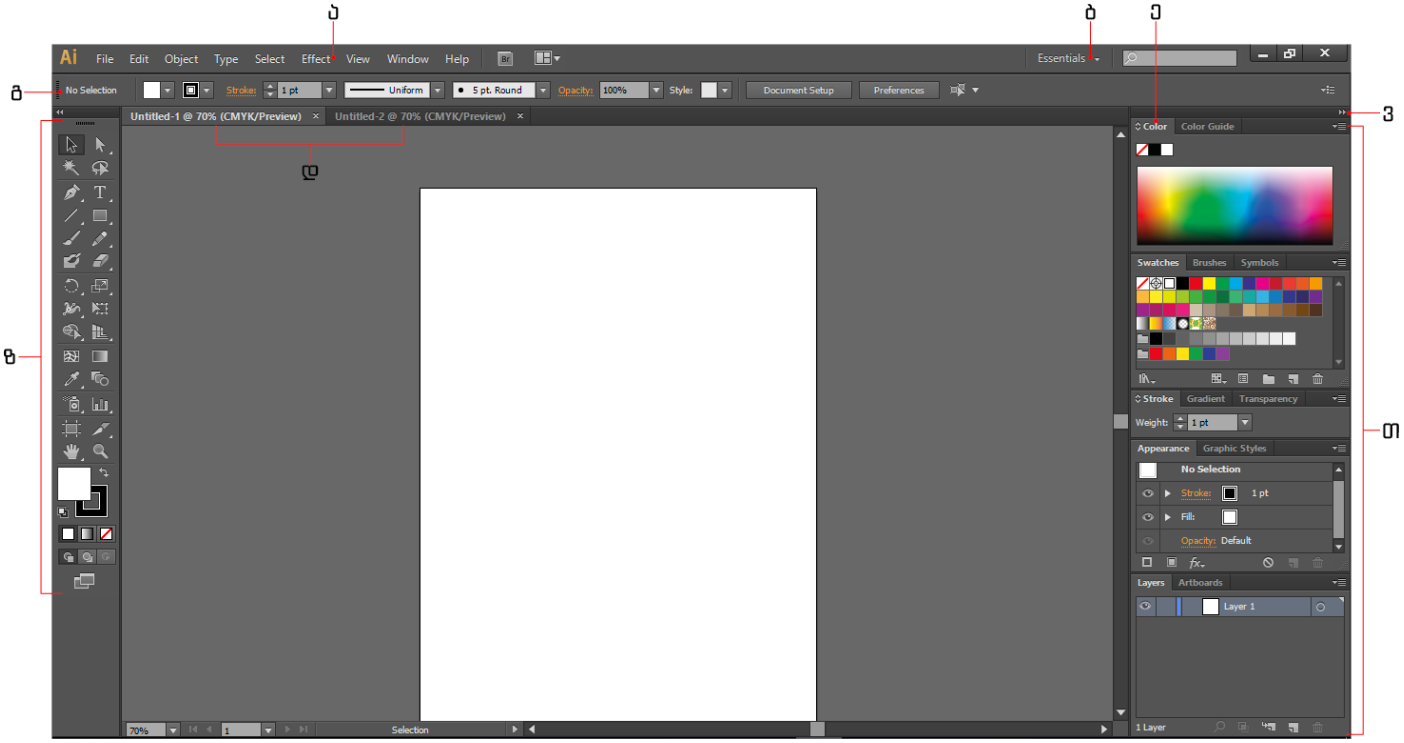
**BMP (BitMap)** - ფორმატი შექმნილია Microsoft-ის მიერ და ორიენტირებულია Windows-ის ოპერაციულ სისტემაში გამოსაყენებლად. იხმარება რასტრული გამოსახულების წარმოსადგენად პროგრამულ რესურსებში. იყენებს ინფორმაციის დაკარგვის გარეშე შეკუმშვის მარტივ ალგორითმს RLE (Run Length Encoding).

[1] PostScript - გვერდების აღწერის ენა. ძირითადად გამოიყენება სამაგიდო საგამომცემლო სისტემებში.

**TIFF** - ამ ფორმატს აქვს მრავალფენიანი დოკუმენტების მხარდაჭერა; გამოსახულება შეუძლია შეინახოს, როგორც შეკუმშვით (LZW ან ZIP), ისე შეკუმშვის გარეშე და მისი ხარისხი არ ფუჭდება. თუმცა მასთან მუშაობისათვის თქვენ დაგჭირდებათ პროგრამა ფოტოშოპი ან რომელი მსგავსი რედაქტორი.

**PSD (PhotoShop Document)** - ფორმატი არის ფოტოშოპის საკუთარი ფორმატი. ერთადერთი ფორმატი, რომელიც ინახავს ყველაფერს ნებისმიერი სახით და პროგრამის ყველა მოთხოვნას უჭერს მხარს. შენახვის დროს ინარჩუნებს დოკუმენტის ფენოვან სტრუქტურას, რათა მარტივი იყოს გამოსახულების შემდგომი რედაქტირება. ამ ფორმატში შეიძლება შუალედური ნამუშევრის შენახვა რედაქტირების დასრულებამდე. ილუსტრატორს შეუძლია ამ ფორმატის იმპორტი/ექსპორტი ფენების სტრუქტურის შენარჩუნებით.

## პროგრამა Adobe Illustrator-ის ინტერფეისი




სურ. 2.1-8. ა - პროგრამის მენიუ; ბ - სამუშაო სივრცეებს შორის გადამრთველი; გ - ფორმატირების ზოლი. მისი პარამეტრები იცვლება იმის მიხედვით, თუ რომელი სამუშაო ინსტრუმენტი გაქვთ არჩეული; დ - სანიშნეები, რომლებიც მიუთითებენ გახსნილ ფაილს. მათი საშუალებით შესაძლებელია ფაილებს შორის გადასვლა; ე - სამუშაო პანელის დასახელება; ვ - სამუშაო პანელის ნიშნულის სახით ჩაკეცვის ღილაკი; ზ - ინსტრუმენტთა პანელი; თ - ვერტიკალურად ჩამაგრებული სამუშაო პანელები.

## სამუშაო სივრცე

თქვენი სამუშაო პროფილის ან კონკრეტული ამოცანის სპეციფიკიდან გამომდინარე, შესაძლოა დაგჭირდეთ სხვადასხვა ტიპის სამუშაო პანელები. გარდა იმ სამუშაო პანელებისა, რომელსაც სურათზე ხედავთ (სურ. 2.1-8, თ), შესაძლებელია კიდევ სხვების გამოჩენაც. ამისათვის უნდა შეხვიდეთ პროგრამის მენიუში Window და ჩამოშლილი სიიდან აირჩიოთ თქვენთვის სასურველი პანელი ან დამალოთ უსარგებლო.

პანელები შეგიძლიათ დააჯგუფოთ, შეურჩიოთ მათ პოზიცია და დააყენოთ, როგორ უნდა ჩანდეს ის ეკრანზე (ღილაკის სახით თუ გაშლილ მდგომარეობაში) ანუ მოაწყოთ სამუშაო სივრცე ისე, როგორც თქვენ გჭირდებათ.

ილუსტრატორს გააჩნია რამდენიმე მზა სამუშაო სივრცე. სასურველი სამუშაო სივრცის არჩევისას ხდება ინტერფეისის ნაწილობრივ შეცვლა და ამ სივრცისთვის დამახასიათებელი სამუშაო პანელების გამოტანა. გარდა ამისა ეს მოცემული სამუშაო სივრცე შეგიძლიათ გადააწყოთ თქვენი სურვილისამებრ.

**Essentials** – ძირითადი სამუშაო სივრცე. ამ დროს გამოტანილია ძირითადი სამუშაო პანელები, როგორიცაა ფერების პალიტრა და ფერის ნიმუშები, ფენებთან და კონტურებთან სამუშაო პანელები და სხვა.

**Layout** - სივრცე, სადაც გამოტანილია ყველა საჭირო ინსტრუმენტი მაკეტთან სამუშაოდ. აქ გამოდის ბევრი სამუშაო პანელი, რომელთაგანაც ზოგი ჩაკეცილ, ზოგი კი გაშლილ მდგომარეობაში არიან.

**Painting** - სივრცე, რომელიც გამოადგება მხატვარს ან ილუსტრატორს. აქ გამოტანილია ფერების პალიტრა, მზა ფუნჯები და ფენებთან რეგულირების ფანჯარა.

**Printing and Proofing** – სამუშაო სივრცე, რომელიც მორგებულია იმ შემთხვევისათვის, როდესაც პროექტის გადამოწმება გსურთ მისი დასაბეჭდად გაგზავნამდე.

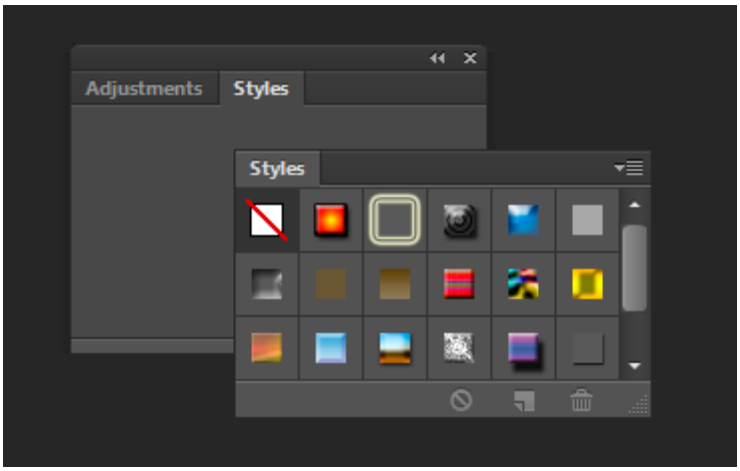
**Tracing** – Trace ანუ გადახატვა – ეს არის პროცესი, როდესაც ხდება პიქსელური გამოსახულების ვექტორულში გადაყვანა. ეს სამუშაო სივრცე მორგებულია ამ შემთხვევისათვის.

**Typography** - სამუშაო სივრცე მათთვის, ვინც აპირებს იმუშაოს შრიფტების გამოყენებით და შექმნას ტიპოგრაფიული ეფექტები პოსტერებისთვის, ბანერისთვის და სხვა მსგავსი სახის პროექტისთვის.

**Web** -სამუშაო სივრცე მათთვის, ვინც ამზადებს მაკეტს ინტერნეტისთვის ან მოწყობილობისათვის.

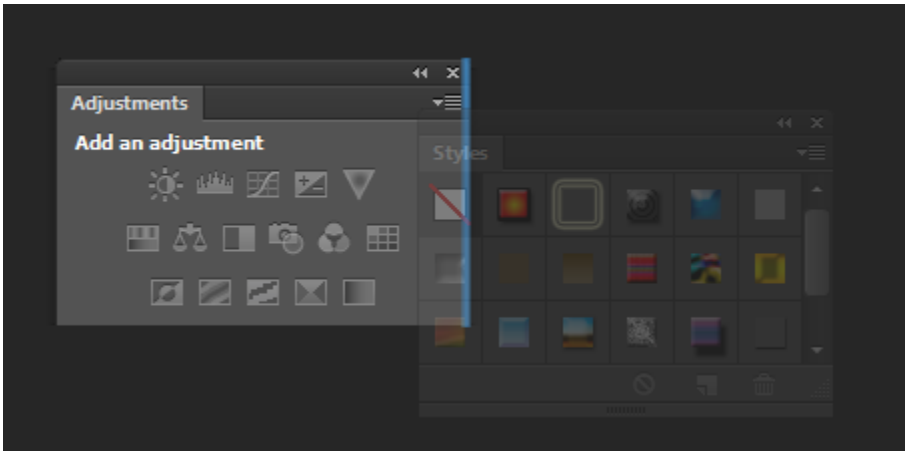
სამუშაო პანელებთან რომ გაგიადვილეთ მუშაობა, შესაძლებელია მათი დაჯგუფება. მაგალითისთვის თუ შეხედავთ გამოსახულებას (სურ. 2.1-8, თ), დაინახავთ რომ პანელები Swatches, Brushes და Symbols ერთად არიან დაჯგუფებული. შესაბამის დასახელებაზე დაჭერისას ხდება პანელებს შორის გადართვა და მისი ფუნქციების გამოჩენა. ასევე შეგიძლიათ უკვე არსებული ჯგუფების დაშლა და პანელების გადალაგება თქვენი სურვილის მიხედვით.

ჯგუფიდან რომ მოხსნათ პანელი, ამისათვის მაუსის კურსორით მოკიდეთ და გაიტანეთ ცალკე:

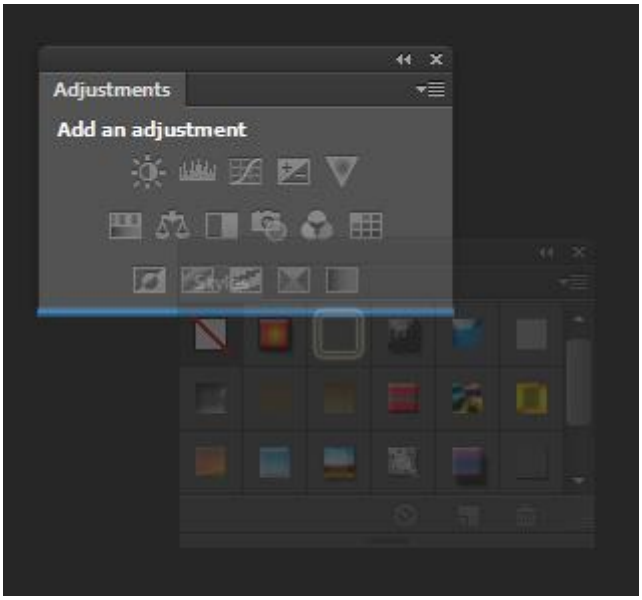



სურ. 2.1-9. პანელების ჯგუფის დაშლა.

იმისათვის, რომ პანელები ჰორიზონტალურად ან ვერტიკალურად ჩაამაგროთ, პანელი მიიტანეთ მეორე პანელის კიდესთან (გვერდიდან ან ზემოდან/ქვემოდან). გამოჩნდება ლურჯი ზოლი, რაც კავშირის მაჩვენებელია:

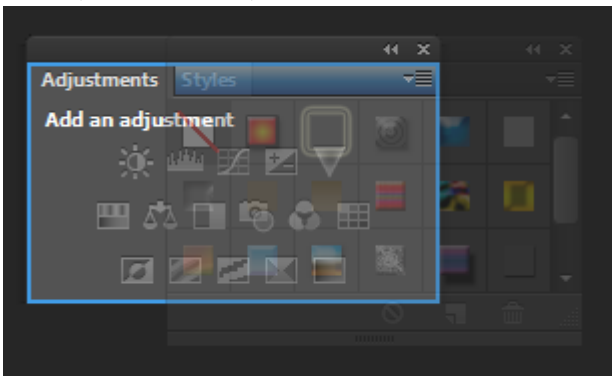



სურ. 2.1-10. პანელების ჰორიზონტალურად დაჯგუფება.




სურ. 2.1-11. პანელების ვერტიკალურად დაჯგუფება.

იმისათვის, რომ დააჯგუფოთ პანელები (გვერდიგვერდ), პანელი მიიტანეთ მეორე პანელის დასახელებასთან ახლოს:

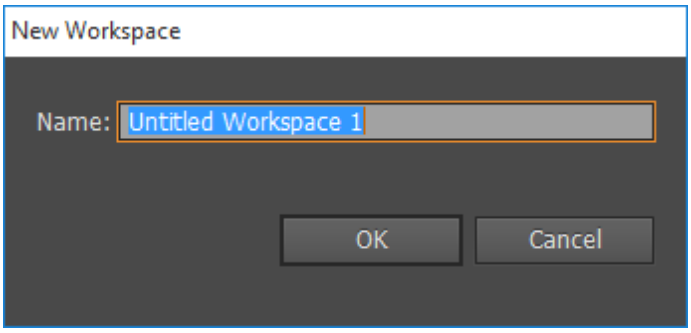



სურ. 2.1-12. პანელების დაჯგუფება.

მას შემდეგ, რაც თქვენ დაალაგებთ საჭირო პანელებს, სასურველი იქნება ეს განლაგება შეინახოთ, რადგან სხვა სამუშაო სივრცეზე გადართვის შემდეგ მოგიწევთ ყველა პროცედურის ხელახლა გავლა.

ამისათვის შედით Window -> Workspaces. აქ თქვენ ნახავთ სამუშაო სივრცის განამზადებს და დამატებით სამ ფუნქციას.

იმისათვის, რომ შეინახოთ თქვენს მიერ დაყენებული სამუშაო სივრცე, აირჩიეთ Window -> Workspaces -> New Workspace (ახალი სამუშაო სივრცე).




სურ. 2.1-13. ახალი სამუშაო სივრცის შენახვის ფანჯარა

გამოსულ ფანჯარაში შეიყვანეთ სასურველი სახელი და დააჭირეთ OK. თქვენს მიერ შექმნილი სივრცე დაემატება სამუშაო სივრცეების სიაში.

მუშაობის პროცესში შესაძლოა დაგჭირდეთ სხვა პანელის გამოძახება, არსებულის დახურვა, ფანჯრების გადაჯგუფება და სხვა. რომ დაბრუნდეთ სამუშაო სივრცის საწყის განლაგებაზე, შედით Window -> Workspaces -> Reset Workspace (სამუშაო სივრცის გადატვირთვა). სამუშაო სივრცე აღდგება იმ სახით, რა სახითაც თქვენ შეინახეთ.

თქვენ შეგიძლიათ მართოთ შექმნილი სამუშაო სივრცეები. ამისათვის შედით Window -> Workspaces -> Manage Workspace (სამუშაო სივრცის მართვა). ამ ფანჯრის დახმარებით შეგიძლიათ შექმნათ ან წაშალოთ სამუშაო სივრცე.

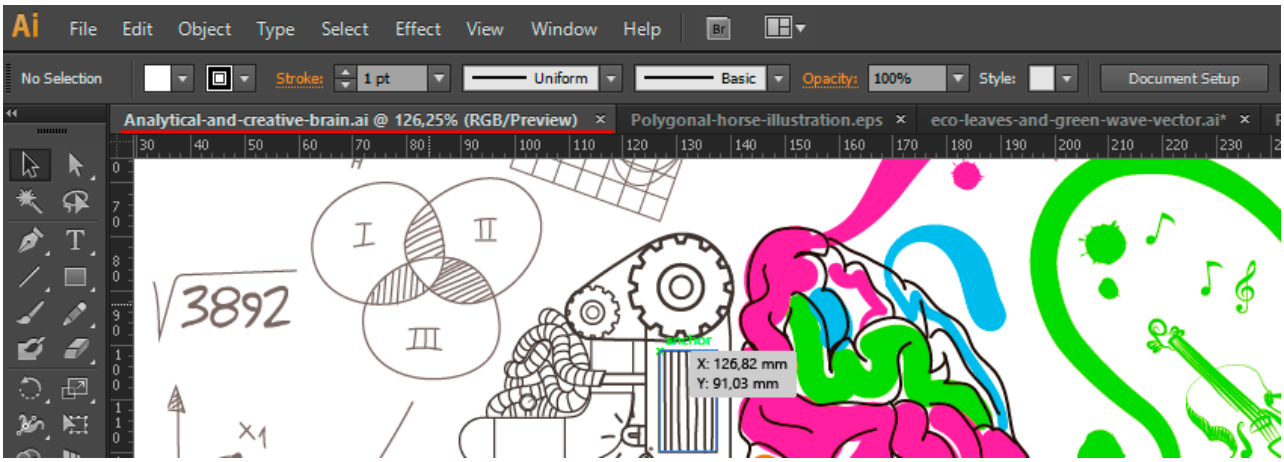
## ინსტრუმენტთა პანელი


სურ. 2.1-14. ილუსტრატორის ინსტრუმენტთა პანელი. ყოველი ინსტრუმენტი შეიცავს დამატებით ფუნქციებს, რომელთა მართვა შესაძლებელია ფორმატირების ზოლიდან (სურ. 2.1-8, ბ) ან მართვის პანელის (სურ. 2.1-8, თ) დახმარებით. ასევე თუ ორჯერ სწრაფად დააწვებით ინსტრუმენტს, ხდება მისი მახასიათებლების ფანჯრის გამოტანა.

## ფანჯრების განლაგება

ილუსტრატორში შესაძლებელია ერთდროულად რამდენიმე ფაილის გახსნა და მათთან მუშაობა. ეს გაძლევთ საშუალებას ობიექტების ან რაიმე ელემენტის ერთი ფაილიდან მეორეში მარტივად, უბრალო გადათრევით გადაიტანოთ, მოახდინოთ გამოსახულებების შედარება ან უბრალოდ რამდენიმე მოცემული ნამუშევრიდან გსურთ აარჩიოთ საუკეთესო პროექტში გამოსაყენებლად.

უნდა გაითვალისწინოთ, რომ თითოეული ფაილი დამოუკიდებელ ფანჯარაში იხსნება.



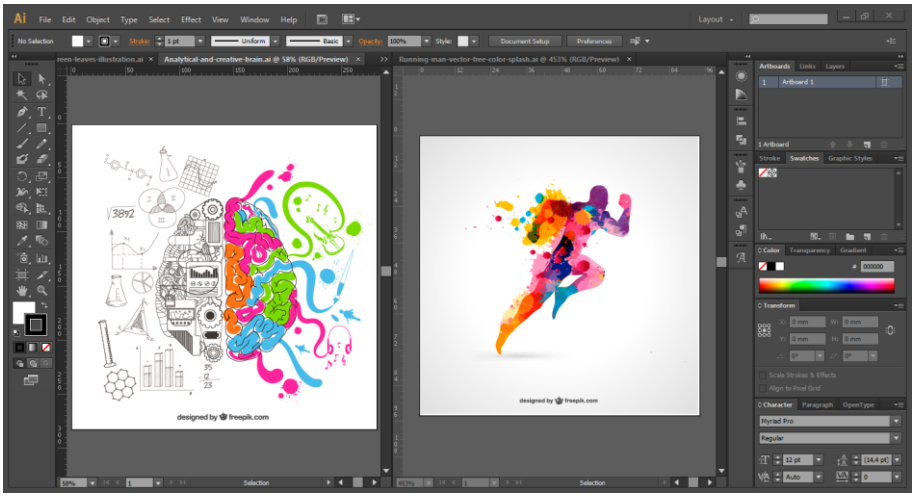

სურ. 2.1-15. ფანჯრის დასახელება ემთხვევა ფაილის სახელწოდებას. ეს გეხმარებათ დაინახოთ, რომელი ფაილები გაქვთ გახსნილი. ასევე აქ არის სხვა სახის ინფორმაციაც: Analytical-and-creative-brain.ai - ფაილის სახელწოდება და ფორმატი; @ - გამყოფი ნიშანი; 126,25% - გვაჩვენებს გამოსახულების სამუშაო მასშტაბს; RGB - გვაჩვენებს რა ფერთა მოდელი აღწერს გამოსახულებას; * - მიუთითებს, რომ ფაილში შეტანილია ცვლილებები.

სტანდარტულად პროგრამაში ფანჯრების განლაგება არის ჩანართის სახით (სურ. 2.1-8, დ). თუმცა ჩვენ მისი ცვლა შეგვიძლია და გამოვაჩინოთ გახსნილი ფაილები ისე, რომ გაგვიადვილდეს მათთან მუშაობა ან მთლიანობაში ინფორმაციის აღქმა.

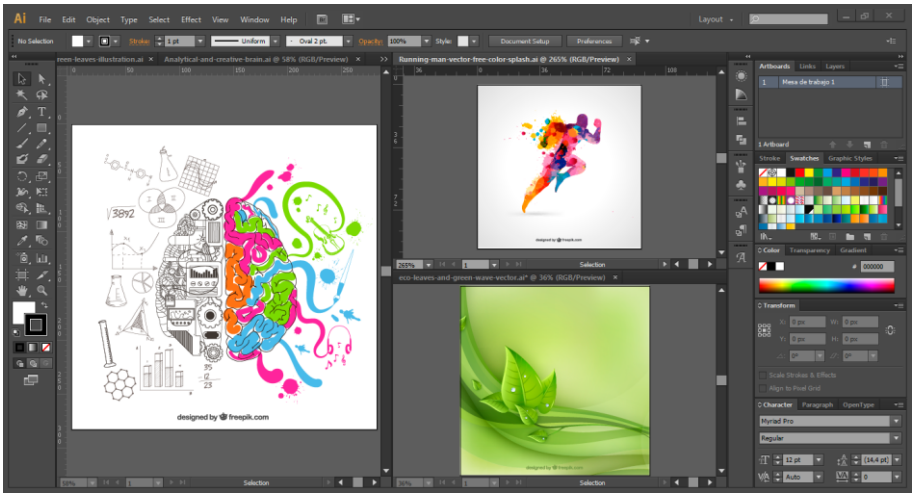
პროგრამული მენიუს გვერდით განთავსებულია ღილაკი  .

მისი დახმარებით შეგიძლიათ ბევრი გახსნილი ფანჯრის მართვა. ღილაკზე დაჭერისას იხსნება მენიუ, სადაც განთავსებული ღილაკების დახმარებით შეგიძლიათ მართოთ ფანჯრების განლაგება.

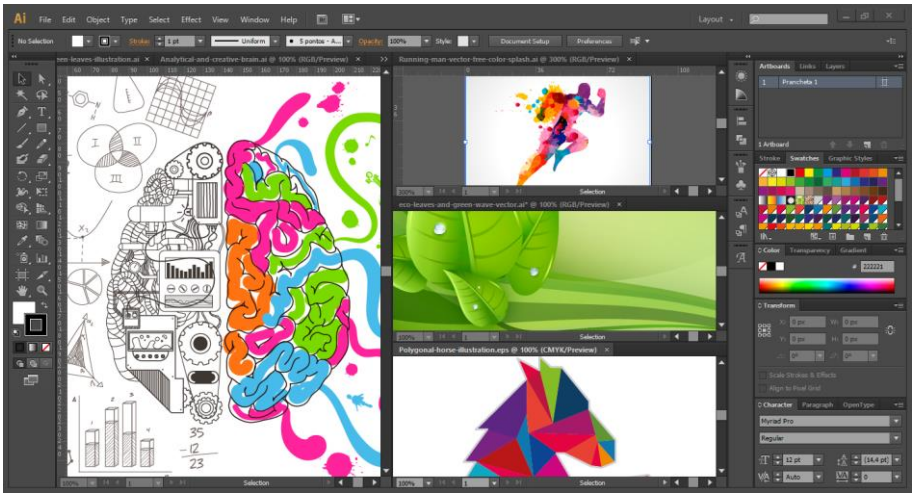
გახსნილი ფაილები შეგიძლიათ დაალაგოთ ვერტიკალურად ან ჰორიზონტალურად. სამართავი ფაილების მაქსიმალური რაოდენობაა ექვსი (განლაგების მაგალითები იხილეთ ქვემოთ).



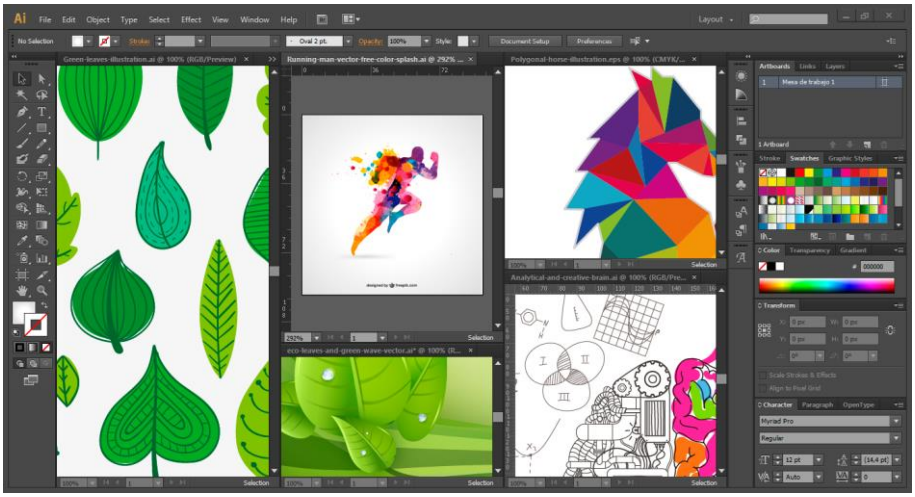
სურ. 2.1-16. 2-up Vertical - მხოლოდ ორი გამოსახულების დალაგება ვერტიკალურად.



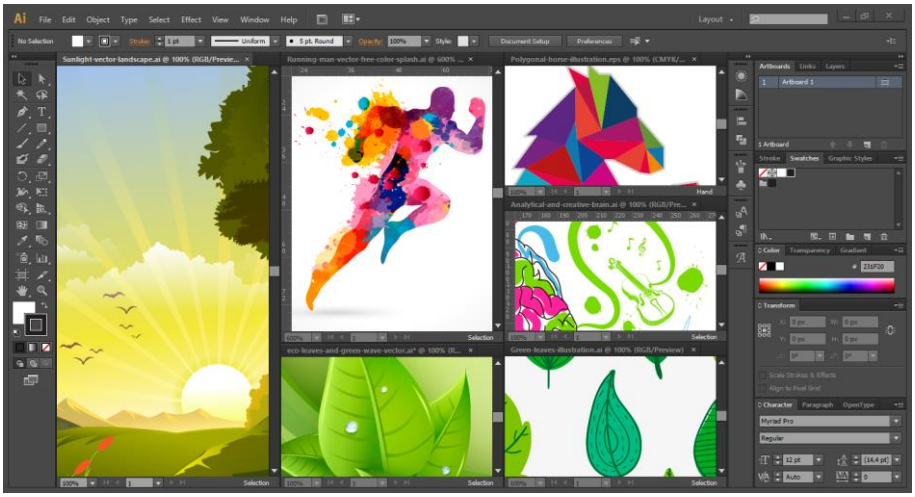
სურ. 2.1-17. 3-up Stacked - სამად დალაგებული გამოსახულებები.



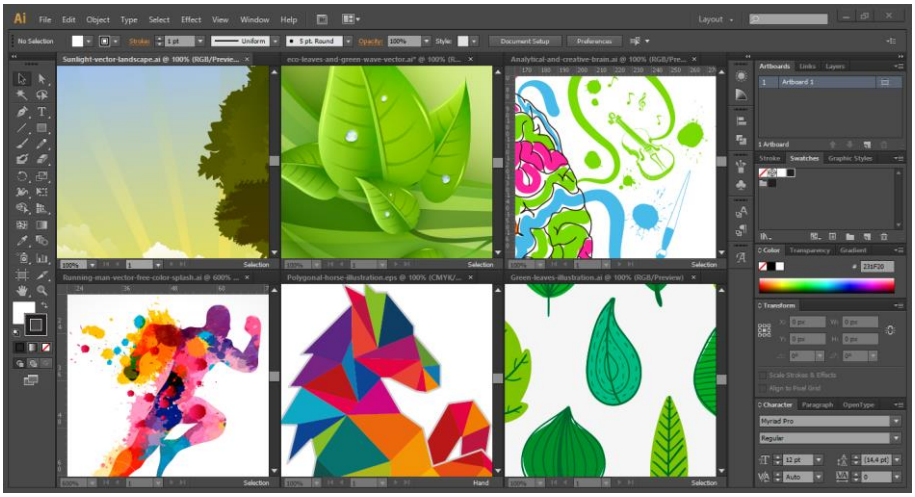
სურ. 2.1-18. 4-up - ოთხად დალაგებული გამოსახულებები.



სურ. 2.1-19. 5-up - ხუთად დალაგებული გამოსახულებები.



სურ. 2.1-20. 6-up - ექვსად დალაგებული გამოსახულებები.



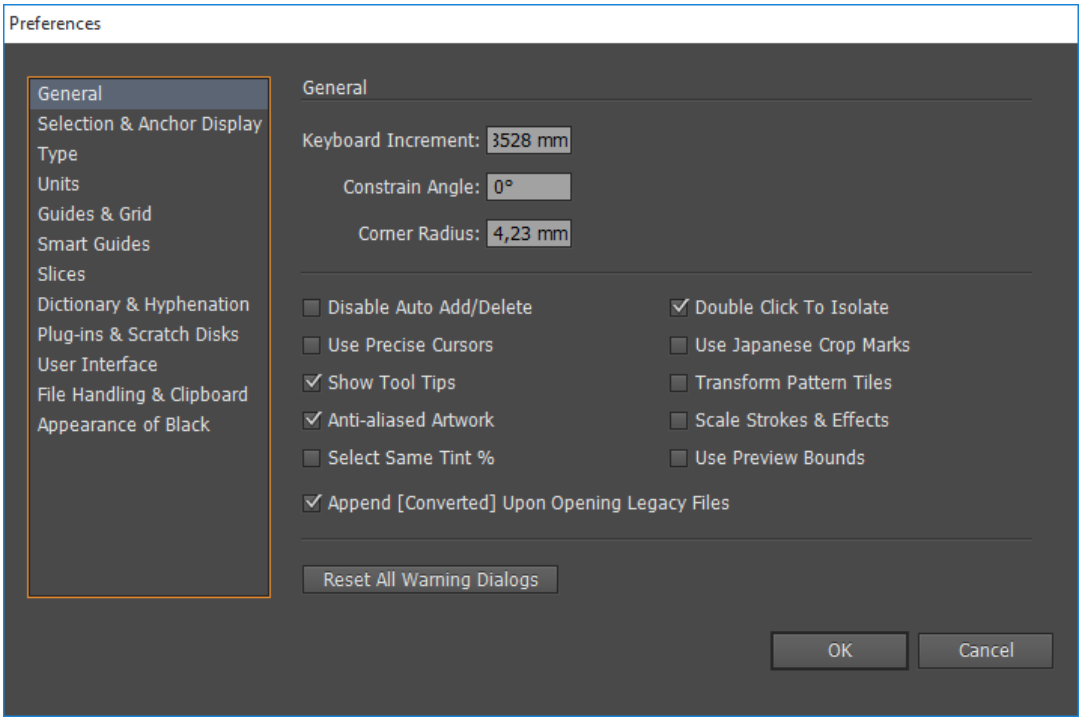
სურ. 2.1-21. Tile - გამოსახულებების მოზაიკურად დალაგება.

ფანჯრების ამ მეთოდებით დალაგებისას, ისინი მაინც მიმაგრებულები არიან ძირითად ფანჯარაზე ჩანართების სახით. გარდა ამისა, შეგვიძლია ფანჯრების სხვაგვარად დალაგებაც. ამისათვის შედით Window -> Arrange. გამოსულ ქვემენიუში შეგიძლიათ აირჩიოთ ვარიანტები: Cascade (კასკადურად), Tile (მოზაიკურად), Float in Window (ერთი ფანჯარა თავისუფალ მდგომარეობაში), Float All in Window (ყველა ფანჯარა თავისუფალ მდგომარეობაში), Consolidate All Windows (ყველა ცალკე გამოტანილ ფანჯარას ამაგრებს სანიშნეების სახით).

## გრაფიკული რედაქტორის პარამეტრების განხილვა/დაყენება

მუშაობის დაწყებამდე ან პროცესში შესაძლებელია სხვადასხვა ცვლილებები შეტანა პროგრამის პარამეტრებში. ზოგიერთი პარამეტრის შეცვლა თქვენ დაგჭირდებათ მხოლოდ ერთხელ, ხოლო ზოგიერთი შესაძლოა პერიოდულად ცვალოთ. პარამეტრების დაყენება თქვენი საქმიანობიდან და დასახული ამოცანიდან გამომდინარე შეიძლება. ილუსტრატორი, მხატვარი, დიზაინერი, ტექნიკური დიზაინერი - ყველა ირგებს პროგრამას სათავისოდ და აყენებენ მათთვის სასურველ პარამეტრებს.

პროგრამის პარამეტრები გამოიძახეთ Edit -> Preferences -> General (Ctrl +K). გამოსულ ფანჯარა შეიცავს ყველა იმ მახასიათებელს, რომელიც მოქმედებს სამუშაო პროცესზე ან პროგრამის წარმადობაზე:




სურ. 2.1-22. პროგრამა ილუსტრატორის მოწყობის ფანჯარა.

General – პროგრამის ზოგადი მახასიათებლების დაყენება;

Selection & Anchor Display - მონიშვნისა და საკვანძო წერტილების მართვა;

Type – რამდენიმე პარამეტრი, რომელიც დაგეხმარებათ ტექსტთან მუშაობაში;

Units – საზომი ერთეულების მართა;

Guides, Grid - მიმმართველების და ბადის პარამეტრების მართვა;

Smart Guides – „ჭკვიანი“ მიმმართველების პარამეტრების მართვა;

Slices - ინტერნეტისთვის გამოსახულების დაყოფის პარამეტრების მართვა;

Dictionary & Hyphenation - ტექსტისთვის ლექსიკონისა და გადატანების მართვა;

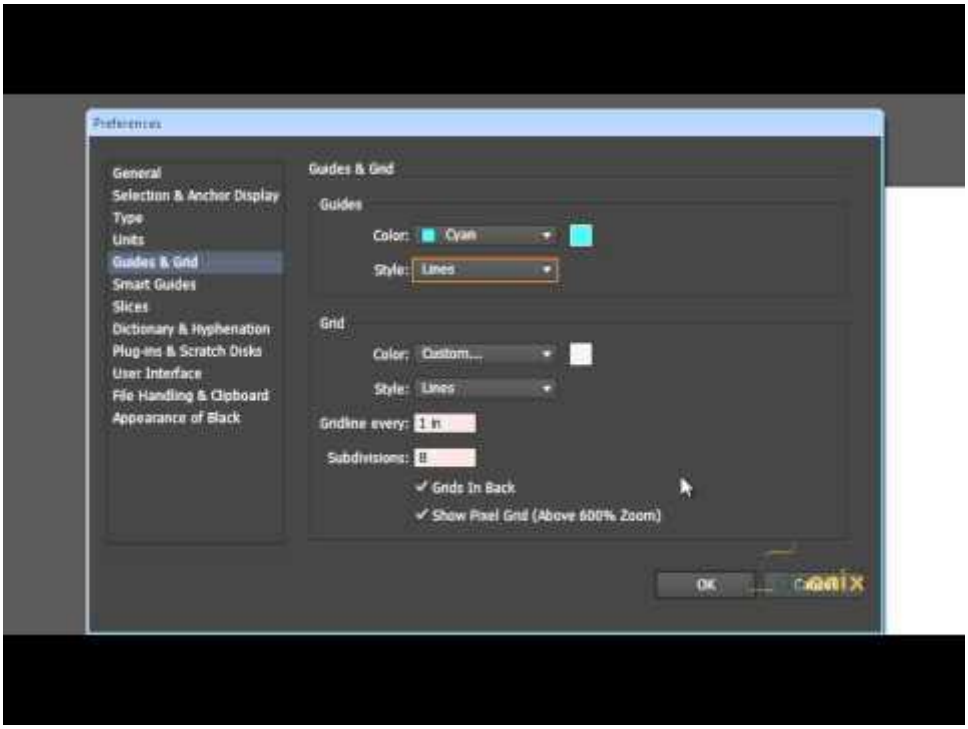
Plug-Ins & Scratch Disks – პროგრამისთვის გამოსული სხვადასხვა დამატებითი მოდულების მართა. ასევე შეგიძლიათ მიუთითოთ დისკი, რომელსაც პროგრამა გამოიყენებს ვირტუალური მეხსიერებისათვის;

User Interface – ინტერფეისის პარამეტრების დაყენება, როგორიცაა მაგალითად ფერი;

File Handling & Clipboard – ფაილის დამუშავებისა და გაცვლითი ბუფერის პარამეტრების მართვა;

Appearance of Black - შავი ფერის წარმოჩენა და მისი მახასიათებლების მართა.

**შენიშვნა:** *თუ ახალი ფაილის შექმნამდე დააყენებთ პარამეტრებს, მაგ., საზომ ერთეულებს, ის ავტომატურად გავრცელდება ყველა ახალ ფაილზე.*




ვიდეო 2.1-1. ილუსტრატორის პარამეტრების მიმოხილვა.

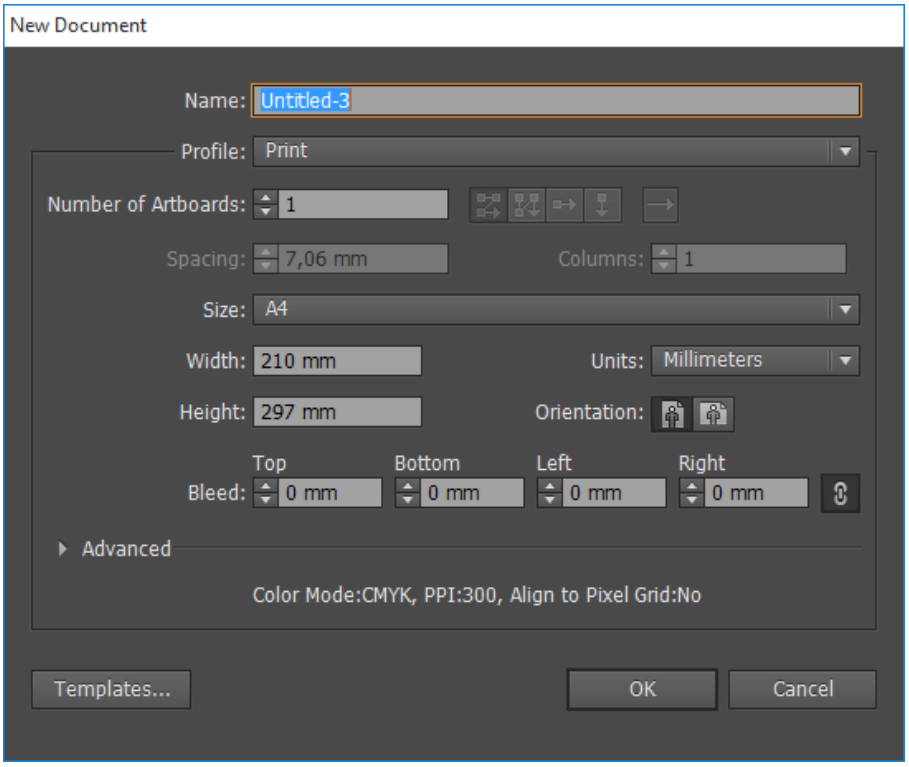
## დოკუმენტის კლასიფიკაცია და საზომი ერთეულები

მას შემდეგ, რაც მოხდება სამუშაოდ ყველანაირი პარამეტრის დაყენება: პროგრამის პარამეტრები, სამუშაო სივრცე, საზომი ერთეულები და სხვა, შეგიძლიათ დაიწყოთ გამოსახულებასთან მუშაობა.

გამოსახულებასთან მუშაობა შეგიძლიათ დაიწყოთ ორი გზით: შექმნათ ახალი გამოსახულება ან/და გახსნათ უკვე არსებული.

მივყვეთ რიგს და დავიწყოთ პირველით ანუ ახალი გამოსახულების შექმნა.

ამისათვის ვიძახებთ File -> New (Ctrl + N). გამოსულ დიალოგურ ფანჯარაში ჩვენ უნდა დავაყენოთ ახალი დოკუმენტის მახასიათებლები.




სურ. 2.1-23. ახალი ფაილის შექმნის ფანჯარა.

ამ ფანჯარაში უნდა შევიყვანოთ ის მახასიათებლები, როგორებიცაა ფაილის დასახელება, ზომა, სამუშაო დაფების რაოდენობა, ბლიდის[1] ზომა და სხვა.

Profile-ში შეგიძლიათ აირჩიოთ პროფილი, რის საფუძველზე გინდათ დაიწყოთ მუშობა. ყოველ პროფილს აქვს წინასწარ გამზადებული თავისი მახასიათებლები:

**Print** - ამ ტიპის დოკუმენტები განკუთვნილია ბეჭდვითი მედიისათვის. მოცემული აქვს სტანდარტული ქაღალდის ზომები (A3, A4 და სხვა). მისი საზომი ერთეულები მილიმეტრებია, ხოლო წერტილების რაოდენობა 300 px/inch.

**Web** - აქ მოცემულია ფორმატები, რომლებიც ინტერნეტისთვისაა განკუთვნილი. საზომი ერთეულები პიქსელებია. გარჩევადობა 72 px/inch.

**Devices** - გამოიყენეთ იმ შემთხვევაში, როდესაც პროექტს ამზადებთ სიაში მოცემული რომელიმე მოწყობილობისათვის.

**Film & Video** - დოკუმენტის ნაირსახეობა, რომლის პარამეტრები ორიენტირებულია ვიდეოპროდუქციაზე. მენიუ Size-ში თქვენ ნახავთ უამრავ ვიდეო ფორმატს, რომელიც შეგიძლიათ აირჩიოთ თქვენი ამოცანიდან გამომდინარე. გარჩევადობა 72 px/inch.

**Basic RGB** - სტანდარტული RGB დოკუმენტი.

**Flash Builder** - აირჩიეთ ეს პროფილი თუ თქვენ ფაილებს ამზადებთ Flash პროგრამისათის.

**Custom** - მომხმარებელზე მორგებული. ავტომატურად ააქტიურდება, როდესაც ზემოთ ჩამოთვლილ პროფილში შეიტანთ ნებისმიერ ცვლილებას.

რა ძირითად მონაცემებს ვუთითებთ, როდესაც გვინდა შევქმნათ ახალი გამოსახულება?

**Width** - გამოსახულების სიგანე

**Height** - გამოსახულების სიმაღლე

**Units** - საზომი ერთეულები.

**Orientation** - სამუშაო დაფის განლაგება - ვერტიკალურად ან ჰორიზონტალურად.

**Bleed** - ჩამოსაჭრელი ზონის ზომა.

**Advanced** - აქ უთითებთ დამატებით მახასიათებლებს, როგორიცაა ფერების რეჟიმი, გარჩევადობის ხარისხი და სხვა.

**Templates** - ამ ღილაკის დახმარებით შეგიძლიათ გამოიძახოთ დოკუმენტის მზა შაბლონები[2].

[1] ბლიდი (bleed) - არე დოკუმენტის გარშემო. როგორც წესი 3-5 მმ-ის ზომის. გამოიყენება სრულად შევსებული ფურცლის ბეჭდვისათვის. ამ ზონაში მოხვედრილი ფონის თუ სურათის ნაწილი შემდეგში სტამბის მიერ ჩამოიჭრება და თქვენ მიიღებთ სუფთა გამოსახულებას. მინიმუმადე დაყავს ჩამოჭრის ცდომილებები.

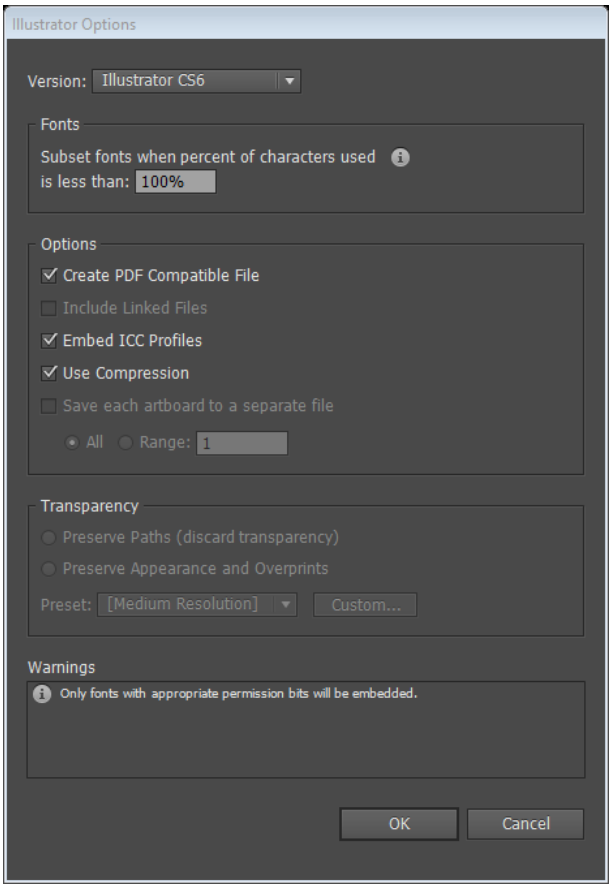
[2] თუ ხშირად გიწევთ მსგავსი ზომების ფაილებთან მუშაობა, შეინახეთ ისინი, როგორც შაბლონები და შემდეგში გამოიძახეთ ამ ღილაკის დახმარებით. აღარ მოგიწევთ ყოველთვის თავიდან მონაცემების ხელახლა დაყენება.

## ფაილების შენახვა და ექსპორტი.

ილუსტრატორს, ისევე როგორც პროგრამების უმეტესობას შეუძლია სხვადასხვა ფორმატში ფაილების შენახვა.

### *შენახვა (Save)*

ჩვეულებრივ, როდესაც პირველად ახდენთ შენახვას, პროგრამა გთავაზობთ შეინახოთ *.AI ფორმატში. ეს ილუსტრატორის საკუთარი ფორმატია.




სურ. 2.1-24. ილუსტრატორის ფაილის მახასიათებლების დაყენება.

ილუსტრატორის ფაილის შენახვისას გამოსულ ფანჯარაში სხვადასხვა მახასიათებლები შეგიძლიათ მიუთითოთ. მაგ., თუ მომავალში ეს ფაილი უფრო ძველ ვერსიაში უნდა გაიხსნას, მაშინ Version ჩამოსაშლელ მენიუში მიუთითეთ ვერსია, რომლისთვისაც გინდათ ეს ფაილი შეინახოთ[1].

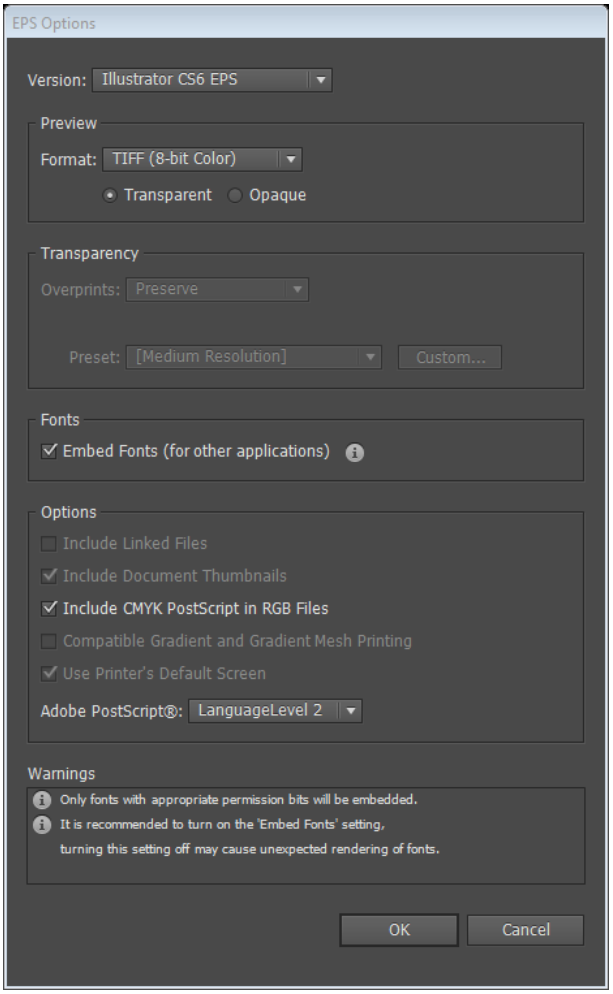
თუ ფაილს შემდეგში შეიტანთ ინდიზაინში ან ფოტოშოპში და დაყენებული ფერთა პროფილის შესაბამისად რომ მოხდეს ფაილის გახსნა აუცილებლად ჩაურთეთ Embed ICC Profiles (ფერთა პროფილის ჩაშენება). ამ ინფორმაციას ამოიკითხავენ პროგრამები და ფაილის ფერები სწორად აღიქმება სხვა პროგრამების მიერ.

### *შენახვა როგორც... (Save As...)*

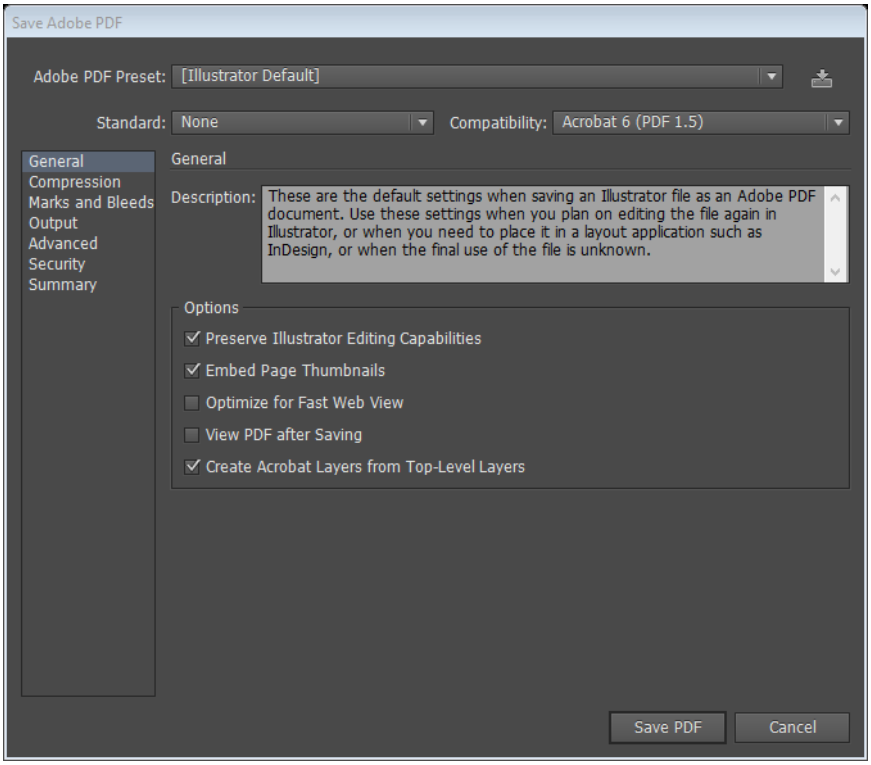
გამოიყენეთ ეს ბრძანება, როდესაც გინდათ ფაილი შეინახოთ სხვა ფორმატში ან სხვა სახელით. აქედან შეგიძლიათ აირჩიოთ *.EPS, *.PDF და სხვა ფორმატები.

ისევე, როგორც ილუსტრატორის ფორმატში შენახვისას, აქაც თქვენ წინაშე გამოჩნდება ფანჯარა, რომელიც ფაილისთვის სხვადასხვა სახის პარამეტრების დაყენებას შემოგთავაზებთ.

[1] პროგრამის ძველი ვერსიები ვერ ან არაკორექტულად ხსნიან ახალ ვერსიაში მომზადებულ ფაილებს.




სურ. 2.1-25. EPS ფორმატის ფაილისთვის მახასიათებლების დაყენება. ისევე, როგორც ილუსტრატორის შემთხვევაში, აქაც შეგიძლიათ მიუთითოთ პროგრამის ვერსია. მაგრამ AI ფაილისგან განსხვავებით, EPS-ში თქვენ ვერ მოახდენთ ფერების პროფილის ჩაშენებას.




სურ. 2.1-26. ფაილის შენახვა PDF ფორმატში. მისი საშუალებით თქვენ შეგიძლიათ მოამზადოთ როგორც დაბალი ხარისხის (ინტერნეტისთვის, ელფოსტით გასაგზავნად), ისე მაღალი ხარისხის (დასაბეჭდად, სტამბისთვის) PDF ფაილები.

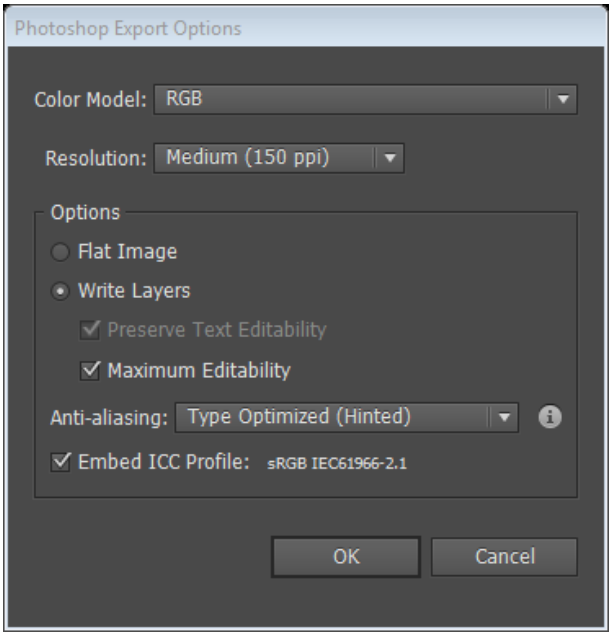
### *ინტერნეტისთვის შენახვა (Save for Web)*

გამოიყენეთ იმ შემთხვევაში, თუ თქვენი ნამუშევარი სამომავლოდ უნდა განთავსდეს ინტერნეტში (ფბ-ზე, ტვიტერზე, საიტზე და სხვა).


სურ. 2.1-27. შეგიძლიათ შეინახოთ ინტერნეტისთვის თავსებად ფორმატში - *.JPEG ან *.PNG. აქვე არის ხედვის ფანჯარა, სადაც წინასწარ შეგიძლიათ ნახოთ, როგორი გამოვა გამოსახულება (ხარისხი, ფერები და სხვა). შეგიძლიათ შენახვის პროცესში მოახდინოთ გამოსახულების მასშტაბირება - შეამციროთ ან გაადიდოთ - მოთხოვნის შესაბამისად.

### *ექსპორტი (Export)*

პროგრამიდან ასევე შესაძლებელია ფაილების გატანა სხვა ფორმატში, როგორებიცაა მაგ. *.TIFF ან *.PSD ფორმატები. მათთვისაც შეგიძლიათ სხვადასხვა მახასიათებლების დაყენება.




სურ. 2.1-28. PSD ფორმატში ექსპორტირება. ამ ფანჯარაში შეგიძლიათ დააყენოთ ფერთა მოდელი, გარჩევადობა, მოხდეს თუ არა ფენების გაერთიანება და ფერების პროფილის ჩაშენება ფაილში.

## სავარჯიშო

1. რა განსხვავებაა ვექტორულ ობიექტსა და რასტრულ გამოსახულებას შორის?
2. ჩამოთვალეთ ვექტორული და რასტრული ფაილის ფორმატები. რას წარმოადგენს ილუსტრატორის ფაილის ფორმატი?
3. ჩამოთვალეთ ფერთა მოდელები.
4. მოახდინეთ ფერთა სინქრონიზაცია პროგრამებს შორის.
5. მოაწყეთ თქვენთვის სასურველი სამუშაო სივრცე და შეინახეთ.
6. რა სახის ფაილების შექმნაა შესაძლებელი ილუსტრატორის დახმარებით.
7. რა ფორმატშია შესაძლებელი ფაილების შენახვა/ექსპორტირება ილუსტრატორიდან.

## 2.2. ვექტორული ობიექტის შექმნა შესაბამისი თვისებების გათვალისწინებით

პარაგრაფის შესაბამისი თემატიკა

- მარტივი გეომეტრიული ფიგურების შექმნა. სხვადასხვა მეთოდების გამოყენება
- ვექტორული ობიექტების შიგთავსისა და კონტურის თვისებები
- ფუნჯისა და სხვა სახატავი ინსტრუმენტების გამოყენება
- სიმბოლოების ბიბლიოთეკის გამოყენება
- ტექსტი და მასთან მუშაობა: რედაქტირება, ფორმატირება და სხვა
- გრაფიკული გამოსახულების შემადგენელ ნაწილებად დაშლა

### გეომეტრიული და სხვა სახის ფიგურები და ფორმები

ნებისმიერი რთული გამოსახულება შესაძლებელია დაყვანილი იქნეს მარტივ გეომეტრიულ ფიგურებამდე. შესაბამისად, ნებისმიერი რთული გამოსახულების აგება მარტივი გეომეტრიული ფიგურებისგან შეიძლება.

ილუსტრატორში ბევრი ინსტრუმენტია, რომელთა დახმარებით გეომეტრიული და თავისუფალი ფორმის ფიგურების აგება, რედაქტირება და ტრანსფორმირება შეიძლება. ამისათვის გამოიყენება ხატვის ინსტრუმენტები, რომლებიც ინსტრუმენტთა პანელზეა განთავსებული (სურ. 2.1-14, B).

აქ შეგვიძლია გამოვყოთ 3 ძირითადი ჯგუფი:

**გეომეტრიულ ფიგურები:** (Rectangle), მომრგვალებული მართკუთხედი (Rounded Rectangle), ელიფსი (Ellipse), მრავალკუთხედი (Polygonal), ვარსკვლავი (Star) და ათინათი (Flare). (სურ. 2.2-2)

**ხაზები:** ხაზი (Line), რკალი (Arc), სპირალი (Spiral), მართკუთხა ბადე (Rectangular Grid) და პოლარული ბადე (Polar Grid). (სურ. 2.2-4)

**თავისუფალი ფორმები:** ფუნჯი (Brush), ფანქარი (Pencil), წვეთოვანი ფუნჯი (Blob Brush) და კალამი (Pen). (სურ. 2.2-5)

#### *გეომეტრიული ფიგურები*

აირჩიეთ შესაბამისი ინსტრუმენტი და დახაზეთ. დახაზვისას. ობიექტის ზომები ფაილში დაყენებული საზომი ერთეულების შესაბამისად განისაზღვრება (მმ, პქს).

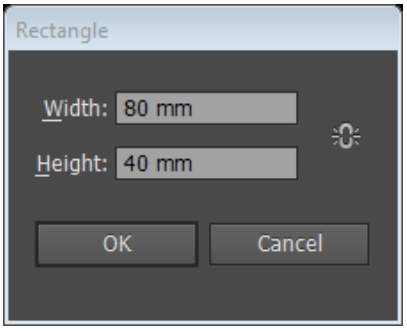
ფიგურების დახაზვისას გამოიყენეთ Shift და Alt კლავიშები:

Shift-ის დახმარებით დახაზვისას თქვენ მიიღებთ სიმეტრიულ ფიგურას, მაგალითად კვადრატს;

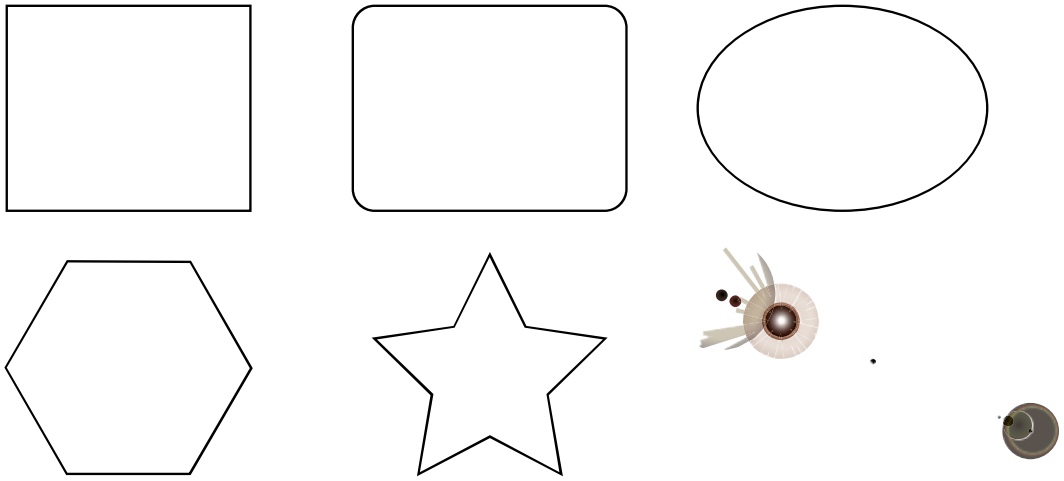
Alt-ის დახმარებით დახაზვისას ობიექტი იხაზება ცენტრიდან;

Shift-ისა და Alt-ის ერთდროულად გამოყენებისას, ცენტრიდან დაიხაზება სიმეტრიული ფიგურა.

იმ შემთხვევაში თუ წინასწარ იცით რა ზომის ან როგორი პროპორციების ფიგურა უნდა შექმნათ, გამოიყენეთ შემდეგი მეთოდი - ინსტრუმენტის არჩევის შემდეგ კურსორი დააწექით ნებისმიერ ადგილას სამუშაო დაფაზე (შეიძლება მის გარეთაც). გამოსულ ფანჯარაში მიუთითეთ სასურველი ზომები.



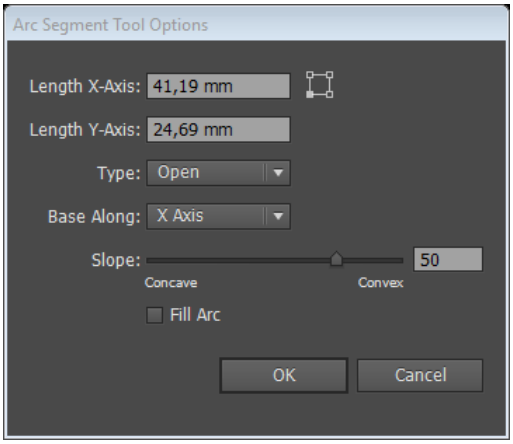
სურ. 2.2-1. მართკუთხედის შექმნა მითითებული ზომებით.



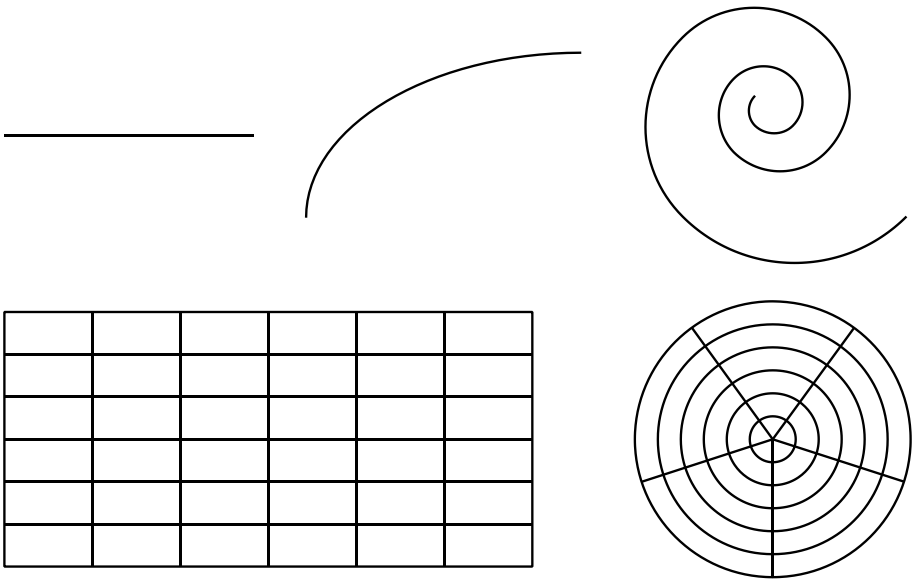
სურ. 2.2-2. გეომეტრიული ფიგურების ჯგუფი.

### *ხაზები*

ხაზების გამოყენებისას იგივე მეთოდი შეგიძლიათ გამოიყენოთ, როგორც გეომეტრიული ფიგურების შექმნისას: შეგიძლიათ ხაზი გაავლოთ ან თავისუფალ ადგილზე დაჭერის შემდეგ მიუთითოთ სასურველი მახასიათებლები:



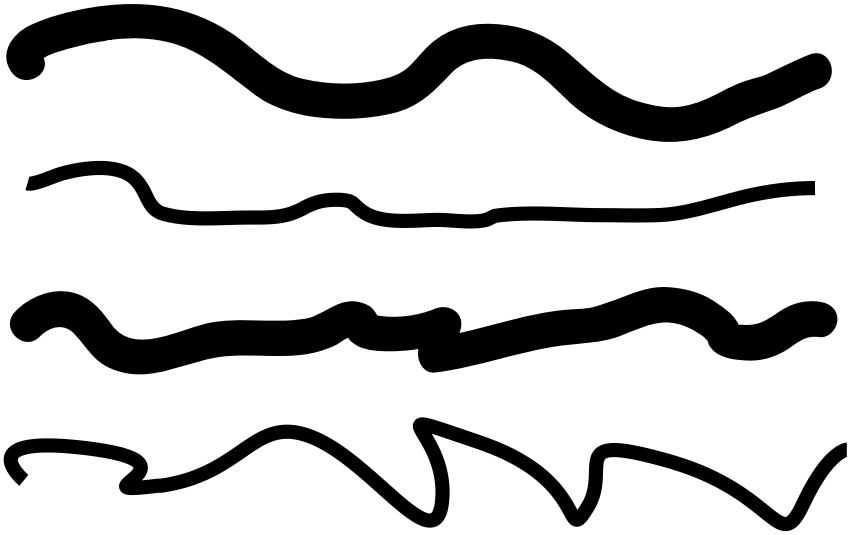

სურ. 2.2-3. ამ ფანჯარაში შეგიძლიათ დააყენოთ შესაქმნელი რკალის მახასიათებლები.



სურ. 2.2-4. ხაზების ჯგუფი

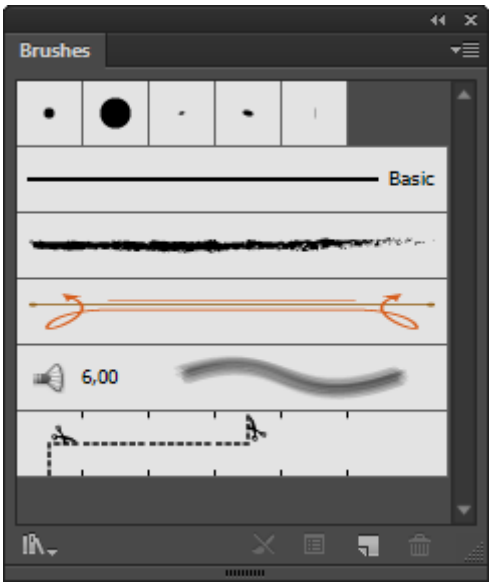
### თავისუფალი ფორმები

*ფუნჯი*



სურ. 2.2-5. თავისუფალი ფორმების ჯგუფი

ფუნჯის დახმარებით შესაძლებელია თავისუფალი ფორმების შექმნა. იქმნება ხელგაკრული ხაზის ეფექტი. ნახატი წარმოდგენილი იქნება კონტურის სახით. ფუნჯისთვის შეგიძლიათ დააყენოთ კონტურის სისქე და ფუნჯის ტიპი:

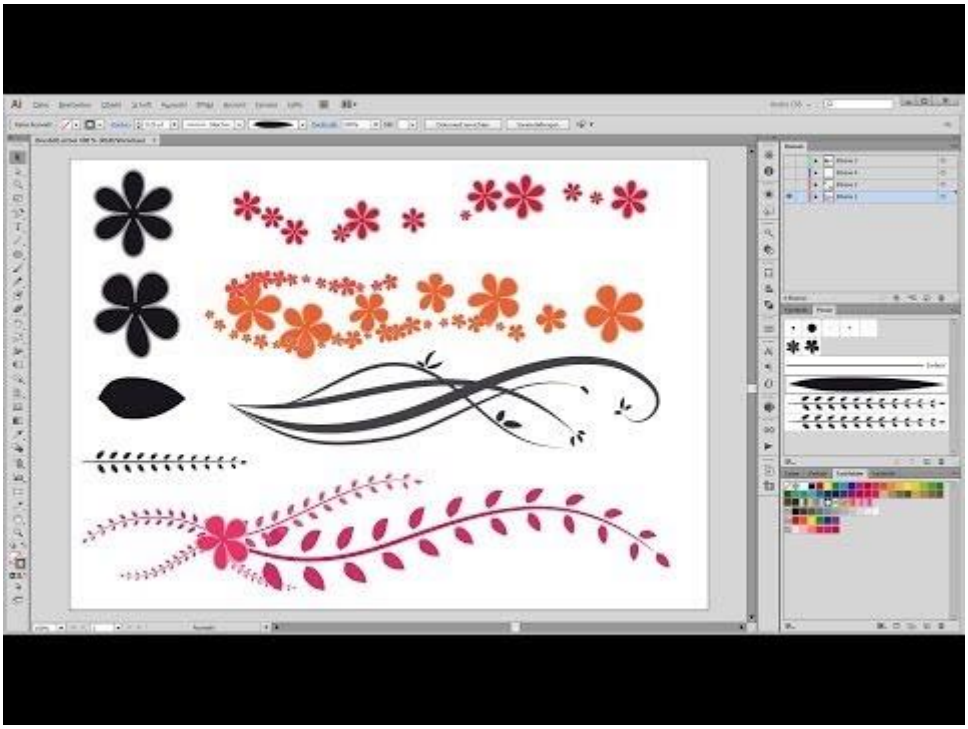



სურ. 2.2-6. ფუნჯის ტიპები.

სურათზე (სურ. 2.2-6) მოცემულია პროგრამის ფუნჯების სტანდარტული ნაკრები. გარდა ამისა თქვენ შეგიძლიათ შექმნათ საკუთარი ფუნჯი ან გამოიძახოთ ფუნჯების დამატებითი ნაკრები. ამისათვის შეგიძლიათ ისარგებლოთ ფუნჯების პანელზე მარცხენა ქვედა კუთხეში მოცემული ღილაკით ან Window -> Brush Libraries. ქვემენიუში მოცემულია ფუნჯების ნაირსახეობები, რომლებიც ჯგუფებადაა დაყოფილი.

დამატებითი ბიბლიოთეკიდან არჩეული ფუნჯი ავტომატურად ემატება ფუნჯების პანელზე.

ინსტრუმენტი **წვეთოვანი ფუნჯი** (Blob Brush) ჩვეულებრივი ფუნჯისგან განსხვავებით ხატავს არა კონტურებით, არამედ შევსებული ფორმებით.



ვიდეო 2.2-1. საკუთარი ფუნჯის შექმნა

**შენიშვნა:** *ინსტრუმენტთა პანელზე ფუნჯის ნიშნულზე ორჯერ სწრაფად დაჭერით შეგიძლიათ გამოიძახოთ ფუნჯის პარამეტრების მართვის ფანჯარა.*

### *ფანქარი*

ფანქარი ახდენს ფანქრით მუშაობის იმიტაციას. ქმნის უწყვეტ თავისუფალ ფორმებს. შეგიძლიათ აირჩიოთ ხაზის სისქე, ფერი და სწრაფად დახატოთ სასურველი ფორმა. ფანქრის გამოყენების შემდეგ შექმნილ კონტურს შეგიძლიათ მიუთითოთ ასევე სხვადა ფორმები და ხაზების ნაირსახეობები და ფუნჯის ტიპები.

ფანქართან ერთად გაერთიანებულია კიდევ ორი ინსტრუმენტი, რომელთა დახმარებით შექმნილი მრუდეების ან ფორმების რედაქტირება შეგიძლიათ.

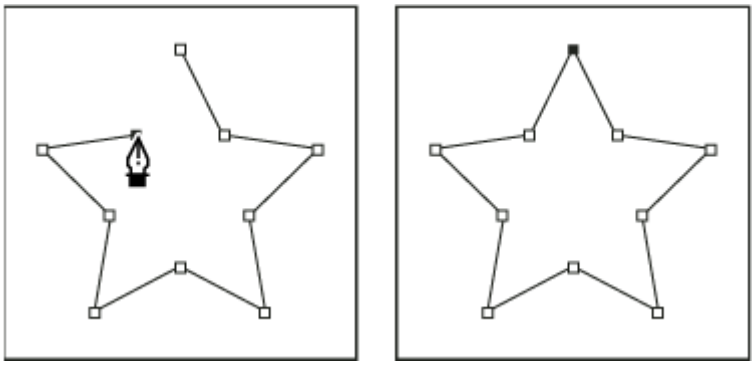
სურ. 2.2-7. მარცხნიდან მარჯვნივ: ფანქარი, სიგლუვე - მისი დახმარებით ხდება სიგლუვის მომატება მრუდის კუთხოვან ადგილებში, საშლელი - საკვანძო წერტილების წაშლა (შესაბამისად იშლება მრუდის ნაწილი).

**შენიშვნა:** *ინსტრუმენტთა პანელზე ფანქრის ნიშნულზე ორჯერ სწრაფად დაჭერით შეგიძლიათ გამოიძახოთ ფანქრის პარამეტრების მართვის ფანჯარა.*

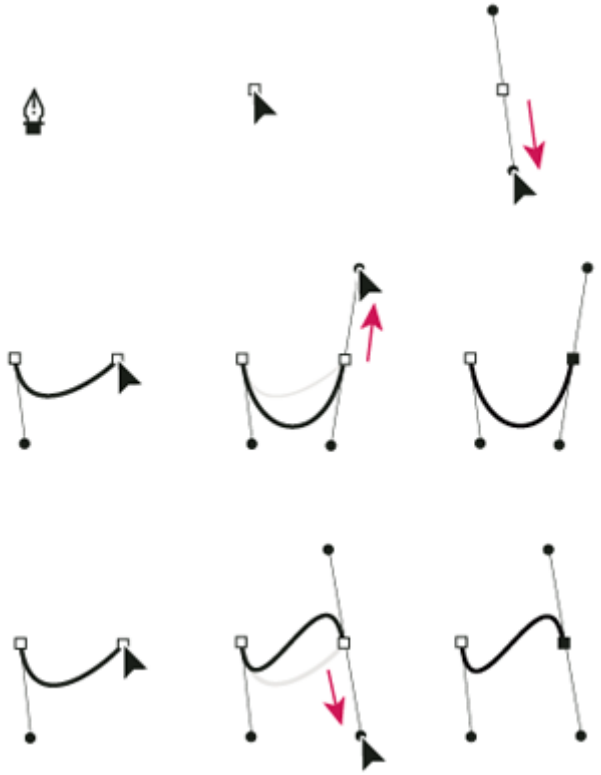
**შენიშვნა:** *ფუნჯით ხატვისას დააწექით Alt კლავიშს და ფუნჯი დროებით გადაიქცევა ფანქრად.*

### *კალამი*

კალმის დახმარებით იქმნება თავისუფალი ფორმები. ფორმების შედგენა ხდება წერტილ-წერტილ და უკვე შექმნისას თქვენ შეგიძლიათ ცვალოთ მისი ფორმა და მოხაზულობა. კალმით ხატვისას თქვენ შეიძლება შექმნათ როგორც სწორკუთხოვანი, ისე მომრგვალებული ფორმები.



სურ. 2.2-8. კალმის გამოყენებით ფორმის შექმნა. შექმნილი ფორმა შეიძლება იყოს როგორც გახსნილი, ისე ჩაკეტილი. ჩაკეტილი ფორმა რომ მიიღოთ, ბოლო წერტილი შეაერთეთ პირველ წერტილთან.



სურ. 2.2-9. მრუდი წირების შექმნა კალმით ხატვისას.

კალამთან ერთად მოთავსებული სხვა ინსტრუმენტების დახმარებით შეგიძლიათ საკვანძო წერტილების რედაქტირება.

სურ. 2.2-10. მარცხნიდან მარჯვნივ: კალამი, საკვანძო წერტილის დამატება, საკვანძო წერტილის წაშლა, საკვანძო წერტილის რედაქტირება (მასზე მოქმედება რეალურად იწვევს არსებული მრუდის ფორმის შეცვლას).

## ვექტორული ობიექტის თვისებები

ნებისმიერ ვექტორულ ობიექტს გააჩნია თვისებები. თვისებებში იგულისხმება არამხოლოდ მისი ზომები და მდებარეობა სამუშაო დაფაზე. თვისებებში შედის ასევე ობიექტის ფონისა და კონტურის ფერი, კონტურის სისქე, ხაზის ტიპი და სხვა.

ამ მხრივ ობიექტები იყოფიან ორ ნაწილად: ობიექტები, რომლებსაც აქვთ ფონის და კონტური; ობიექტები, რომლებსაც მხოლოდ კონტური აქვთ.

ობიექტის მონიშვნის შემდეგ, ფორმატირების ზოლზე გამოტანილი პარამეტრების დახმარებით შეგიძლიათ ცვალოთ ობიექტის თვისებები.


სურ. 2.2-11. ვექტორული ობიექტის თვისებების ნაწილი გამოტანილია ფორმატირების ზოლზე. აქედან თქვენ შეგიძლიათ მართოთ მისი ფონის ფერი, ხაზის სისქე და ბევრი სხვა მახასიათებელი. მაგ., ამ შემთხვევაში ჩანს, რომ მონიშნულია ობიექტი, რომელსაც აქვს თეთრი ფონი, შავი კონტური, კონტურის სისქე 1pt, ხაზის ტიპი ჩვეულებრივი, გამჭვირვალობა 100% და ა.შ.

გარდა ამისა, ინსტრუმენტთა პანელის ქვემოთ მოთავსებულია ფონისა და კონტურის ფერების აღმნიშვნელი. ობიექტის მონიშვნის შემდეგ ის მიიღებს შესაბამის ფერებს ფონისა და კონტურისათვის.

სურ. 2.2-12. წინა პლანზე რომელი თვისებაც იქნება წამოწეული, ფერის არჩევის შემთხვევაში, ის დაიფერება.

პატარა ისრის დახმარებით თქვენ შეგიძლიათ უცვალოთ ადგილები ფონსა და კონტურს ანუ გახადოთ მიმდინარე. მოცემულ სურათზე (
სურ. 2.2-12) აქტიური არის ობიექტის ფონი.

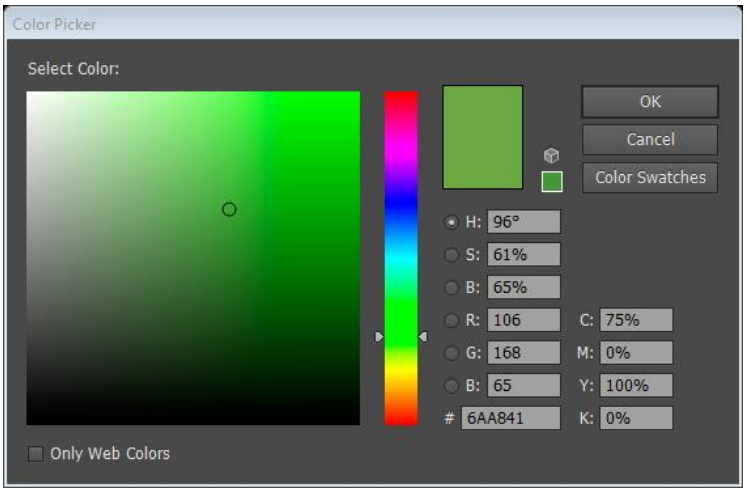
**შენიშვნა:** *კლავიატურიდან სწრაფად რომ უცვალოთ ადგილები ფონსა და კონტურს ანუ გახადოთ აქტიური, გამოიყენეთ კლავიში (X); ფონისა და კონტურის ფერებს რომ შეუცვალოთ ადგილი გამოიყენეთ (Shift + X). გამოიყენეთ კლავიში (D), რომ აღადგინოთ საწყისი ფერები (თეთრი ფონი და შავი კონტური).*

### *ფერი*

ფონისა და კონტურისთვის ფერების არჩევა რამდენიმე გზით შეიძლება. ობიექტი წინასწარ უნდა მონიშნოთ ან თუ არ მონიშნავთ და ისე აირჩევთ ფერებს, შემდეგში შექმნილი ობიექტი შესაბამისად იქნება დაფერილი:

ფორმატირების ზოლის დახმარებით (სურ. 2.2-11);

ინსტრუმენტთა პანელზე ფონისა ან კონტურის (
სურ. 2.2-12) შესაბამის ნიშნულზე ოჯერ სწრაფად დაჭერით გაიხსნება ფერთა პალიტრა, საიდანაც აირჩევთ ფერს;




სურ. 2.2-13. ფერი შეგიძლიათ აირჩიოთ სხვადასხვა ფერთა მოდელების გათვალისწინებით - HSB, RGB ან CMYK.

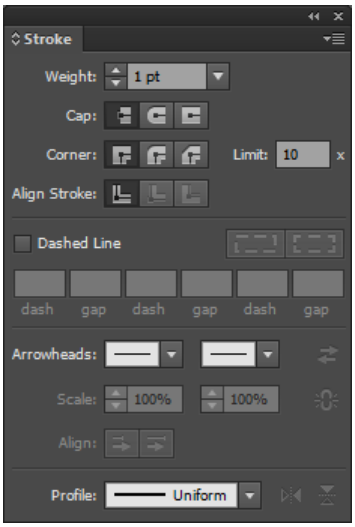
ფერთა პანელის დახმარებით: Window -> Color;

ფერთა ნიმუშების პანელის დახმარებით: Window -> Swatches;

### *კონტური*

ობიექტის კონტურები სხვადასხვა სახით შეიძლება იყოს წარმოდგენილი.

ამისათვის შეგვიძლია შესაბამისი სამუშაო პანელი გამოვიძახოთ Window -> Stroke

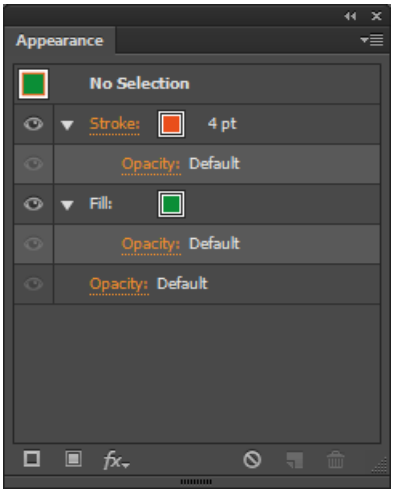



სურ. 2.2-14. პანელის დახმარებით შეგვიძლია სხვადასხვა მახასიათებელი დავაყენოთ კონტურისთვის.

შეგვიძლია ვცვალოთ ხაზის სისქე, კუთხოვანი ნაწილების მომრგვალება, სრული ხაზი ვაქციოთ წყვეტილად, საჭიროების შემთხვევაში ხაზს გავუკეთოთ ისრები ერთი ან ორივე მხრიდან (იმ შემთხვევაში თუ მრუდი შეკრული არ არის).

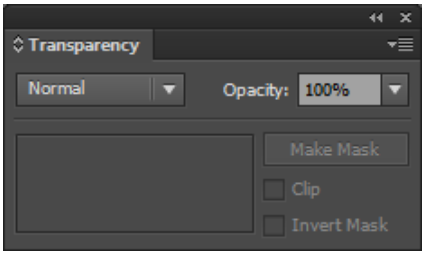
### *წარმოდგენის პანელი*

წარმოდგენის პანელიდან შეგვიძლია ობიექტის არამხოლოდ ფონისა და კონტურის ფერი ვმართოთ, არამედ მისი სხვა თვისებებიც. პანელის გამოძახება ხდება Window -> Appearance.




სურ. 2.2-15. ამ სამუშაო პანელის დახმარებით შეგიძლიათ არამხოლოდ ფერების შერჩევა კონტურისა (Stroke) და ფონისთვის (Fill), არამედ შეგიძლიათ დააყენოთ გამჭვირვალობა (Opacity), შერევის (Blend) რეჟიმები, სხვადასხვა ეფექტები და ა.შ.

გამჭირვალობის (Opacity) ფუნქციაზე დაჭერით იხსნება დამატებითი პანელი, საიდანაც შეგვიძლია გამჭვირვალობისა და შერევის რეჟიმების დაყენება ობიექტისთვის. ამ პანელის გამოძახება შესაძლებელია მენიუ Window -> Transparency:




სურ. 2.2-16. ამ პანელის დახმარებით შეგვიძლია ვმართოთ ობიექტის გამჭვირვალობა და შერევის რეჟიმები.

პანელზე ჩანს (სურ. 2.2-15), რომ ჩვენ შეგვიძლია გამჭვირვალობა და შერევის რეჟიმები დავაყენოთ როგორც მთლიანად ობიექტისთვის, ისე ცალკე კონტურისთვის და ცალკე ფონისთვის.

### *შერევის რეჟიმები*

**Normal** - ნორმალურ რეჟიმში ზედა და ქვედა ფენის პიქსელების შერევა არ ხდება. ზედა ფენის პიქსელები ავტომატურად ჩაანაცვლებენ ქვედა ფენის პიქსელებს, თუ ზედა ფენის გამჭვირვალობა 100%-ის ტოლია. პიქსელების ფერების შერევის ეფექტი, ფენის გამჭვირვალობის შემცირებით მიიღწევა, ისიც იმ შემთხვევაში, თუ ფენებზე სხვადასხვა სახის გამოსახულება არის მოთავსებული. ავტომატურ მდგომარეობაში, ყველა ფენა Normal რეჟიმში იმყოფება.

**დამუქების რეჟიმები:**

**Darken** - მუქი ფერით ჩანაცვლება. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო ღია ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელი ფერი, ხოლო პიქსელი, რომელიც უფრო მუქი ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Multiply** - გამრავლება. ხდება ზედა და ქვედა ფენების ფერების გადამრავლება და შედეგად მიღებული გამოსახულების ფერი უფრო მუქია, ვიდრე ცალკე აღებული თითოეული ფენის გამოსახულების ფერი. პროცესში მონაწილეობს ყველა ფერის პიქსელი, თეთრი ფერის გარდა.

**Color Burn** - ფუძის დამუქება. ხდება მუქი ფერის პიქსელების დამუქება, ხოლო ღია ფერის პიქსელები უცვლელი რჩება.

**გაღიავების რეჟიმები:**

**Lighten** - ღია ფერით ჩანაცვლება. Darken რეჟიმის საპირისპირო რეჟიმია. ზედა და ქვედა ფენის პიქსელების ფერების შედარების შემდეგ, საბაზო ფენის იმ პიქსელების ფერის ჩანაცვლება ხდება ზედა ფენის პიქსელებით, რომლებიც უფრო მუქი ფერისაა, ვიდრე მის ზემოთ მდებარე პიქსელის ფერი, ხოლო პიქსელი, რომელიც უფრო ღია ფერისაა ვიდრე ზედა ფენის პიქსელი, უცვლელი რჩება.

**Screen** - განათება. Multiply რეჟიმის საპირისპირო რეჟიმია. ხდება გამოსახულების ყველა ფერის პიქსელის გაღიავება, შავი ფერის გარდა.

**Color Dodge** - ფუძის განათება. Color Burn რეჟიმის საპირისპირო რეჟიმია. აღიავებს გამოსახულების ღია ფერის პიქსელებს და უცვლეს ტოვებს მუქი ფერის პიქსელებს.

**კონტრასტის რეჟიმები:**

**Overlay** - გადაფარვა. ეს არის Multiply და Screen რჟიმების კომბინირებული რეჟიმი. ხდება მუქი პიქსელების დამუქება და ღია პიქსელების გაღიავება. ეფექტი არ მოქმედებს შავ და თეთრ ფერებზე. ამ რეჟიმში 50%-იანი ნაცრისფერი გაუმჭვირვალე ხდება.

**Soft Light** - რბილი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**Hard Light** - ძლიერი განათება. იგივენაირად მოქმედებს, როგორც Overlay რეჟიმი, მაგრამ ეფექტი უფრო ძლიერია.

**შედარების რეჟიმები**:

**Difference** - განსხვავება. ხდება პიქსელების სიმკვეთრის მაჩვენებლების შედარება, დიდი მაჩვენებლებიდან პატერა მაჩვენებლების გამოკლება და სხვაობის შებრუნება. ერთნაირი მაჩვენებლების პიქსელები შავი ფერით აისახება.

**Exclusion** - გამორიცხვა. ისევე მოქმედებს, როგორც Difference რეჟიმი, მაგრამ ეფექტი უფრო სუსტია.

**გამოსახულების კომპონენტების რეჟიმები:**

**Hue** - ტონი. მიიღება გამოსახულება ზედა ფენის ტონით და ქვედა ფენის გაჯერებულობით და სიმკვეთრით.

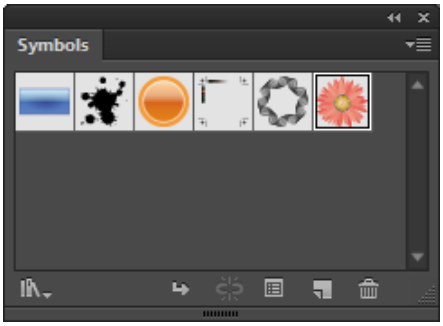
**Saturation** - გაჯერებულობა. მიიღება გამოსახულება ზედა ფენის გაჯერებულობით და ქვედა ფენის ტონით და სიმკვეთრით.

**Color** - ფერი. მიიღება გამოსახულება ზედა ფენის ტონით და გაჯერებულობით და ქვედა ფენის სიმკვეთრით.

**Luminosity** - სიმკვეთრე. მიიღება გამოსახულება ზედა ფენის სიმკვეთრით და ქვედა ფენის ტონით და გაჯერებულო- ბით.

## სიმბოლოები

ეს არის მზა, სხვადასხვა სახის ობიექტები, რომლებიც პროგრამას მოყვება. მათი გამოძახება შეიძლება Window -> Symbols:




სურ. 2.2-17. სიმბოლოების პანელი.

დამატებითი სიმბოლოების ჩატვირთვა შესაძლებელია სიმბოლოების პანელზე, მარცხენა ქვედა კუთხეში განთავსებული ღილაკის დახმარებით ან Window -> Symbol Libraries.

მოკიდეთ ხელი სიმბოლოს და გადაიტანეთ ის სამუშაო დაფაზე. ეს არის ობიექტებისგან შემდგარი კომპლექსი და ამ ეტაპზე შეგიძლიათ მას მხოლოდ ზომები უცვალოთ.

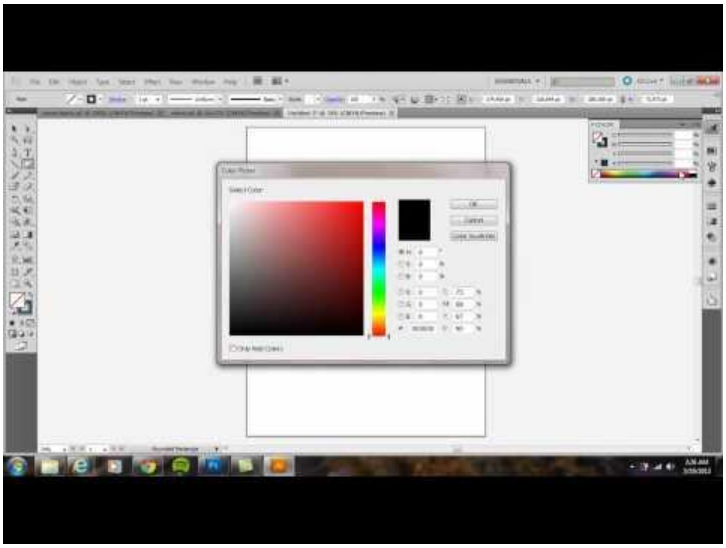
სიმბოლოს რედაქტირებისათვის, სიმბოლოების პანელზე ორჯერ სწრაფად დააწკაპუნეთ სიმბოლოს და გადახვალთ სიმბოლოს რედაქტირების რეჟიმში, სადაც მას შეგიძლიათ შეუცვალოთ ზომა, ფერი, მოხაზულობა და სხვა. რეჟიმიდან გამოსვლისას, თქვენ შეცვლილი სიმბოლო დაგხვდებათ.

**შენიშვნა:** *რედაქტირების შედეგი აისახება ყველა მანამდე გამოყენებულ სიმბოლოზეც.*

ინსტრუმენტთა პანელზე არის ინსტრუმენტები, რომელთა დახმარებით შესაძლებელია სიმბოლოებთან მუშაობა:

სურ. 2.2-18. მარცხნიდან მარჯვნივ: 1. სიმბოლოების სპრეი - მისი დახმარებით ხდება არჩეულის სიმბოლოების დიდი რაოდენობით დატანა დაფაზე; 2. წანაცვლება - მჭიდროდ დაყრილი სიმბოლოების გაშლა; 3. შემჭიდროვება - სიმბოლოების ერთმანეთთან მიახლოება; 4. ზომა - სიმბოლოს ზომაში გადიდება. Alt კლავიშის გამოყენებით შემცირება; 5. შემობრუნება; 6. გაუფერულება - ერთფეროვანი სიმბოლოს შექმნა; 7. სიმბოლოს სიღიავის მართვა; 8. სიმბოლოსთვის გრაფიკული სტილის გამოყენება.

გარდა ამისა შეგიძლიათ საკუთარი სიმბოლოების შექმნა და მათი გამოყენება მუშაობაში.



ვიდეო 2.2-2. გაეცანით სიმბოლოებთან მუშაობის გაკვეთილს.

## ტექსტი და მისი რედაქტირება

### *ჩვეულებრივი ტექსტი*

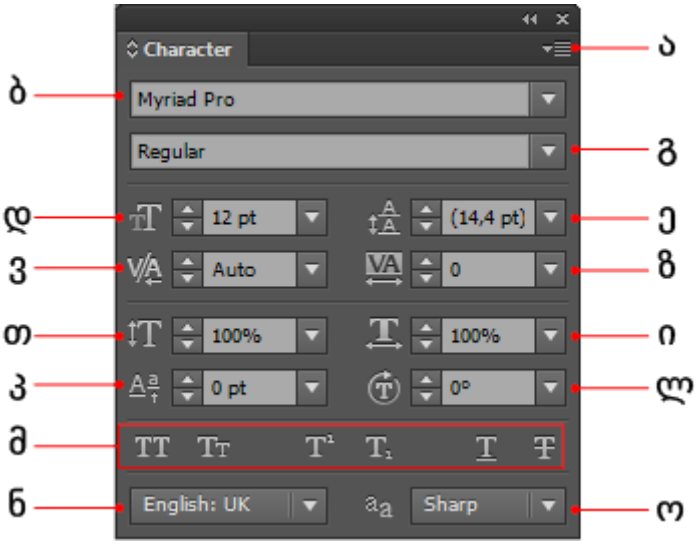
ტექსტებთან სამუშაოდ გამოიყენება ტექსტის ინსტრუმენტი ინსტრუმენტთა პანელზე:

სურ. 2.2-19. მარცხნიდან მარჯვნივ: 1. ქმნის ტექსტის ბლოკს, რომელშიც ტექსტი განთავსდება; 2. ნებისმიერი ფორმის არეში ტექსტის განთავსება; 3. წირზე ტექსტის განთავსება; 4. ვერტიკალური ტექსტი; 5. ნებისმიერი ფორმის არეში ვერტიკალური ტექსტის განთავსება; 6. ვერტიკალური ტექსტის განთავსება წირზე.

ტექსტის ინსტრუმენტის არჩევის შემდეგ ფორმატირების ზოლზე ჩნდება შესაბამისი ფუნქციები. მათი დახმარებით შეგიძლიათ აირჩიოთ შრიფტი, მისი ზომა, ფერი, სწორება და სხვა.

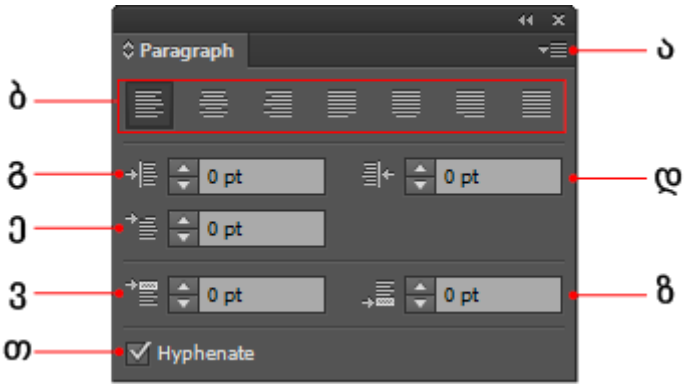

ტექსტის ფორმატირებისთვის სამუშაო პანელების მოთავსებული მენიუ Window-ში:

Window -> Type -> Character - გამოიყენება შრიფტის ფორმატირებისათვის;




სურ. 2.2-20. ა - სიმბოლოს ფორმატირების პანელის მენიუ; ბ - შრიფტის არჩევა; გ - შრიფტის სტილის დაყენება (მუქი, დახრილი და ა.შ.); დ - შრიფტის ზომის დაყენება; ე - სტრიქონებს შორის მანძილი; ვ - კერნინგი; ზ - ინტერვალი; თ - სიმბოლოს მასშტაბირება (ვერტიკალურად); ი - სიმბოლოს მასშტაბირება (ჰორიზონტალურად); კ - საბაზისო ხაზის მართვა; ლ - ტექსტის დახრა; მ - ცალკეული სიმბოლოების ფორმატირების ინსტრუმენტები; ნ - ენის არჩევა; ო - შრიფტის მოხაზულობის გასწორების მეთოდები

Window -> Type -> Paragraph - გამოიყენება აბზაცების ფორმატირებისათვის.




სურ. 2.2-21. ა - პარაგრაფის პანელის მენიუ; ბ - აბზაცის სწორების მეთოდები (მარჯვნივ, მარცხნივ, ცენტრზე და ა.შ.); გ - დაცილება მარცხენა კიდიდან; დ - დაცილება მარჯვენა კიდიდან; ე - პირველი სტრიქონის დაცილება მარცხნიდან; ვ - დაცილება აბზაცამდე; ზ - დაცილება აბზაცის შემდეგ; თ - გადატანები ტექსტისთვის.

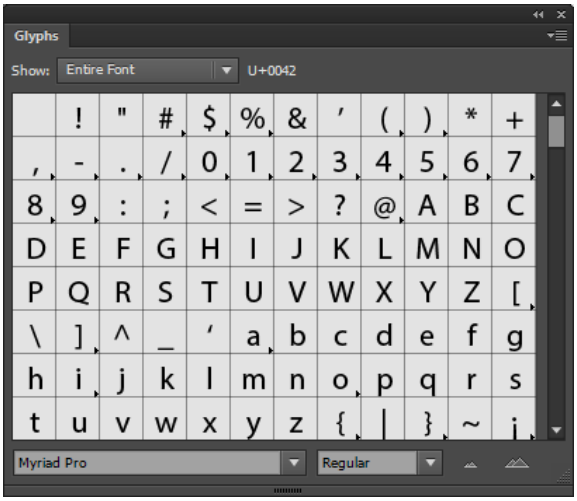
ილუსტრატორს საკმაოდ მოქნილად შეუძლია ტექსტებთან მუშაობა. მენიუ Type-ის დახმარებით შეგიძლიათ სხვადასხვა მოქმედებების შესრულება, მათ შორის შრიფტისა და მისი ზომის არჩევაც.

ხანდახან ჩნდება საჭიროება, როდესაც გვინდა ისეთი სიმბოლოების აკრეფა, რომელიც ან ვერ თავსდება სტანდარტული კლავიატურის განლაგებაზე ან უბრალოდ არ ვიცით, როგორ უნდა ავკრიფოთ. ამისათვის გამოიყენება Glyph პანელი, რომლის დახმარებითაც თქვენ შეგიძლიათ შრიფტში შემავალი ნებისმიერი სიმბოლოს აკრეფა.

მისი გამოძახება შეიძლება:

Type -> Glyph ან

Window -> Type -> Glyph




სურ. 2.2-22. ამ პანელის დახმარებით ნებისმიერი სიმბოლოს მოძიებაა შესაძლებელი.

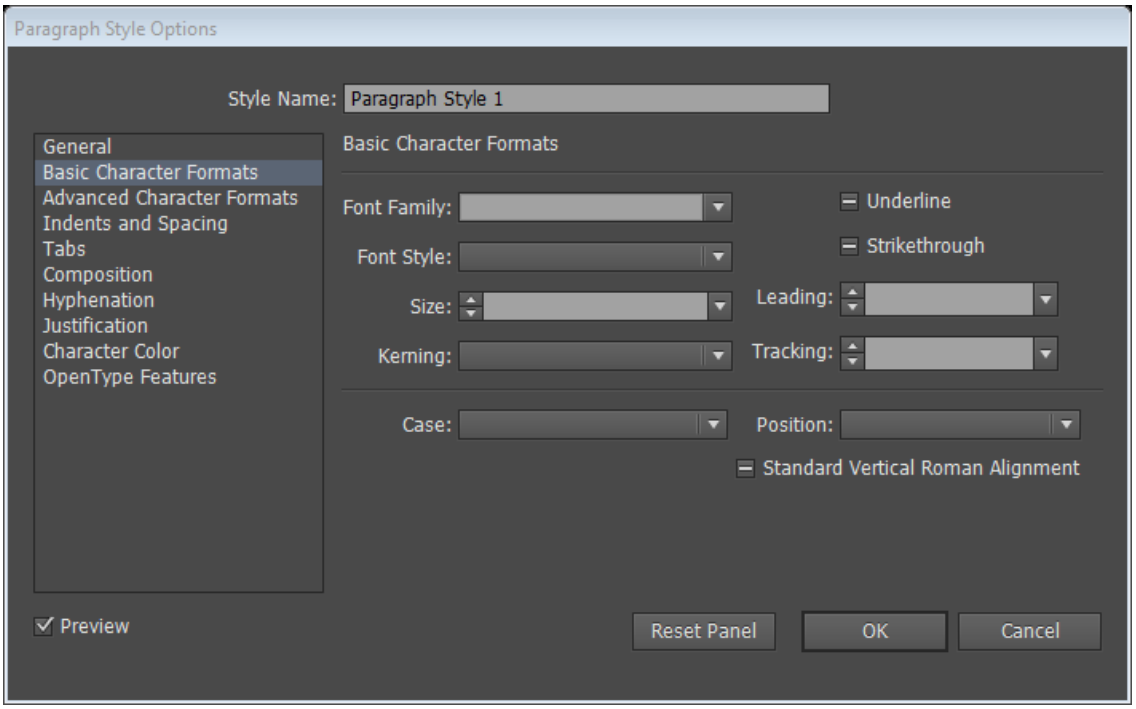
მუშაობის პროცესში არის ისეთი მომენტები, როდესაც თქვენ ხშირად გჭირდებათ ერთი და იმავე ფორმატირების გამოყენება ტექსტისთვის, მაგალითად ბანერზე ტექსტური წარწერები ან სათაურები, ცალკეული სიმბოლოს ფორმატი და სხვა.

ამისათვის თქვენ შეგიძლიათ ისარგებლოთ აბზაცისა და სიმბოლოს სტილებით. სტილები, ისევე როგორც ინდიზაინში, გაძლევენ საშუალებას სწრაფად მოახდინოთ ტექსტის ფორმატირება. მასში უკვე ჩადებულია ფორმატირების ის მახასიათებლები, რომლებიც თქვენ გამოიყენეთ აბზაცისთვის თუ ცალკეული სიმბოლოსთვის.

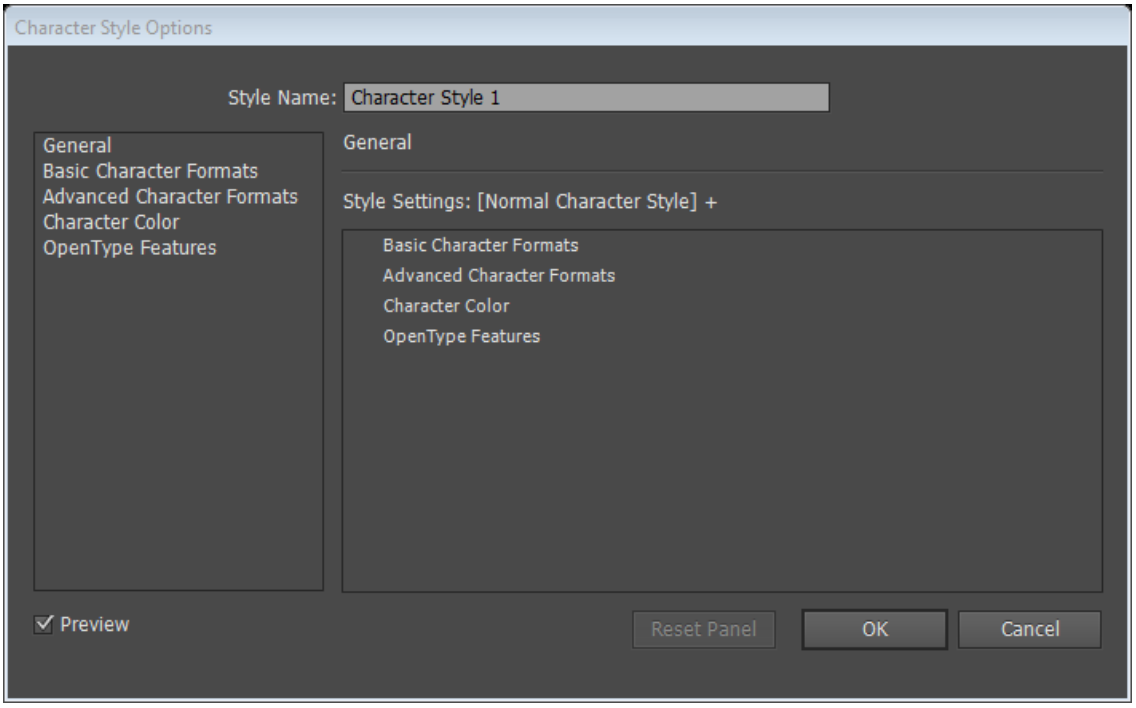
Window -> Paragraph Style - აბზაცის სტილების პანელი;

Window -> Character Style - სიმბოლოების სტილების პანელი.

უნდა გაითვალისწინოთ, რომ აბზაცის სტილი მოქმედებს მთლიანად აბზაცზე, ხოლო სიმბოლოს სტილი ვრცელდება მხოლოდ ერთ კონკრეტულ სიმბოლოზე (ის წინასწარ უნდა მონიშნოთ).




სურ. 2.2-23. ახალი აბზაცის სტილის შექმნის ფანჯარა. აქ შეგიძლიათ სტილს მიანიჭოთ დასახელება, დააყენოთ შრიფტი, ზომა, ფერი და ყველა ის მახასიათებელი რაც სიმბოლოსა და აბზაცის ფორმატირების პანელებზე იყო მოცემული. შემდეგ საკმარისია მონიშნოთ ტექსტი აირჩიოთ სტილი და ის ავტომატურად მიიღებს შესაბამის სახეს.




სურ. 2.2-24. სიმბოლოს სტილის შექმნა. აქ თქვენ გამოიყენებთ მხოლოდ სიმბოლოს ფორმატირების პანელის მახასიათებლებს. შეგიძლიათ სტილს დაარქვათ სახელი, აირჩიოთ შრიფტი, დააყენოთ ზომა, ფერი და ა,შ. შემდეგში ეს სტილი გავრცელდება მხოლოდ იმ სიმბოლოზე ან სიტყვაზე, რომელსაც მონიშნავთ. მაგალითად, როდესაც გინდათ, რომ მთელი აბზაცის მასშტაბით ერთი კონკრეტული სიტყვა იყოს ყოველთვის წითელი, ასეთ შემთხვევაში სიმბოლოების სტილი შეუცვლელია.

### *ტექსტის წირებში გადაყვანა*

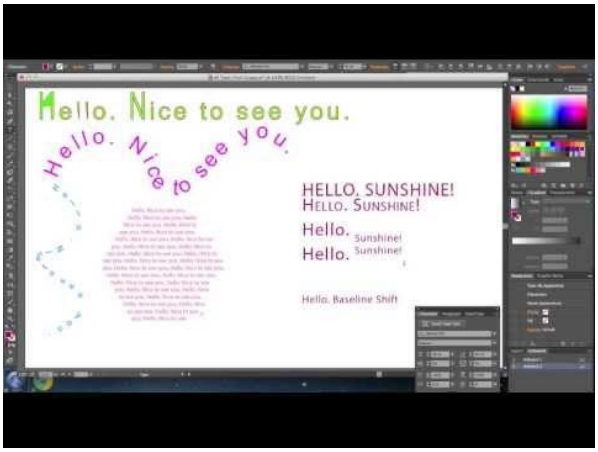
როდესაც გადაწყვეტთ ჩვეულებრივი წარწერისგან მხატვრული წარწერა შექმნათ, მაშინ ის წირებში უნდა გადაიყვანოთ. წინააღმდეგ შემთხვევაში თქვენ მხოლოდ შრიფტის ზომის და ფერის შეცვლა თუ შეგიძლიათ.

ამისათვის მონიშნეთ ტექსტი და გამოიძახეთ Type -> Create Outlines (Shift + Ctrl + O).

ტექსტი გარდაიქმნება წირებში და თქვენ მისი რედაქტირება უკვე სხვა ინსტრუმენტებით შეგიძლიათ, მაგრამ მას უკვე ვერ გაარედაქტირებთ, როგორც ტექსტს.


სურ. 2.2-25. მარცხნივ - ჩვეულებრივი ტექსტი, მარჯვნივ - წირებში გადაყვანილი.

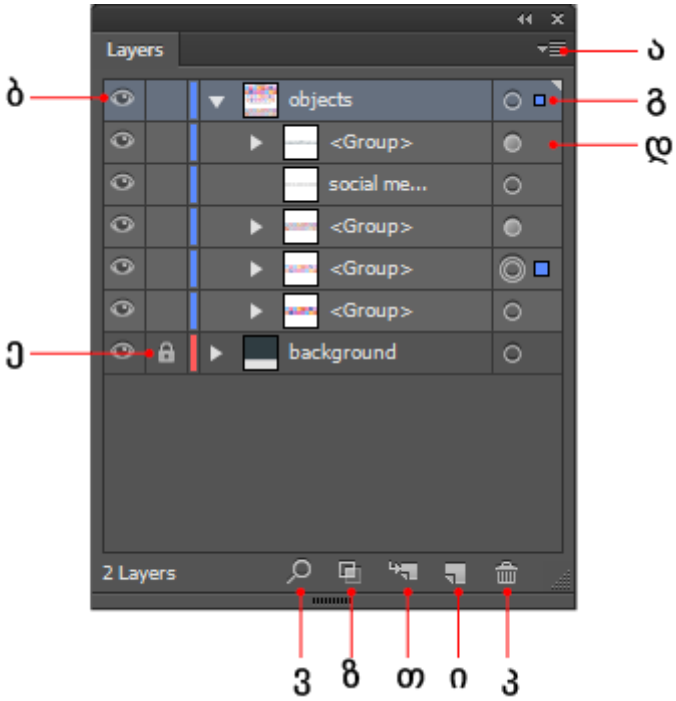
წირებში გადაყვანილი ტექსტის ფორმების შეცვლა შესაძლებელია არსებული საკვანძო წერტილების დახმარებით.




ვიდეო 2.2-3. ილუსტრატორში ტექსტთან მუშაობის საწყისები.

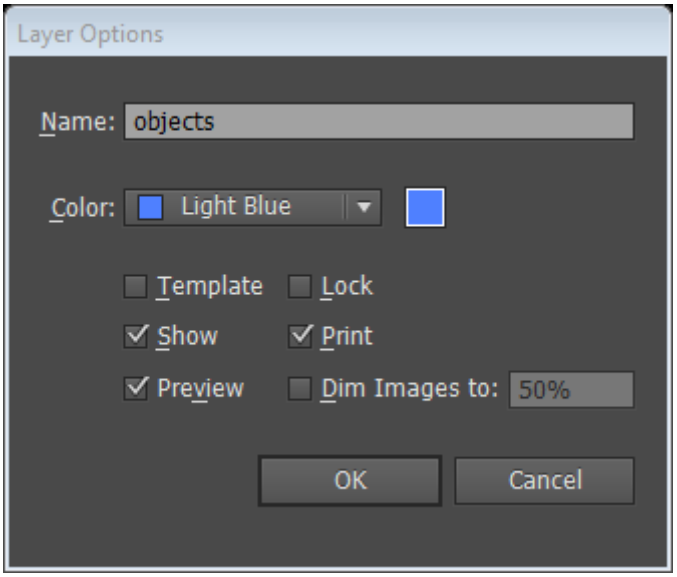
## ფენები და მათთან მუშაობა

ფენებთან სამუშაოდ გამოიძახეთ Window -> Layers (F7).




სურ. 2.2-26. ფენების პანელი. ა - ფენების პანელის მენიუ;, ბ - ფენების/ობიექტების გათიშვა/გამოჩენა; გ - მიუთითებს, რომ მიმდინარე ფენაზე იმე ობიექტი მონიშნულია; დ - მიუთითებს, რომ ობიექტთან (ჯგუფთან) გამოყენებულია რაიმე ეფექტი. შესამოწმებლად გამოიძახეთ წარმოდგენის პანელი (Window –> Appearance); ე - ფენის/ობიექტის დაბლოკვა/განბლოკვა; ვ - ობიექტის მოძიება ფენების პანელზე; ზ - შემოჭრის ნიღაბის შექმნა/დაშლა; თ - ქვეფენის შექმნა; ი - ახალი ფენის შექმნა; კ - ფენის/ობიექტის წაშლა.

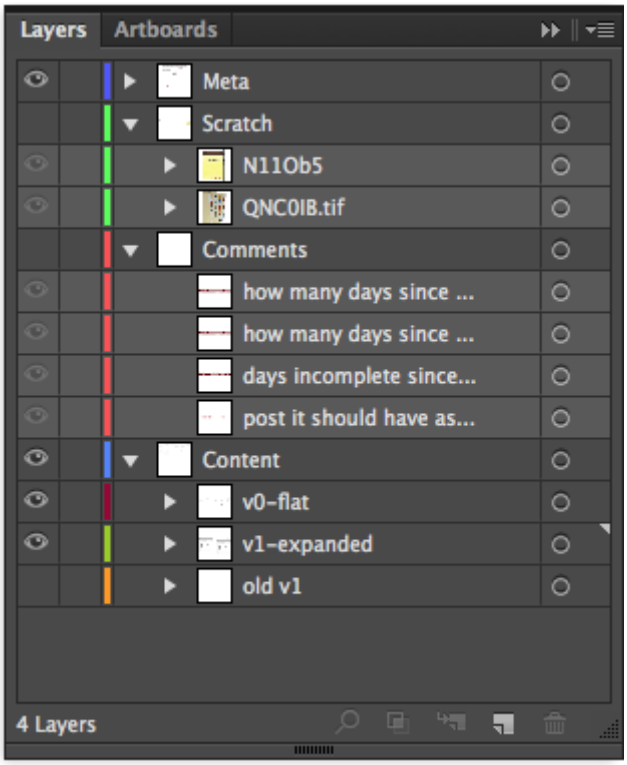
სტანდარტულად ფაილში მხოლოდ ერთი ფენაა. საჭიროების შემთხვევაში შეგიძლიათ დაამატოთ ახალი ფენები. ამისათვის გამოიყენეთ ახალი ფენის შექმნის ღილაკი (სურ. 2.2-26, ი). ფენების სტანდარტული სახელებია Layer 1, Layer 2 … და ა.შ. და მათ ავტომატურად ენიჭებათ ფერი. იმისათის, რომ შეცვალოთ ფენის მახასიათებლები, ორჯერ დააწექით ფენას და გამოვა ფენების მახასიათებლების ფანჯარა:




სურ. 2.2-27. ამ ფანჯრის დახმარებით შეგიძლიათ ფენას შეუცვალოთ სახელი, ფერი და კიდევ სხვა დამატებითი პარამეტრები ჩაურთოთ, მაგ., დაბლოკვა, გამოჩენა, ბეჭდვა. თუ ამ უკანასკნელს გამორთავთ, ამ ფენაზე განთავსებული ობიექტები არ დაიბეჭდება.

ობიექტები, რომლებიც მუშაობის პროცესში იქმნება, ავტომატურად მიმდინარე ფენაზე თავსდება. ყოველ ობიექტს, რომელსაც მონიშნავთ, გარშემო უჩნდება ჩარჩო, რომლის დახმარებითაც შეგიძლიათ სხვადასხვა მოქმედებები შეასრულოთ. იმისდა მიხედვით, რომელ ფენაზე იმყოფება ობიექტი, ჩარჩოს შეფერილობა იქნება შესაბამისი (წითელი, მწვანე, ლურჯი). ეს გეხმარებათ თქვენ მიხვდეთ, რომელ ფენაზეა განთავსებული ობიექტი და შესაბამისად თქვენ რომელ ფენაზე მუშაობთ.

შენიშვნა: აუცილებლად მიიღეთ წესად ფენებსა და ჯგუფებს დაარქვათ სახელები. როდესაც ფენებისა და ობიექტების რაოდენობა ძალიან დიდი იქნება, ეს დაგეხმარებათ თქვენ მარტივად იპოვით საჭირო ფენა და შესაბამისად ობიექტიც.




სურ. 2.2-28. ფენებისთვის და ობიექტებისთვის სახელების გამოყენების მაგალითი.

ობიექტები შეგიძლიათ გადაადგილოთ როგორც ერთი ფენის ფარგლებში, ისე ფენებს შორის. ფენების პანელზე მოკიდეთ ხელი ობიექტის შესაბამის ფენას და გადაადგილეთ ის ფენების ფარგლებში ზემოთ-ქვემოთ ან ფენიდან ფენაზე.

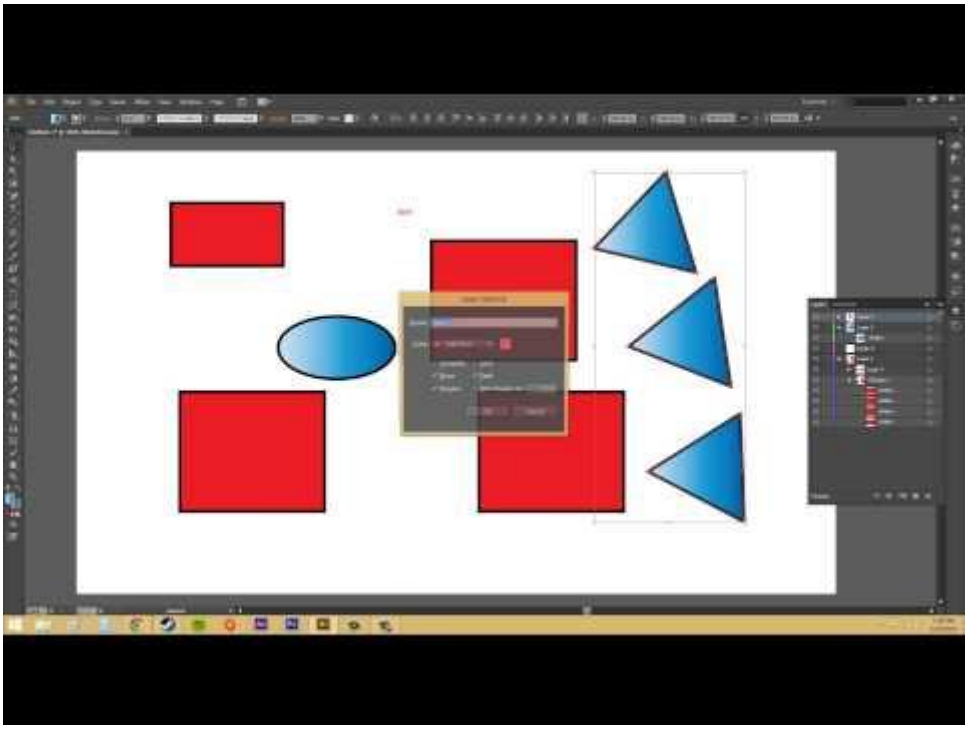
იმ შემთხვევაში, თუ ფენაზე ძალიან ბევრი კომპონენტია და თქვენ გინდათ სიაში იპოვოთ სასურველი, მონიშნეთ ის სამუშაო დაფაზე და შემდეგ ფენების პანელზე დააწექით ძიების ღილაკს (სურ. 2.2-26, ვ). ავტომატურად მოინიშნება შესაბამისი ობიექტი.

უამრავი ობიექტის მოსაწესრიგებლად შეგიძლიათ გამოიყენოთ ქვეფენის შექმნის ღილაკი (სურ. 2.2-26, თ). ეს არ მოგცემთ საშუალებას დაიბნეთ, როდესაც ძალიან ბევრი ობიექტია ერთ ფენაზე განთავსებული.

იმ შემთხვევაში, თუ ფენაზე მოთავსებული ობიექტები გინდათ გაიტანოთ ცალკე ფენაზე ანუ განაცალკევოთ ამ ფენისგან, გამოიყენეთ შემდეგი გზა. მონიშნეთ სასურველი ობიექტები, შემდეგ ფენების პანელის მენიუდან გამოიძახეთ ბრძანება Collect in New Layer. შეიქმნება ახალი ფენა და ეს ობიექტები იმ ფენაზე იქნებიან განთავსებულნი.

თუ გინდათ, რომ რამდენიმე ფენის ან ქვეფენის შიგთავსი გააერთიანოთ, მონიშნეთ სასურველი ფენები და პანელის მენიუში მიუთითეთ Merge Selected. ავტომატურად ობიექტები ერთ ფენაზე/ქვეფენაზე განთავსდებიან.

დამატებითი ინფორმაციისთვის იხილეთ ქვემოთ მოცემული ვიდეო:



ვიდეო 2.2-4. ფენების გამოყენება ილუსტრატორში.

## სავარჯიშო

1. შექმენით სხვადასხვა სახის გეომეტრიული ფიგურები.
2. შექმენით თავისუფალი ფორმები ფუნჯის, ფანქრისა და კალმის დახმარებით.
3. აიღეთ რომელიმე კომპანიის ლოგო (მაგ., ადიდასი, ნაიკი) და კალმის დახმარებით შექმენით იდენტური ფორმები.
4. წარმოდგენის პანელის დახმარებით შექმნილ ფორმებს შეუცვალეთ ფონისა და/ან კონტურის ფერი.
5. შექმენით საკუთარი ფუნჯები.
6. შექმენით სხვადასხვა სახის ტექსტი.
7. წირზე ტექსტის განთავსების დახმარებით შექმენით ბეჭდის ყალიბი.
8. Warp-ის დახმარებით შექმენით სხვადასხვა სახის ტექსტური ეფექტები.
9. შექმენით სხვადასხვა სახის სიმბოლოები.

## 2.3. ობიექტების რედაქტირება/ტრანსფორმირება

პარაგრაფის შესაბამისი თემატიკა

- მონიშვნის ინსტრუმენტები და მათი მოხმარება
- მონიშნული ობიექტების გასწორება ერთმანეთის და გვერდის მიმართ. მიმმართველების გამოყენება
- საკვანძო წერტილების შეცვლა
- ობიექტების ტრანსფორმაცია და დეფორმაცია
- ობიექტის გარსის დეფორმაცია
- ობიექტების დაჯგუფება
- ობიექტების კომბინირება
- ობიექტის დაშლა

რასაკვირველია მუშაობის პროცესში მხოლოდ ფორმების შექმნა არ არის. სასურველი შედეგი რომ მივიღოთ, უნდა მოვახდინოთ შექმნილი ობიექტების რედაქტირება. ეს შეიძლება იყოს მისი ფორმის შეცვლა, შემოტრიალება, დეფორმირება და სხვა.

### მონიშვნის ინსტრუმენტები

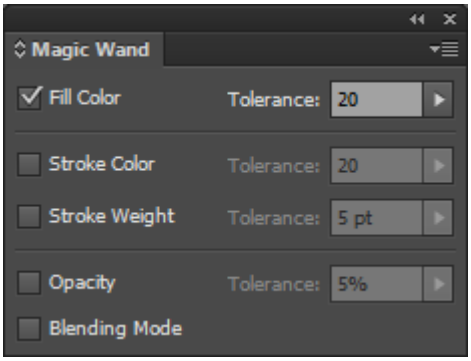
ობიექტზე რომ რაიმე მოქმედება შეასრულოთ, ამისათვის ის უნდა მონიშნოთ. მონიშვნის ინსტრუმენტები და ხერხები ერთმანეთისგან განსხვავდება: შეიძლება მონიშნოთ მთლიანი ობიექტი ან ობიექტთა ჯგუფი, ხოლო შეიძლება მონიშნოთ ობიექტის ან ობიექტთა ჯგუფის მხოლოდ რამდენიმე საკვანძო წერტილი. აქედან გამომდინარე პროგრამა ილუსტრატორში მოცემულია შესაბამისი მონიშვნის ინსტრუმენტები:

სურ. 2.3-1. ილუსტრატორის მონიშვნის ინსტრუმენტები (სურ. 2.1-14, A).

**მონიშვნის ინსტრუმენტი** (V) - მისი დახმარებით შესაძლებელია მთლიანი ობიექტის ან ობიექტთა ჯგუფის (Shift-ის დახმარებით) მონიშვნა;

**პირდაპირი მონიშვნა** (A) - მისი დახმარებით შესაძლებელია როგორც მთლიანი ობიექტის (როდესაც თვითონ ობიექტს დააწვებით), ისე ცალკეული საკვანძო წერტილების მონიშვნა;

**ჯადოსნური ჯოხი** (Y) - მისი დახმარებით შესაძლებელია ერთნაირი მახასიათებლების მქონე ობიექტების მონიშვნა. მახასიათებლები შეიძლება იყოს ფერი, კონტური ან ორივე ერთად და ა.შ. ჯადოსნური ჯოხის სამართავად გამოიძახეთ შესაბამისი პანელი Window -> Magic Wand:




სურ. 2.3-2. ჯადოსნური ჯოხის მართვის პანელი.

**ლასო** (Q) - მისი დახმარებით შესაძლებელია როგორც მთლიანი ობიექტის, ისე ცალკეული საკვანძო წერტილების მონიშვნა. ლასოს გამოყენებით მოხაზეთ არე და ამ არეში რა საკვანძო წერტილებიც მოხვდება, ყველა მოინიშნება. საკვანძო წერტილები შეიძლება ერთ და შეიძლება სხვადასხვა ობიექტებს ეკუთვნოდეს.

## ობიექტების გასწორება დაფაზე და ერთმანეთის მიმართ

### *მიმმართველები*

მიმმართველი არის დამხმარე ხაზი, რომელიც ზედა ან გვერდითი (მარცხენა) სახაზავიდან შეგვიძლია გამოვიტანოთ სამუშაო დაფაზე.

ახალი ფაილის შექმნისას სახაზავი არ ჩანს. მისი გამოჩენა შესაძლებელია View -> Rulers -> Show/Hide Ruler (Ctrl + R).

მიმმართველის გადმოსატანად სამუშაო დაფაზე, დადექით სახაზავზე, მოკიდეთ კურსორი და გადმოიტანეთ. შეგიძლიათ მიმმართველების ნებისმიერი რაოდენობის გადმოტანა სამუშაო დაფაზე.

მიმმართელების დახმარებით შეგიძლიათ ობიექტები გაასწოროთ ერთმანეთის მიმართ ან ირჩიოთ ისინი, როგორც საწყისი ათვლის წერტილები. მათი გამოყენება მრავალ საკითხში შეიძლება.

რადგან არსებობს ორი სახის სახაზავი - ჰორიზონტალური და ვერტიკალური - მიმმართველებიც შესაბამისად ორი სახისაა. გარდა ამისა, ნებისმიერი გავლებული ხაზი, ნებისმიერი კუთხით, შესაძლოა ვაქციოთ მიმმართველად. ამისათვის გამოიყენეთ ხაზის ინსტრუმენტი, გაავლეთ სასურველი ზომისა და დახრილობის ხაზი, შემდეგ მასზე გამოიძახეთ მენიუ მაუსის მარჯვენა ღილაკით და აირჩიეთ Make Guides.

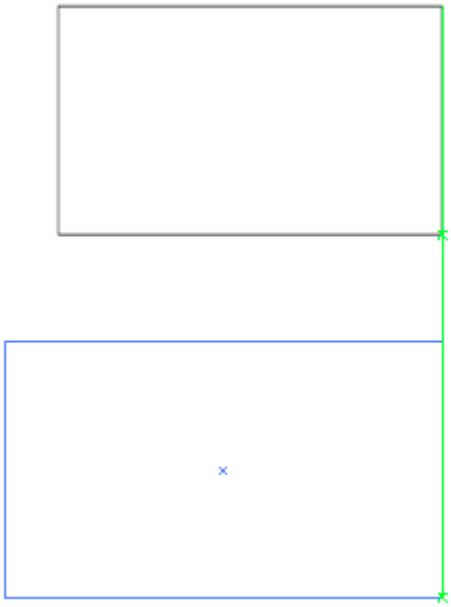
მიმმართველები, როგორც წესი ცისფერი ფერისაა. მათი ფერი რომ შეცვალოთ, გამოიძახეთ General -> Guides & Grid და ჩამოსაშლელ მენიუში Color აირჩიეთ სასურველი ფერი.

**შენიშვნა:** *კარგი იქნება თუ მიმმართველები განათავსებთ ცალკე ფენაზე. ეს მოგცემთ საშუალებას ნებისმიერ დროს გააქროთ ან გამოაჩინოთ ისინი და მარტივად დაბლოკოთ. დაარქვით ამ ფენას Guides და ისარგებლეთ.*

### *ჭკვიანი მიმმართველები*

ძალიან კარგია **ჭკვიანი მიმმართელების** (Smart Guides) გამოყენება მუშაობის პროცესში. მათ გასააქტიურებლად გამოიყენეთ View -> Smart Guides (Ctrl + U).

როდესაც თქვენ გადააადგილებთ ან ხაზავთ ობიექტს, ჭკვიანი მიმმართველი ყოველთვის დაგეხმარებათ განსაზღვროთ, გაუსწორდა თუ არა ობიექტის კიდე ან ცენტრი სხვა ობიექტს, დაინახავთ ობიექტები თანაბრად არიან თუ არა დაცილებულნი ერთმანეთს და ა.შ. ძალიან მოსახერხებელია და პრაქტიკულია მისი გამოყენება. ერთადერთი ნაკლი, რაც გააჩნია ჭკვიან მიმმართველებს, არის ის, რომ როდესაც ძალიან ბევრი ობიექტია, ჭკვიანი მიმმართველი „იბნევა“ და რთული ხდება მისი დახმარებით ობიექტის გასწორება სასურველი ობიექტის მიმართ.



სურ. 2.3-3. ჭკვიანი მიმმართველის გამოყენების მაგალითი. მწვანე ფერით აღნიშნულია ჭკვიანი მიმმართველი.

### *ობიექტების სწორება*

არის მომენტები, როდესაც ვერც ჩვეულებრივი და ვერც ჭკვიანი მიმმართველები, ვერ გვეხმარებიან ობიექტების გასწორებაში ერთმანეთის მიმართ ან მაგალითად, როდესაც საჭიროა თანაბარი მანძილების განსაზღვრა ობიექტებს შორის.

ამისთვის გამოიყენება სწორების (Align) ბრძანებები.

როდესაც მონიშნული გაქვთ ერთზე მეტი ობიექტი, ფორმატირების ზოლზე ჩნდება სწორების ღილაკები:

სურ. 2.3-4. ობიექტების ერთმანეთთან სწორების ღილაკები.

პირველი ღილაკი (შეიცავს ჩამოსაშლელ მენიუს) გამოიყენება სწორების განსაზღვრისათვის, ხოლო დანარჩენი ღილაკები დაყენებული პირობის შესაბამისად ასწორებს ობიექტებს.

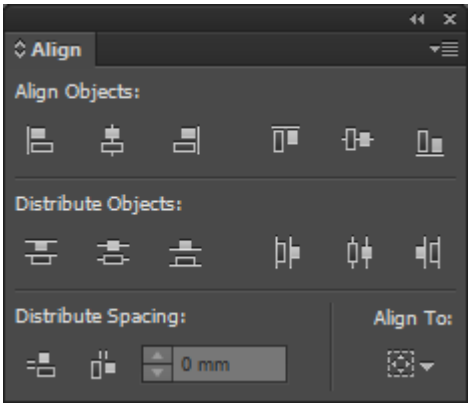
ღილაკი შეიცავს შემდეგ ფუნქციებს:

Align to Selection - მონიშნული ობიექტების სწორება ერთმანეთის მიმართ;

Align to Key Object - ობიექტ(ებ)ის სწორება საკვანძო ობიექტის მიმართ;

Align to Artboard - ობიექტ(ებ)ის სწორება სამუშაო დაფის მიმართ. ეს ფუნქცია შეგიძლიათ გააქტიუროთ, როდესაც ერთი ობიექტი გაქვთ მონიშნული და შემდეგ ისარგებლოთ სწორების ღილაკებით.

გარდა ამისა შეგიძლიათ გამოიძახოთ სწორების პანელი Window -> Align (Shift + F7):



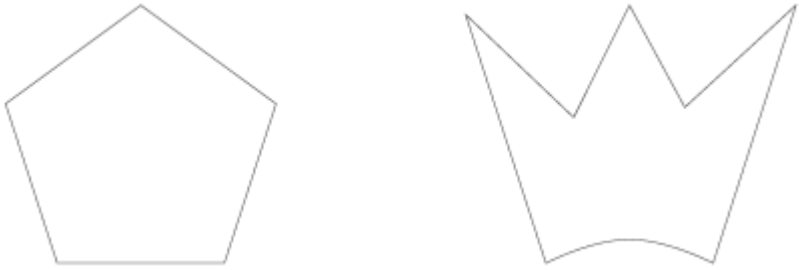

სურ. 2.3-5. ობიექტების სწორებისა და განაწილების პანელი.

როგორც ხედავთ, ეს პანელი შეიცავს ისეთივე ღილაკებს, როგორებიც ფორმატირების ზოლზე ჩნდება და ამის გარდა არის კიდევ დამატებითი ღილაკები (Distribute Spacing), რომელებიც დაგეხმარებათ ობიექტები თანაბარი მანძილით დააცილოთ ერთმანეთს. შეგიძლიათ ასევე მიუთითოთ რიცხობრივი მაჩვენებელი - რა მანძილით უნდა დაცილდნენ ობიექტები ერთმანეთს.

## საკვანძო წერტილების შეცვლა

**პირდაპირი მონიშვნის** ან სხვა ინსტრუმენტის დახმარებით საკვანძო წერტილის მონიშვნის შემდეგ თქვენ შეგიძლიათ მასზე მოქმედებების შესრულება.

კალმის გამოყენებით ჩვენ შეგვიძლია დახატვისას, გზადაგზა ვცვალოთ ობიექტის ფორმები (იხ. სურ. 2.2-9). მაგრამ, როგორ მოვიქცეთ როცა ფიგურა უკვე შექმნილია და მისი შეცვლა გინდათ? ან მოცემულია გეომეტრიული ფიგურა და ეხლა მისი ფორმა უნდა შეიცვალოს. ამისათვის გამოიყენეთ **კუთხის** (კალამთან ერთად ერთ ჯგუფში შედის) (იხ. სურ. 2.2-10) ან იმავე **პირდაპირი მონიშვნის** ინსტრუმენტის დახმარებით. გარდა ამისა, ამ უკანასკნელით შეგიძლიათ მონიშნული საკვანძო წერტილისთვის მდებარეობის შეცვლა



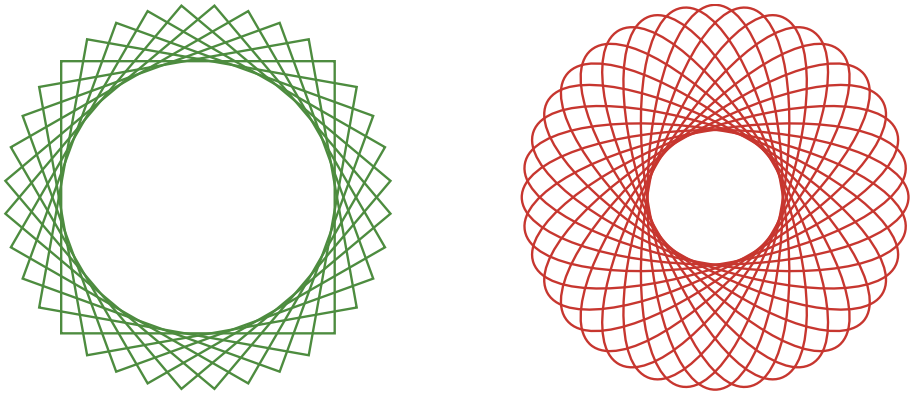
სურ. 2.3-6. ჩვეულებრივი მრავალკუთხედი და მისი სახეცვლილება.

## ტრანსფორმირება

ტრანსფორმირებაში შედის: **გადაადგილება** (Move), **დატრიალება** (Rotate), **ანარეკლი** (Reflect), **მასშტაბირება** (Scale), **წანაცვლება** (Shear).

ამ ფუნქციებს მონახავთ მენიუში Object -> Transform. ყოველი ზემოთ ჩამოთვლილი ფუნქციის გამოძახებისას, გამოდის ფანჯარა, სადაც შესაბამისი მნიშვნელობების მითითებისა და დასტურის შემდეგ ხდება შესაბამისი ტრანსფორმაციის შესრულება.

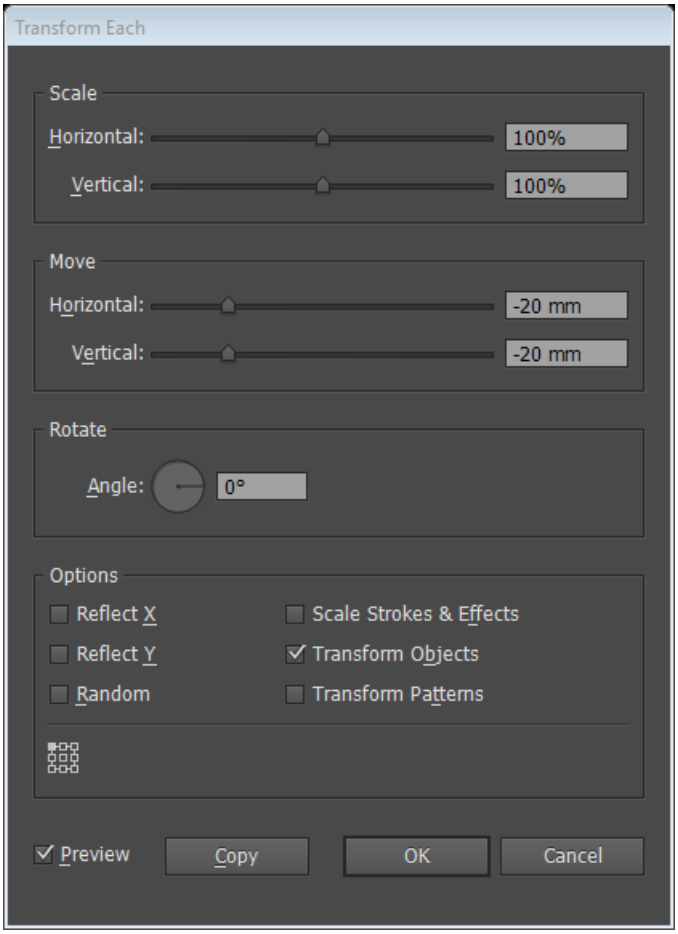
ბოლოს შესრულებული ტრანსფორმაციის ბრძანება რომ შეასრულოთ, გამოიყენეთ ბრძანება **ტრანსფორმაციის გამეორება**: Object -> Transform -> Transform Again (Ctrl + D).



სურ. 2.3-7. ეს ორი ობიექტი მიღებული იყო მხოლოდ გეომეტრიული ფიგურისა და დატრიალების ტრანსფორმაციის გამოყენებით.

ფუნქციაში Object -> Transform -> Transform Each (Alt + Shift + Ctrl +D) თავმოყრილია ყველა ზემოთ ჩამოთვლილი ფუნქცია. შეგიძლიათ აქედანვე არეგულიროთ, როგორი ტრანსფორმაცია უნდა განიცადოს ობიექტმა.

**შენიშვნა:** *ყველა ტრანსფორმაციის ფანჯარა შეიცავს ღილაკს Copy. მისი დახმარებით თქვენ მიიღებთ ტრანსფორმირებულ ობიექტსაც და ორიგინალიც დაგრჩებათ. თუ რამე არ მოგეწონებათ, ორიგინალი ყოველთვის ხელთ იქნება. ხშირად ისარგებლეთ ამ ღილაკით.*




სურ. 2.3-8. ფუნქცია Transform Each.

ზემოთ ჩამოთვლილი ტრანსფორმაციის ფუნქციებს შესაბამისი ინსტრუმენტები გააჩნიათ ინსტრუმენტთა პანელზე. ინსტრუმენტზე ორჯერ სწრაფად დაჭერისას, გამოვა დამატებითი პარამეტრების ფანჯარა, სადაც სასურველი მნიშვნელობების მითითება შეგიძლიათ.

გარდა ამისა პროგრამაში არის ტრანსფორმაციის პანელი. მისი გამოძახება შეიძლება Window -> Transform (Shift + F8). ამ პანელის დახმარებით შეგიძლიათ სწრაფად შემოაბრუნოთ, გაზარდოთ/შეამციროთ ან დახაროთ ობიექტი შესაბამის ველში მნიშვნელობების მითითებით.

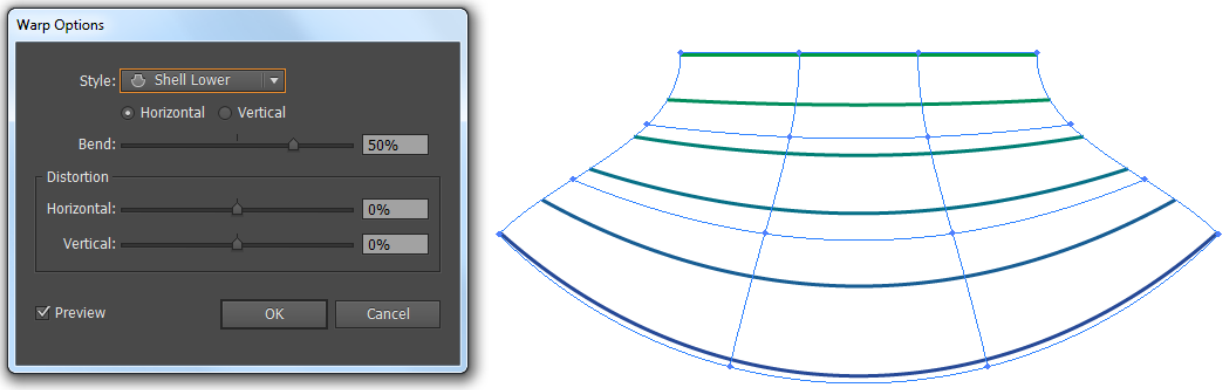
## ობიექტების დეფორმირება

სურ. 2.3-9. დეფორმაციის ინსტრუმენტები.

**სიგანე** - ერთი კონტურისთვის ქმნის სხვადასხვა სისქის კანტს;
**დეფორმაცია** - კურსორის მოძრაობის დახმარებით ობიექტის უცვლის ფორმას;
**ჩახვევა** - ახდენს ობიექტის „ჩახვევას“ ცენტრალური წერტილის მიმართ;
**შეკუმშვა** - ახდენს ობიექტის შეკუმშვას ცენტრალური წერტილისკენ;
**გაბერვა** - ახდენს ობიექტის გაფართოებას ცენტრალური წერტილის მიმართ;
**კბილანები** - ობიექტის კონტურებს შემთხვევითობის პრინციპით უმატებს კბილანებს;
**კრისტალიზაცია** - შემთხვევითობის პრინციპით კონტურს იმატებს ეკლების მსგავს ელემენტებს;
**ნაოჭები** - კონტურს უმატებს ნაოჭების მსგავს ელემენტებს.
პირველი ინსტრუმენტის გარდა, ყველა სხვა ინსტრუმენტზე ორჯერ დაჭერისას გამოვა ფანჯარა, სადაც შეგიძლიათ მისი მახასიათებლები ცვალოთ - ზომა, ინტენსივობა და ა.შ.

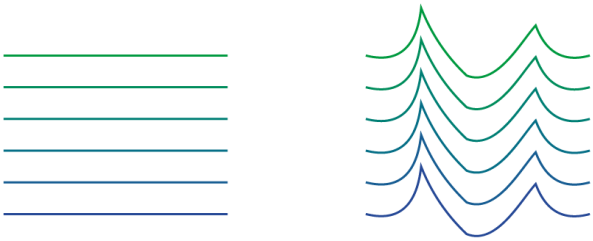
## გარსის დეფორმაცია

Object -> Envelope Distort -> Make with Warp (Alt + Shift + Ctrl + W) - ობიექტის დეფორმირება . გამოსულ ფანჯარაში შეგიძლიათ აირჩიოთ, რა ფორმით უნდა მოხდეს დეფორმირება და მიუთითოთ პარამეტრები.



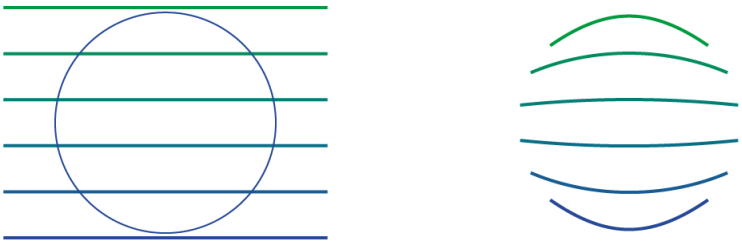

სურ. 2.3-10. გარსის დეფორმაცია.

Object -> Envelope Distort -> Make with Mesh (Alt + Ctrl + M) - ამ დროს ხდება ობიექტის ბადის სახით დაყოფა და ბადის დეფორმირების შედეგად ხდება ობიექტის დეფორმაცია.



სურ. 2.3-11. გარსის დეფორმაცია ბადის დახმარებით.

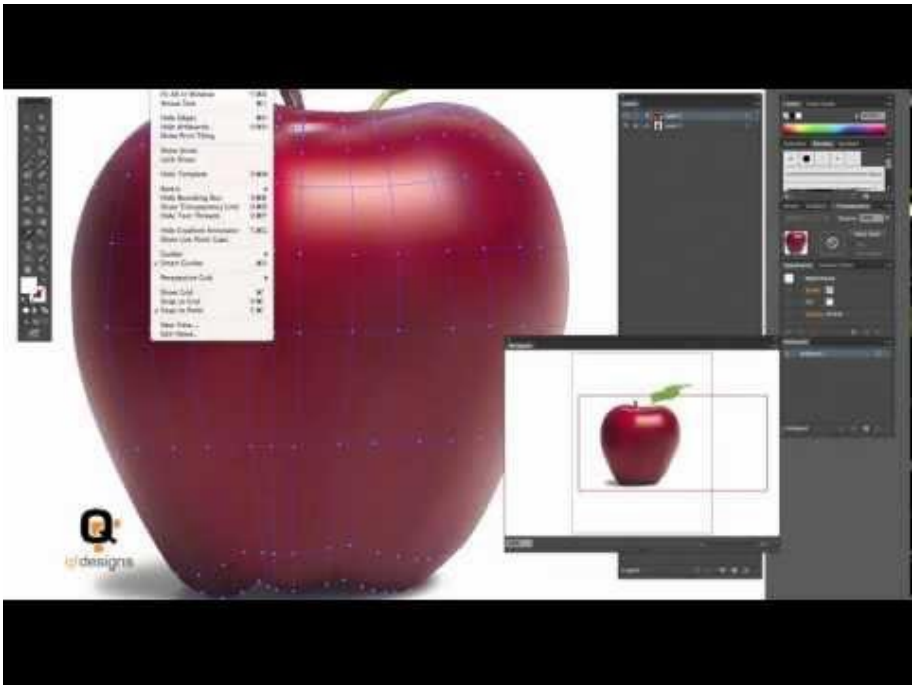
Object -> Envelope Distort -> Make with Top Object (Alt + Ctrl + C) - ამ დროს დეფორმაცია ხდება ზემოდან დადებული ობიექტის ფორმის მიხედვით.



სურ. 2.3-12. გარსის დეფორმაცია ზემოდან დადებული ფორმის მიხედვით.

### *გრადიენტული ბადის გამოყენება*

ინსტრუმენტთა პანელზე ასევე არის ინსტრუმენტი **ბადე** (Mesh). მას ასევე გრადიენტულ ბადეს უწოდებენ. მისი დახმარებით ხდება ობიექტის ბადის სახით დაყოფა. შეგიძლიათ ცალკეულ სეგმენტს უცვალოთ ფორმა და ფერი ჩაასხათ. ფერის ჩასხმა ხდება საკვანძო წერტილის გარშემო. შედეგად ძალიან საინტერესო ეფექტების მიღება შეგიძლიათ.



ვიდეო 2.3-1. გრადიენტული ბადის გამოყენება დამწყებთათვის.

## მუშაობა ობიექტების ჯგუფებთან

მუშაობის პროცესში ხანდახან საჭიროა ობიექტები ერთად დააჯგუფოთ, რომ ყველასთვის ერთიანი მოქმედება შეასრულოთ, მაგ., გადაადგილება, შემობრუნებ და ა.შ. ამ დროს ობიექტები შეიძლება იმყოფებოდნენ როგორც ახლოს, ისე მაკეტზე სხვადასხვა ადგილზე იყვნენ განთავსებულნი.

ამისათვის ისარგებლეთ Object -> Group (Ctrl + G)

ჯგუფის დასაშლელად გამოიყენეთ ბრძანება Object -> Ungroup (Shift + Ctrl +G)

როდესაც მაკეტზე ძალიან ბევრი ობიექტია, ამან შეიძლება ხელი შეგიშალოთ ცალკეული ობიექტების მონიშვნაში ან გადაადგილებაში.

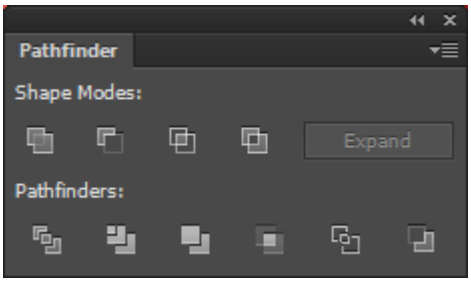
გამოიყენეთ Objcet -> Lock + Selection (Ctrl + 2), რომ დაბლოკოთ მონიშნული ობიექტ(ებ)ი. ამის შემდეგ მათი მონიშვნა ან მათზე რაიმე მოქმედების შესრულება შეუძლებელი გახდება. ბლოკის მოსახსნელად გამოიყენეთ Objcet -> Unlock All (Alt + Ctrl + 2).

ობიექტები, რომლებიც დროებით არ გჭირდებათ შეგიძლიათ. ამისათვის გამოიძახეთ Object -> Hide -> Selection (Ctrl + 3). დამალული ობიექტების გამოსაჩენად გამოიყენეთ Object -> Show All (Alt + Ctrl +3).

## ობიექტების კომბინირება

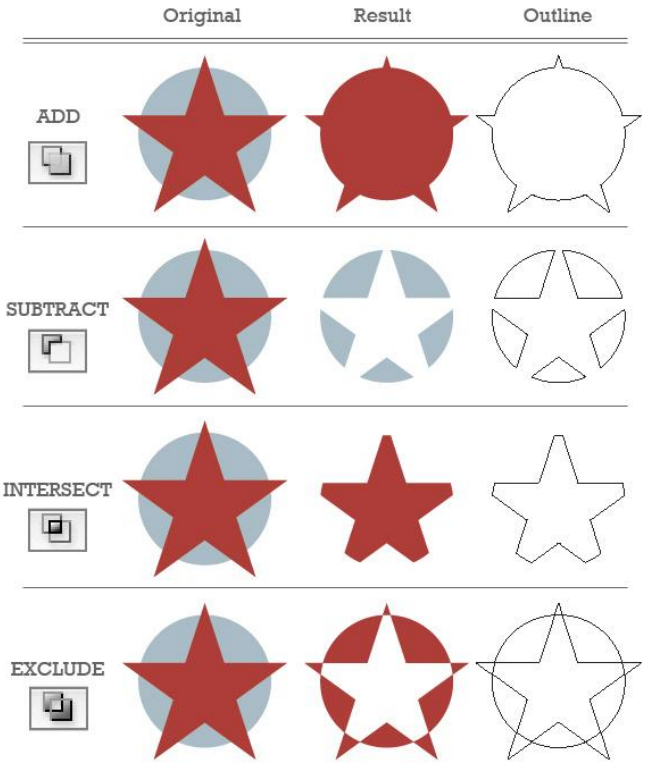
ჯგუფების შექმნის გარდა შესაძლებელია ასევე ობიექტების ერთმანეთთან კომბინირება. შედეგად თქვენ მიიღებთ განსხვავებული ფორმის ობიექტს. ამისათვის გამოიყენება Pathfinder: Window -> Pathfinder (Shift + Ctrl + F9)

მისი დახმარებით შეგიძლიათ სხვადასხვა ფორმის ობიექტების შექმნა და მათთან მუშაობს. ამ დროს შეგიძლიათ გამოიყენოთ სხვადასხვა ფორმისა და ზომის ობიექტები საბოლოო შედეგის მისაღწევად.




სურ. 2.3-13. სამუშაო პანელი Pathfinder.

პანელზე მოთავსებულია ღილაკების ორი ზოლი. ზედა ზოლი (Shape Modes) უზრუნველყოფს ფორმებთან მუშაობას, ხოლო ქვედა ზოლი (Pathfinders) კონტურებთან.




სურ. 2.3-14. ფორმების რეჟიმის მაგალითი

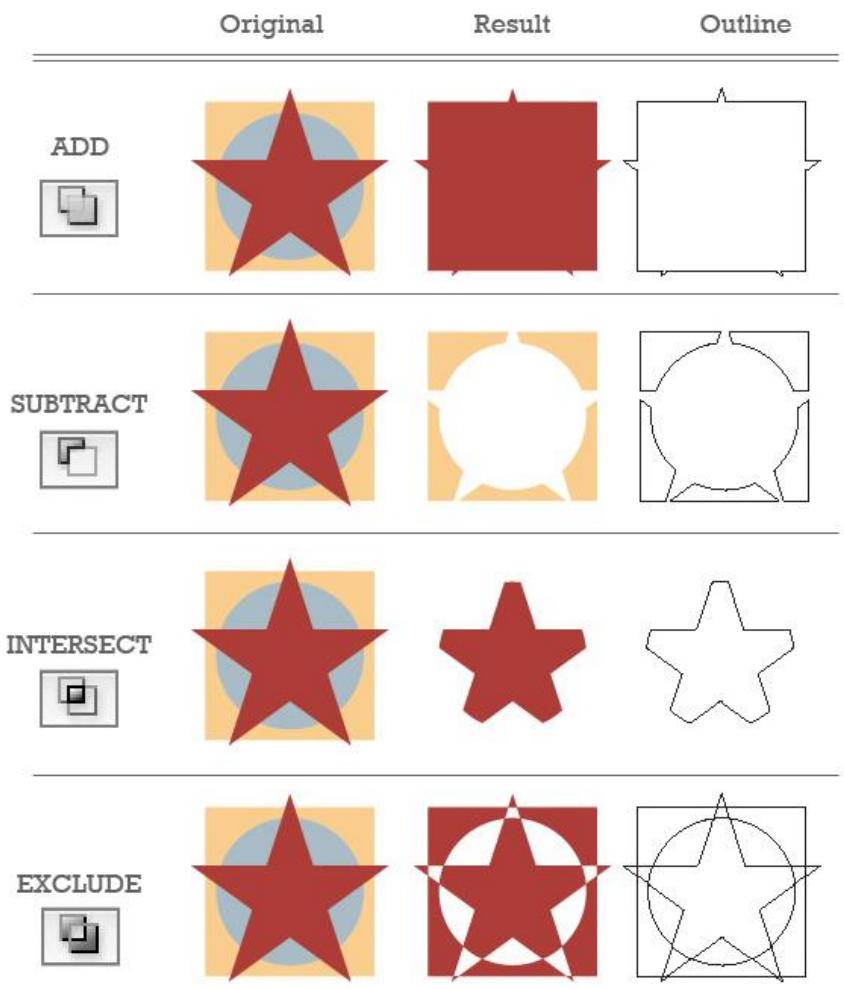
**Add/Unite** - დამატება/გაერთიანება. აერთიანებს ყველა მონიშნულ ობიექტს და ერთ ფიგურად აქცევს. თუ საწყისი ობიექტები სხვადასხვა ფერისაა, საბოლოო ობიექტის ფერი იქნება იმ ობიექტის, რომელიც ყველაზე ზემოდანაა განთავსებული.

**Subtract/Minus Front** - გამოკლება/მინუს წინა. ზემოდან დადებული ფიგურის ფორმის მიხედვით ახდენს ქვემოთ განთავსებული ობიექტებიდან ფორმის გამოჭრას.

**Intersect** - გადაკვეთა. ახალი ფიგურას ქმნის გადაფარვის არეების გამოყენებით. შლის ყველა იმ მონაკვეთს, რომელიც გადაფარვაში არ მონაწილეობს.

**Exclude** - გამორიცხვა. Intersect-ის საპირისპირო მოქმედება ანუ შლის ყველა იმ არეს, სადაც ობიექტები გადაფარავენ ერთმანეთს. ამის შედეგად მიღებული ობიექტები ხშირ შემთხვევაში ნაჭრებად არიან დაყოფილები.

რასაკვირველია, Pathfinder არა მხოლოდ ორი ფიგურის შემთხვევაში შეგიძლიათ გამოიყენოთ. აუცილებლად უნდა გახსოვდეთ ობიექტების განლაგების მიმდევრობა. ქვემოთ მოცემულია სურათი, სადაც გამოყენებულია ორზე მეტი ფიგურა.



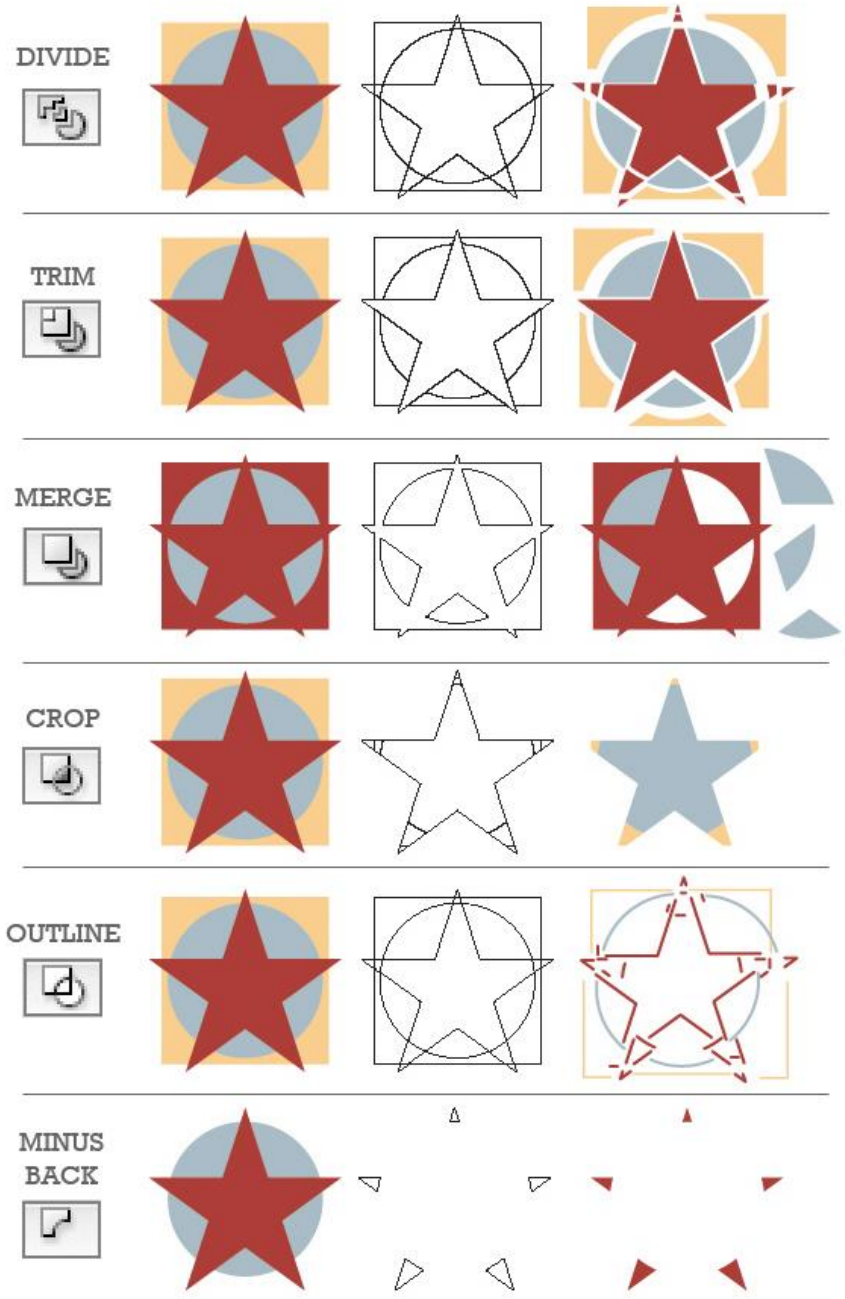

სურ. 2.3-15. Pathfinder-ის მოქმედება სამი ფიგურის გამოყენების შემთხვევაში.

ეხლა კი განვიხილოთ პანელის ქვედა ნაწილი (იხ. სურ. 2.3-16. Pathfinder-ის რეჟიმი).

**Divide** - დაყოფა. Divide ჭრის ფიგურებს ყველგან, სადაც ისინი გადაიკვეთებიან. მიღებული ფიგურები ფერს არ იცვლიან. მიღებული ნაჭრების მონიშვნა და/ან მათი გადაადგილება შესაძლებელია Direct Selection ინსტრუმენტის დახმარებით. თუ მათტან სამუშაოდ გინდათ გამოიყენოთ ჩვეულებრივი მონიშვნის ინსტრუმენტი Selection, მაშინ მიღებული ფიგურა უნდა დაშალოთ: Object > Ungroup.

**Trim** - ჩამოჭრა. მოქმედებს ისევე, როგორც Divide, მაგრამ ამავე დროს შლის გადაფარულ არეებს. თუ ობიექტს კონტურიც გააჩნია, მაშინ Trim კონტურსაც შლის. შედეგად მიღებული ფიგურები ფერს არ იცვლიან.

**Merge** - გაერთიანება. ერთი შეხედვით შეიძლება მოგეჩვენოთ, რომ ბრძანება მუშაობს ისევე, როგორც Trim. მაგრამ ასე არ არის. ის აერთიანებს მხოლოდ იმ არეებს, რომლებშიც ერთი და იგივე ფერია გამოყენებული.




სურ. 2.3-16. Pathfinder-ის რეჟიმი

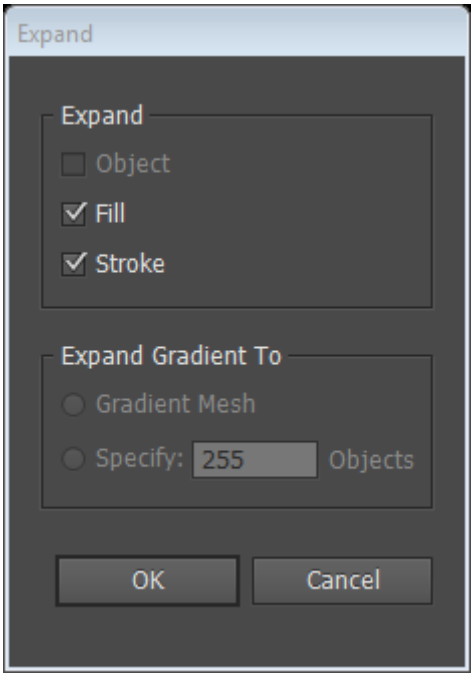
**Crop** - შემოჭრა. ყველაზე ზემოდან დადებულ ფიგურას იყენებს იმისათვის, რომ შემოჭრას ყველაფერი, რაც ამ ფორმის გარეთ მოხვდება. მათ შორის შლის კონტურსაც.

**Outline** - მოხაზულობა. შეიძლება მოგეჩვენოთ, რომ ეს ბრძანება არაფერს არ აკეთებს, მაგრამ თუ დააკვირდებით, რეალურად ეს არის Divide-ის ნაირსახეობა. ფიგურების ნაცვლად ის ყოფს კონტურებს.

**Minus Back** - ქვედას გამოკლება. Minus Front/Subtract საპირისპიროდ მოქმედებს. ყველა ის გადაფარვის არე, რომელიც ყველაზე დაბლა მყოფი ფიგურის ზემოდან იმყოფება - იშლება.

### ობიექტის დაშლა

ვექტორული ობიექტი, შედგება კონტურისა და ფონისგან. ხანდახან არის იმის საჭიროება, რომ ეს ორი კომპონენტი ერთმანეთისგან განცალკევდეს. ამისათვის უნდა გამოიძახოთ Object -> Expand.




აქ მიუთითეთ სასურველი მახასიათებლები.

დაშლის შემდეგ, ობიექტი არის დაჯგუფული და უნდა გაჯგუფოთ (იხ. მუშაობა ობიექტების ჯგუფებთან), რომ იმუშაოთ მის ცალკეულ კომპონენტებთან.

იმ შემთხვევაში თუ ობიექტისთვის გამოყენებული გაქვთ რაიმე ეფექტი, მაშინ მის დასაშლელად ჯერ გამოიყენეთ ჯერ Object -> Expand Appearance, ხოლო შემდეგ ჩვეულებრივი Expand.

### სავარჯიშო

1. გეომეტრიული ფიგურის საფუძველზე შექმენით განსხვავებული ფორმები.
2. გეომეტრიული ფიგურების დახმარებით შექმენით კბილანა.
3. შექმენით კომპანიის ლოგო.
4. შექმენით მექანიკური საათის ციფერბლატი.

## 2.4. ეფექტების გამოყენება

პარაგრაფის შესაბამისი თემატიკა

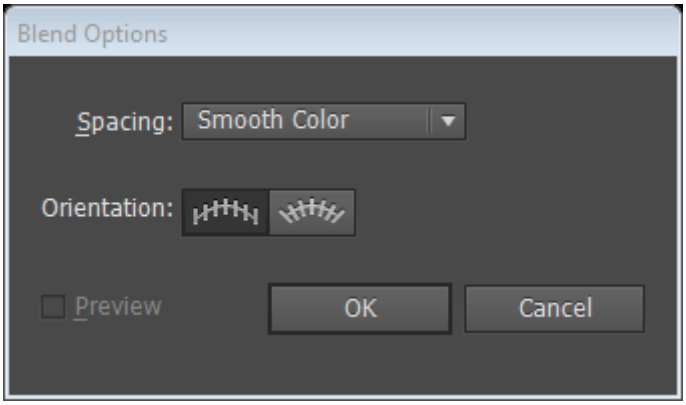
- სხვადასხვა ფორმისა და შეფერილობის ობიექტების ერთმანეთთან შერევა
- შემოჭრისა და გამჭვირვალობის ნიღბების გამოყენება
- ეფექტების გამოყენება
- გრაფიკული სტილების გამოყენება

### შერევა

პროგრამაში შესაძლებელია მოვახდინოთ ობიექტების ერთმანეთთან შერევა. ამისათვის უნდა გამოვიყენოთ ბრძანება Blend (Object -> Blend). შერევის დროს ხდება არა ობიექტების კომბინირება, არამედ ერთი ფორმიდან ან ფერიდან მეორეში გადადინება.

უნდა მონიშნოთ ორი ან მეტი ობიექტი და გამოიძახოთ Object -> Blend -> Make (Alt + Ctrl + B).

შერევის მახასიათებლების სამართავად, გამოიძახეთ Object -> Blend -> Blend Options. მახასიათებლების დაყენება შეგიძლიათ როგორც შექმნამდე, ისე შექმნის შემდეგ.




სურ. 2.4-1. შერევის მახასიათებლების დაყენება.

ობიექტების ერთმანეთთან შერევა ხდება ობიექტის თვისებების - ფორმისა და ფერის - გათვალისწინებით. შერევის სამი ვარიანტია. ჩამოსაშლელი მენიუდან შეგიძლიათ აირჩიოთ ერთ-ერთი.

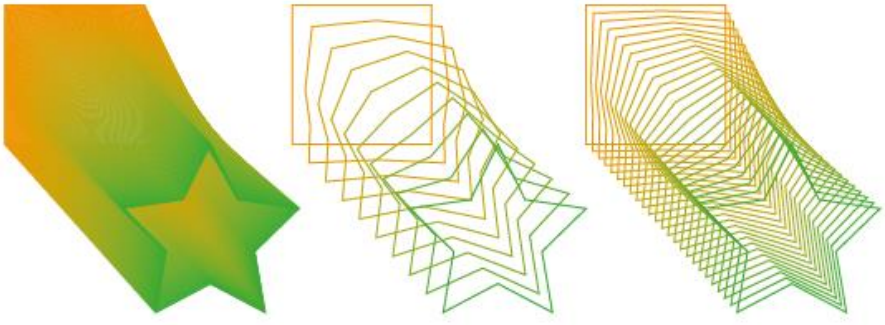
**Smooth Color** - ფერების შერევა.

**Specified Steps** - გარდამავალი ფორმების განსაზღვრული რაოდენობა. უთითებთ, რამდენი გარდამავალი კომპონენტი იქნება მონაწილე.

**Specified Distance** - უთითებთ, რა მანძილი უნდა იყოს გარდამავალ ობიექტებს შორის.

სამივე შემთხვევაში საწყისი ობიექტების ადგილმდებარეობა არ იცვლება.

ქვემოთ მოცემულ მაგალითში გამოყენებულია ორი სხვადასხვა ფორმისა (კვადრატი და ვარსკვლავი) და ფერის (კონტური) ფიგურა.



სურ. 2.4-2. მარცხნიდან მარჯვნივ: Smooth Color, Specified Steps, Specified Distance.

იმისათვის, რომ თითოეული ობიექტზე ცალკე იმუშაოთ გახადოთ, გამოიყენეთ Object -> Blend -> Expand.

### შემოჭრის ნიღაბი

**შემოჭრის ნიღაბი** - ეს არის ობიექტი, რომელიც ნიღბავს სხვა ობიექტს ისე, რომ ხილული რჩება მხოლოდ ის ნაწილი, რომელიც ნიღბის ფარგლებშია მოქცეული ანუ ობიექტი იჭრება ნიღბის ფორმის მიხედვით.

შემოჭრის ნიღაბად შეიძლება გამოყენებული იყოს მხოლოდ ვექტორული ობიექტი (ნებისმიერი ფორმის). საერთოდ კი ნებისმიერი ობიექტის შენიღბვა შეიძლება.

სურ. 2.4-3. სურათი შემოჭრამდე და შემოჭრის შემდეგ.

**შენიშვნა:** *გაითვალისწინეთ, რომ ობიექტი, რომელიც გინდათ ნიღბად გამოიყენოთ, ფენაზე ყველაზე ზემოდან უნდა იყოს განთავსებული. იმის მიუხედავად, რა ატრიბუტები ქონდა შემნიღბავ ობიექტს, ნიღბის შექმნის შემდეგ ის გადაიქცევა ობიექტად კანტისა და ფონის გარეშე.*

ნიღბის შესაქმნელად, მოხაზეთ სასურველი ფორმა ნიღბისთვის, მონიშნეთ შემდეგ ობიექტები და გამოიძახეთ ბრძანება Object -> Clipping Mask -> Make (Ctrl + 7) ან ფენების პანელზე დააწექით შესაბამის ღილაკს (იხ. სურ. 2.2-26, ზ).

ნიღბის დასაშლელად, მონიშნეთ შენიღბული ობიექტი და გამოიძახეთ Object -> Clipping Mask -> Release (Alt + Ctrl + 7) ან ფენების პანელზე დააწექით შესაბამის ღილაკს (იხ. სურ. 2.2-26, ზ).

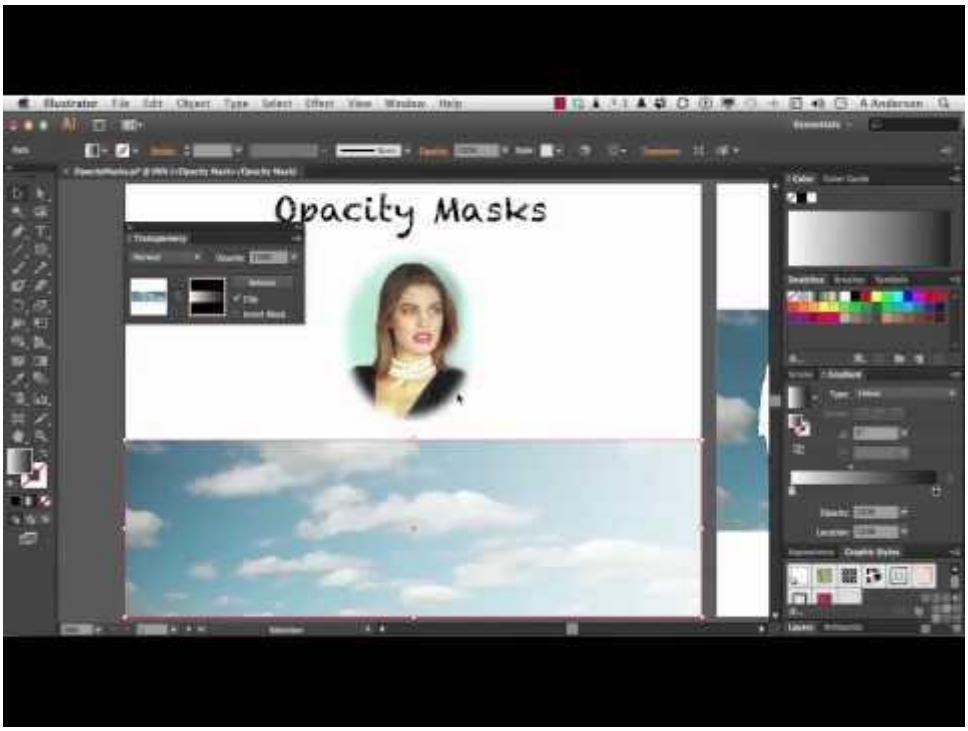
### გამჭვირვალობის ნიღაბი

ამ ნიღბის დახმარებით შეგვიძლია ობიექტების გამჭვირვალობა ვარეგულიროთ. მისი მართვა შესაძლებელია გამჭვირვალობის პანელის დახმარებით Window -> Transparency (Shift + Ctrl + F10).

ამ პანელის დახმარებით შეგიძლიათ მართოთ ობიექტის გამჭვირვალობა და შერევის მეთოდები. ჩვეულებრივი გამჭვირვალობა მართავს მთლიანად ობიექტის გამჭვირვალობა. ნიღაბი კი განსაზღვრავს, გამოსახულების რომელი ნაწილი უნდა იყოს გამჭვირვალე და ასევე გამჭვირვალობის ხარისხს.

გამჭვირვალობის ნიღაბი ნებისმიერი ობიექტის მიმართ შეიძლება იყოს გამოყენებული. ნიღბის იმ ნაწილში, სადაც თეთრი ფერია, გამოსახულება მთლიანად ჩანს; ნიღბის იმ ნაწილში, სადაც შავი ფერია, გამოსახულება მთლიანად დაფარულია. ნაცრისფრის ტონალობა კი განსაზღვრავს გამჭვირვალობის ხარისხს - რაც უფრო მუქი ნაცრისფერია, მით უფრო გამჭვირვალეა ობიექტი.


სურ. 2.4-4. ნიღბის გამოყენების ერთ-ერთი მაგალითი.




ვიდეო 2.4-1. იხილეთ მეტი გამჭვირვალობის ნიღბის შესახებ.

## ეფექტების გამოყენება

ობიექტისთვის რომ გამოიყენოთ ეფექტები, გამოიყენეთ მენიუ Effect ან პანელზე Appearance ღილაკი Fx. შემდეგში გამოყენებული ეფექტი თავსდება ამ პანელზე და მისი რედაქტირება ან წაშლა შეგიძლიათ.

ეფექტების მენიუ გამოყოფილია ორ ნაწილად: ილუსტრატორის და ფოტოშოპის ეფექტებად. ამ უკანასკნელის გამოყენების შემდეგ ვექტორული ობიექტი რასტრულ გამოსახულებად გარდაიქმნება. ფოტოშოპის ეფექტების დასათვალიერებლად გამოიძახეთ Filter Gallery. აქ თითოეული ეფექტისთვის მოცემულია მცირე ზომის მინიატურები, სადაც ნახავთ ეფექტის მოქმედებას და შეგიძლიათ ეფექტის მახასიათებლების ცვლა.

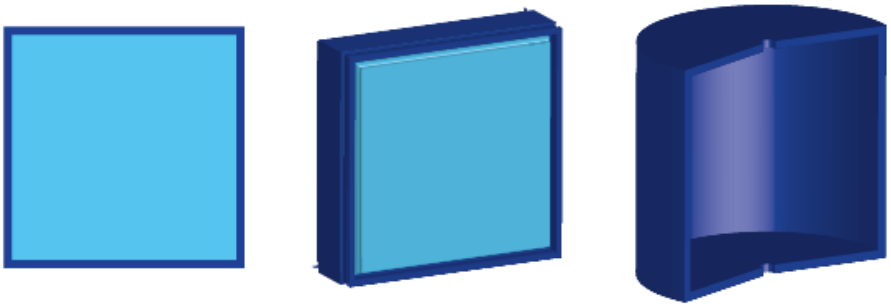
ილუსტრატორის ეფექტებში თქვენ ნახავთ ბევრ ნაცნობ ბრძანებებს, როგორებიცაა Pathfinder ან Warp. ამ შემთხვევაში ეს ბრძანებები გამოიყენება როგორც ეფექტები და ნებისმიერ დროს თქვენ მისი რედაქტირება ან სულაც წაშლა შეგიძლიათ (თუ მიღებული შედეგი არ მოგწონთ).

ეფექტით მიღებული შედეგი ობიექტად რომ გარდაქმნათ, გამოიყენეთ ამისათვის Object -> Expand Appearance.

### *3D ეფექტები*

ამ ეფექტების დახმარებით შეგიძლიათ ორგანზომილებიანი ობიექტი სამგანზომილებიან ობიექტად აქციოთ. აქ არის სამი ბრძანება: Extrude & Bevel, Revolve და Rotate.

სამივე ბრძანების დიალოგური ფანჯარა და ფუნქციები ერთმანეთს გავს, სხვაობა მხოლოდ მიღებულ შედეგშია.



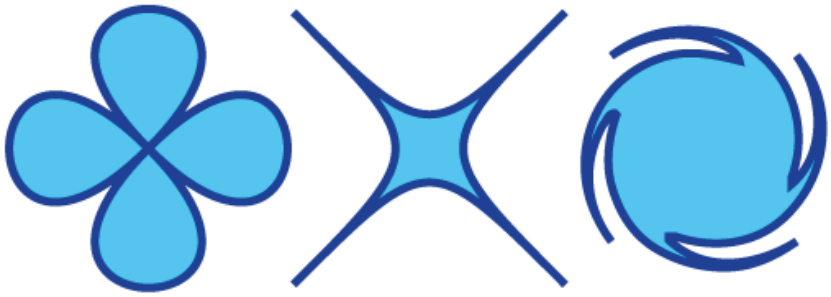
სურ. 2.4-5. სურათზე მოცემული ჩვეულებრივი ფიგურა და გამოყენებული 3D ეფექტები.

### *Convert to Shape*

ამ ეფექტების დახმარებით შეგიძლიათ ობიექტები, რომელიმე გეომეტრიულ ფიგურად გადააქციოთ. აქ არის სამი ბრძანება: Rectangle, Rounded Rectangle და Ellipse.

### *Distort & Transform*

ამ ეფექტების დახმარებით შეგიძლიათ მოახდინოთ ობიექტების დეფორმაცია და/ან მათი ტრანსფორმირება: Free Distort, Pucker & Bloat, Roughen, Transform, Tweak, Twist, Zig Zag.

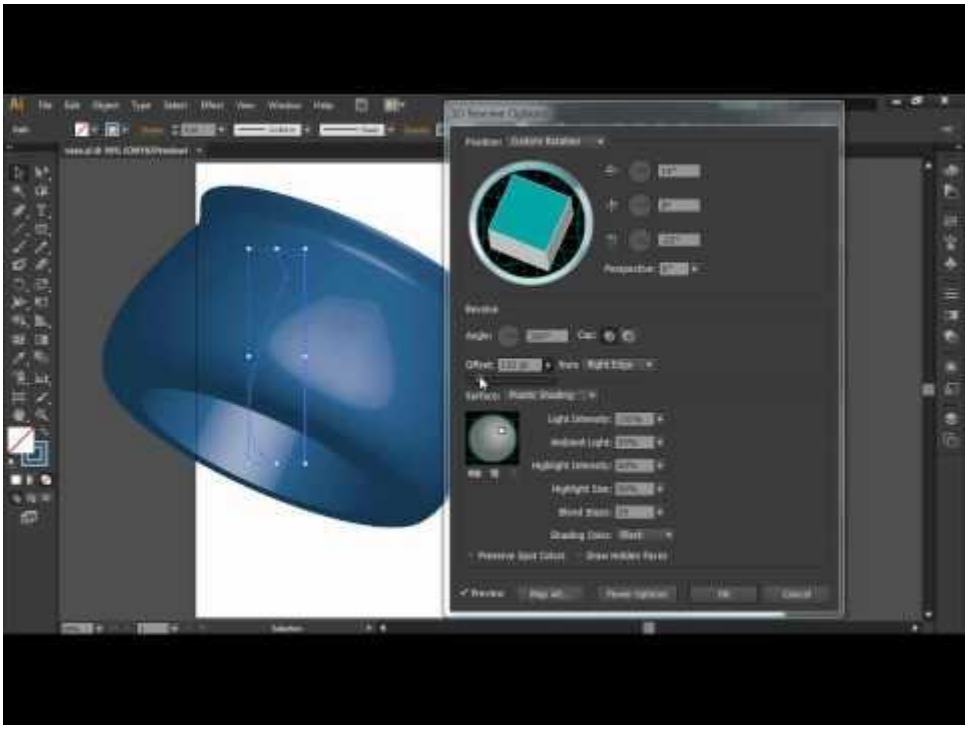


სურ. 2.4-6. პირველ და მეორე გამოსახულებაზე გამოყენებულია ეფექტი Picker & Bloat, მახასიათებლებით +100 და -100 შესაბამისად, ხოლო მესამე გამოსახულებაზე ეფექტი Twist მახასიათებლით 360°

### Stylize

ამ ეფექტების დახმარებით შეგიძლიათ შექმნათ ობიექტისთვის შიდა და გარე ნათება, ჩრდილები, კუთხოვან ობიექტებს მოუმრგვალოთ კუთხეები და სხვა. აქ შედის შემდეგი ეფექტები: Drop Shadow, Feather, Inner Glow, Outer Glow, Round Corners, Scribble.

სურ. 2.4-7. გამოყენებულია ეფექტი Sribble, მახასიათებლებით Tight, Sharp და Sketch.

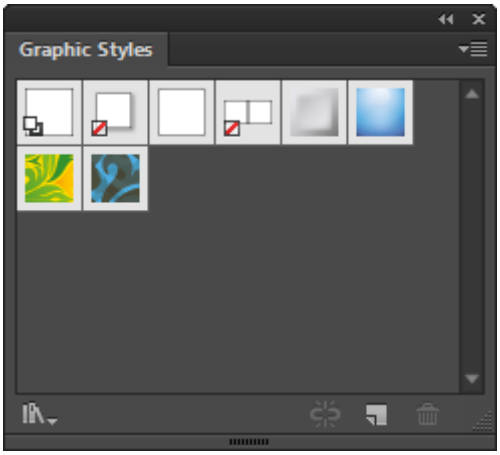


ვიდეო 2.4-2. 3D ეფექტთან მუშაობა ილუსტრატორში.

## გრაფიკული სტილები

გრაფიკული სტილები საშუალებას გაძლევთ სწრაფად შეცვალოთ ობიექტის იერსახე. მაგალითად შეგიძლიათ შეცვალოთ როგორც შევსების, ისე კონტურის სტილი, გამოიყენოთ სამგანზომილებიანი ეფექტი, შეცვალოთ ობიექტის გამჭვირვალობა და ა.შ.

გრაფიკულ სტილებთან სამუშაოდ გამოიძახეთ პანელი Window -> Graphic Styles:

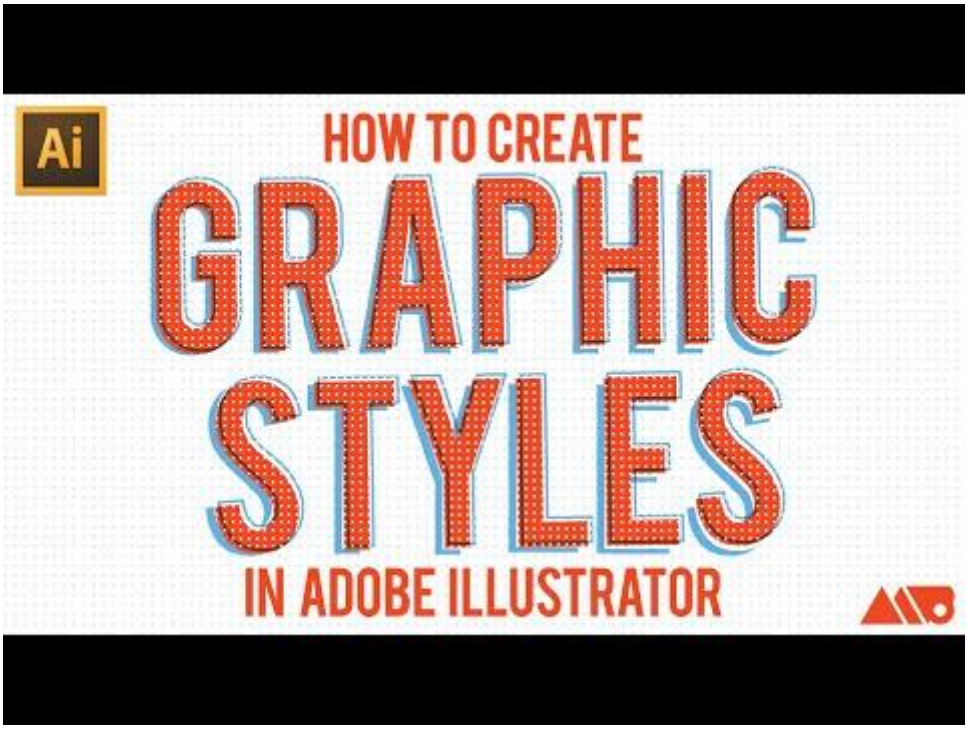



სურ. 2.4-8. გრაფიკულ სტილებთან სამუშაო პანელი.

პროგრამაში ძალიან ბევრი გრაფიკული სტილია. დამატებითი ბიბლიოთეკების გამოსაძახებლად შედით Window -> Graphic Styles Libraries ან ისარგებლეთ ღილაკით, რომელიც გრაფიკული სტილების პანელზე მარცხენა ქვედა კუთხეშია განთავსებული.

გამოსული მენიუდან აირჩიეთ სასურველი ბიბლიოთეკა.

გარდა ამისა, თქვენც შეგიძლიათ შექმნათ ახალი სტილები და დაამატოთ ის გრაფიკული სტილების პანელზე. ამისათვის შექმენით ობიექტი სასურველი შევსებისა თუ კონტურის ფერით, ეფექტებით და გაფორმებებით. შემდეგ მონიშნეთ და პანელის მენიუდან აირჩიეთ New Graphic Style და დაარქვით სახელი; ან მოკიდეთ ობიექტს კურსორი და გადაიტანეთ ის გრაფიკული სტილების პანელზე. ორივე შემთხვევაში ობიექტის გაფორმება განთავსდება, როგორც ახალი გრაფიკული სტილი.




სურ. 2.4-9. ახალი გრაფიკული სტილის შექმნა და დამატება.

## სავარჯიშო

1. შექმენით აბსტრაქტული სახის შპალერი.
2. შექმენით მოცულობითი ჭადრაკის ფიგურა
3. შექმენით ღვინის ბოთლი. გადააკარით მას ეტიკეტი.
4. შექმენით სამგანზომილებიანი ეფექტები ტექსტისთვის.
5. შექმენით საკუთარი გრაფიკული სტილები.

## 2.5. რასტრული გამოსახულების გარდაქმნა ვექტორად

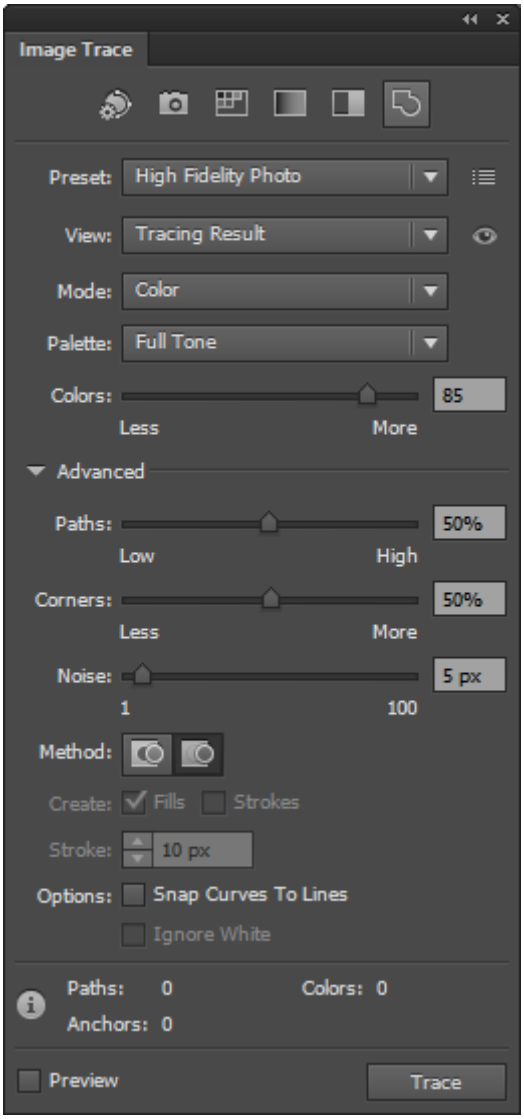
პარაგრაფის შესაბამისი თემატიკა

- რასტრული გამოსახულების ტრასირება.
- ტრასირების მახასიათებლების დაყენება

ვექტორული ობიექტის გადაყვანა რასტრში საკმაოდ მარტივი პროცედურაა, აი პირიქით კი გარკვეულ სირთულეებთან არის დაკავშირებული. არ ხერხდება გამოსახულების იდეალურად გადაყვანა რასტრიდან ვექტორში - ყოველთვის არის ხარვეზები და დანაკარგები.

მოათავსეთ ან გახსენით რასტრული გამოსახულება, რომელსაც გამოიყენებთ, როგორც საწყისს. შემდეგ გამოიძახეთ:

Object -> Image Trace -> Make ან Window -> Image Trace:




სურ. 2.5-1. ტრასირების ფანჯარა. აქედან შეგიძლიათ მართოთ ტრასირების პროცესი და ცვალოთ მისი მახასიათებლები, რომ მიიღოთ სასურველი შედეგი.

შეგიძლიათ აირჩიოთ ტრასირების მზა შაბლონები ან დააყენოთ სასურველი პარამეტრები:

**Presets** - აქ მოთავსებულია ნაგულისხმევი შაბლონები. აირჩიეთ სასურველი გამოსახულების ხასიათიდან გამომდინარე;

**View** - განსაზღვრავს ტრასირებული ობიექტის ხედს. ტრასირებული ობიექტი შედგება ორი კომპონენტისგან: საწყისისგან და შედეგისგან (შედეგი ვექტორული ობიექტია).

**Mode** - შედეგისთის აყენებს ფერების რეჟიმებს (ფერადი, ნაცრისფერი ტონალობის და შავ-ტეთრი).

**Palette** - შედეგის ფორმირებისთვის განსაზღვრავს ფერთა პალიტრას.

**Colors** - განსაზღვრავს, რამდენი ფერი მონაწილეობს გამოსახულების ფორმირებაში.

**Paths** - მართავს გამოსახულების ტრასირების სიზუსტეს. ნაკლები მნიშვნელობა საშუალებას იძლევა უფრო ზუსტი კონტური შექმნას, ხოლო მეტი მნიშვნელობა ქმნის მიახლოებით კონტურს.

**Corners** - განსაზღვრავს კუთხეების რაოდენობას. რაც უფრო მეტია მნიშვნელობა, მით უფრო მეტი კუთხე წარმოიქმნება.

**Noise** - პიქსელებში განისაზღვრება არის ზომა, რომელიც ტრასირების დროს იგნორირდება. მეტი მნიშვნელობის მითითება იწვევს ხმაურის შემცირებას.

**Methods** - ეთითება ტრასირების მეთოდი. პირველ შემთხვევაში ხდება კონტურების ამოჭრდა და შემადგენელი ელემენტების მიბჯენით განთავსება, ხოლო მეორე შემთხვევაში ხდება ობიექტების ერთმანეთზე გადაფარვა.

სურ. 2.5-2. რასტრული ლოგო: ტრასირებამდე და ტრასირების შემდეგ. გამოყენებულია შაბლონი "3 colors", რადგან ლოგოში ძალიან ცოტა ფერია გამოყენებული.

## სავარჯიშო

1. მოიძიეთ კომპანიის რასტრული ლოგო. მოახდინეთ მისი ვექტორში გადაყვანა.
2. მოახდინეთ რასტრული გამოსახულების ვექტორში გადაყვანა
    a. გამოყავით გამოსახულებიდან მხოლოდ ძირითადი კომპონენტები
    b. გადაყვანილი გამოსახულება მაქსიმალურად მიუახლოვეთ ორიგინალს

## 2.6. დოკუმენტის დასაბეჭდად მომზადება

პარაგრაფის შესაბამისი თემატიკა

- დოკუმენტის მომზადება დასაბეჭდად
- ახალი და/ან არსებული დოკუმენტის მომზადება დასაბეჭდად
- ფაილის ამობეჭდვა პრინტერზე. შესაბამისი მახასიათებლების მითითება
- Pdf ფაილის შექმნა. შესაბამისი მახასიათებლების მითითება

პერიოდულად, მუშაობის პროცესში, საჭირო ხდება დოკუმენტის ამობეჭდვა. ამის მიზეზი შეიძლება ბევრი რამ იყოს - ტექსტების კორექტურა/რედაქტირება, კლიენტისთვის ჩვენება (დიზაინის შესათანხმებლად) და სხვა. გარდა ამისა, სამუშაოს დასრულების შემდეგ, დგება საკითხი ნამუშევრის მომზადება სტამბაში გასაგზავნად.

### დოკუმენტის დასაბეჭდად მომზადება

*ახალი დოკუმენტის შექმნა (ბეჭდვის გათვალისწინებით)*

ახალი დოკუმენტის შესაქმნელად, ვიძახებთ File -> New (Ctrl + N). გამოსულ ფანჯარაში, Preset-ში ავირჩიოთ Print. ჩამოსაშლელი მენიუდან ავირჩიოთ ქაღალდის ზომა ან მივუთითოთ სასურველი. Bleed-ში მივუთითოთ 5 მმ (თუ აქტიურია ჯაჭვის ნიშნული, ეს მნიშვნელობა სხვა ველებში ავტომატურად გაიწერება). ბლიდი დოკუმენტს, როგორც წესი, ოთხივე მხრიდან ეთითება, თუმცა შეიძლება იყოს გამონაკლისი შემთხვევაც.

სურ. 2.6-1. წითელი კანტი, ფურცლის გარშემო, გვიჩვენებს ბლიდის საზღვარს.

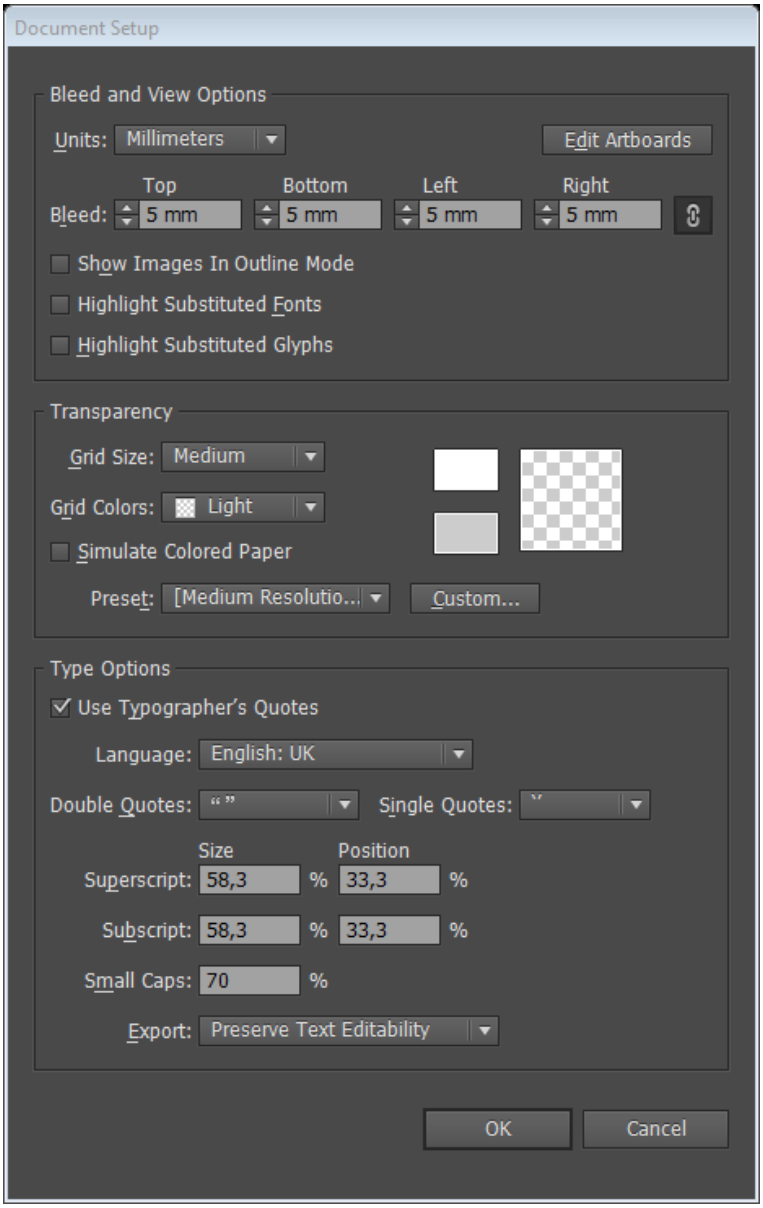
იმ შემთხვევაში, თუ გამოსახულება (ფერი ან სხვა ელემენტი) ბეჭდვისას უნდა ავსებდეს გვერდს ანუ მის კიდემდე უნდა იყოს დაბეჭდილი, მაშინ ასეთ შემთხვევაში ის აუცილებლად უნდა გადიოდეს ბლიდის საზღვრამდე. სტამბაში ბლიდის ზონის ჩამოსუფთავების შემდეგ მიიღებთ ზუსტ ფორმატს (რომელიც დააყენეთ) და ჩამოჭრისას თუ რაიმე ცდომილება იყო, ეს საბოლოო შედეგზე არ აისახება.

დოკუმენტის მახასიათებლების რედაქტირება მუშაობის პროცესის ნებისმიერ ეტაპზე შეიძლება.

ეს შეიძლება იყოს ბლიდების ზომების ან დოკუმენტის სხვა მახასიათებლების ცვლილება.

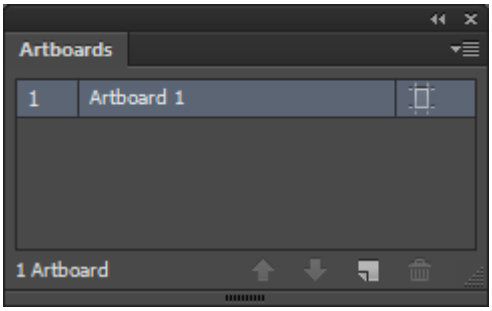
### *შექმნილი დოკუმენტის მახასიათებლების შეცვლა*

მახასიათებლების შესაცვლელად გამოიძახეთ File -> Document Setup (Alt + Ctrl + P).



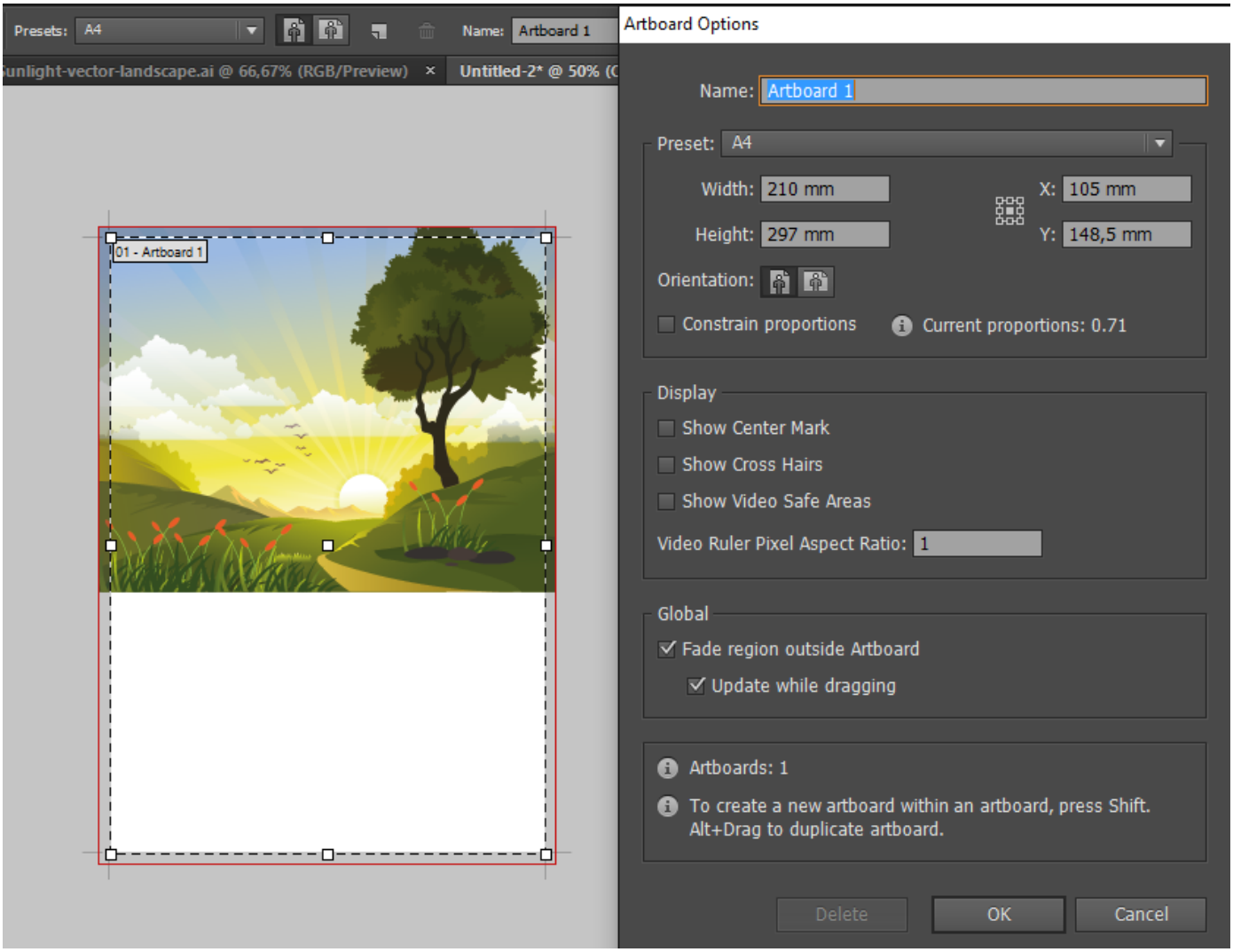

სურ. 2.6-2. აქ შეგიძლიათ ცვალოთ მრავალი მახასიათებელი, მათ შორის საზომი ერთეულები, ბლიდის ზომა, ტექსტის პარამეტრები (მხოლოდ ამ დოკუმენტს შეეხება) და ა.შ. ღილაკი Edit Artboard გაძლევთ საშუალებას შეცვალოთ დოკუმენტის ზომები, მისი ორიენტაცია (ჰორიზონტალური ან ვერტიკალური).

სამუშაო დაფასთან/ფურცელთან სამუშაოდ გამოიძახეთ შესაბამისი პანელი Window -> Artboards:




პანელის დახმარებით შეგიძლიათ დაამატოთ ახალი სამუშაო დაფები ან წაშალოთ უსარგებლო. მოახდინოთ მათი დუბლირება (მაგალითად, როდესაც ერთი და იმავე პოსტერის განსხვავებულ ვერსიებს აკეთებთ).

მონიშნეთ სასურველი სამუშაო დაფა და პანელის მენიუდან აირჩიეთ Artboard Options:



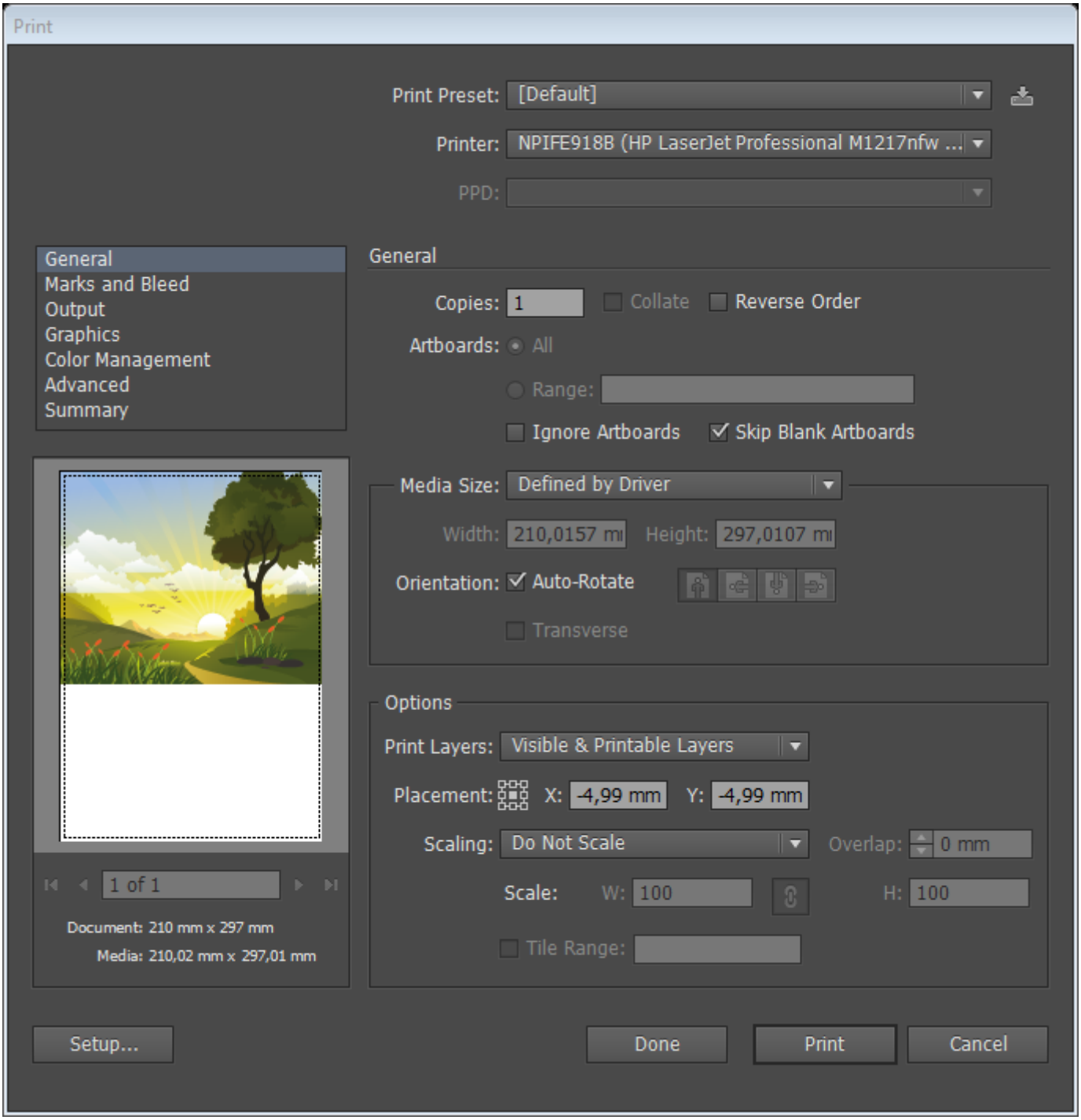

სურ. 2.6-3. სამუშაო დაფის მახასიათებლების შეცვლა.

სამუშაო დაფა მოექცევა ჩარჩოში (წყვეტილი ხაზი), რომელიც გაჩვენებთ მის საზღვრებს. მარცხენა ზედა კუთხეში კი მისი დასახელება და ნუმერაციაა მოთავსებული (სამუშაო დაფები ინომრება, რომ ადვილად მიხვდეთ, რომელი ნამუშევარი, რომელ გვერდზეა განლაგებული). სამუშაო დაფას კუთხეებში აქვს ორ-ორი შავი ხაზი - ეს ჭრის ხაზებია, რომელთა მიმართულება (წარმოსახვითი) ემთხვევა სამუშაო დაფის საზღვრებს. რაც საზღვრებს სცდება, ის ჩამოიჭრება და საბოლოო პროდუქტზე ის არ იქნება ასახული.

ეს მახასიათებლები ასევე შეგიძლიათ ცვალოთ ინსტრუმენტთა პანელზე შესაბამისი ინსტრუმენტის არჩევით - Artboard Tool (Shift + O). გვერდის მახასიათებლები ასეთ შემთხვევაში ჩნდება ფორმატირების ზოლზე:

## ფაილის ამობეჭდვა პრინტერზე

პრინტერზე დასაბეჭდად ვიძახებთ ბეჭდვის ბრძანებას File -> Print (Ctrl + P).




სურ. 2.6-4. დოკუმენტის პრინტერზე გაშვების ფანჯარა.

ამ ფანჯარაში შეგიძლიათ დააყენოთ პარამეტრები, რომელთა გათვალისწინებითაც ის დაიბეჭდება. დაყენებული მახასიათებლები შეგიძლიათ შეინახოთ შაბლონის სახით და საჭიროების შემთხვევაში სწრაფად გამოიძახოთ.

**General** - ეთითება ძირითად მახასიათებლებს, როგორებიცაა, ფურცლის ზომა, მისი ორიენტაცია, ასლების რაოდენობა და სხვა.

**Marks and Bleed** – ეთითება დამატებითი მახასიათებლები, მაგ., ქონდეს თუ არა ჭრის ხაზები, ბლიდების ზომა (თუ ის განსხვავდება დოკუმენტში დაყენებულისგან, შეგიძლიათ აქვე შეცვალოთ), გვერდის შესახებ ინფორმაცია.

**Output** - ეთითება, რა სახით უნდა დაიბეჭდოს დოკუმენტი. ზოგი ფუნქცია მიუწვდომელია იმ შემთხვევაში, თუ პრინტერს არ აქვს შესაბამისი მხარდაჭერა. მაგალითად თქვენ ვერ განახორციელებთ ფერდაშლას (ანუ CMYK-ის შემადგენელ კომპონენტებად დაშლას) თუ პრინტერს შესაბამისი მხარდაჭერა არ აქვს. როგორც წესი ასეთი რამე PostScript ტიპის პრინტერებს შეუძლიათ.

**Graphic** - ეთითება, რა სახით უნდა დაიბეჭდოს გრაფიკული ინფორმაცია - აქცენტი გაკეთდეს ხარისხზე თუ სიჩქარეზე (თუ გრაფიკული ინფორმაცია ძალიან დიდია, ბეჭდვის პროცესი შესაბამისად დროში გაწელილია)

**Color Management** - ფერების პროფილის მართვა. ავტომატურად ყენდება ის ფერთა პროფილი, რომელიც თქვენ სამუშაოდ გაქვთ არჩეული (იხ. ფერების ერთიანი პროფილის გამოყენება). საჭიროების შემთხვევაში შეგიძლიათ აირჩიოთ სხვა (სამიზნე მოწყობილობის შესაბამისი).

**Advanced** - ეთითება დამატებითი მახასიათებლები, მაგალითად, როგორი ხარისხით უნდა დაიბეჭდოს გამჭვირვალე ელემენტები.

**Summary** - აქ თავმოყრილია ყველა ის თვისება, რომელიც თქვენ დააყენეთ. შეგიძლიათ თვალი გადაავლოთ და თუ რამე ცვლილება დაგჭირდებათ, გადახვალთ შესაბამის ფუნქციაზე და შეიტანთ ცვლილებებს.

დაყენებული თვისებები, შეგიძლიათ შეინახოთ შაბლონის სახით, რომ მომავალში არ დაგჭირდეთ ყველა მახასიათებლის ხელახლა დაყენება.

### PDF-ის მომზადება

ფაილის მომზადება დასაბეჭდად (pdf-ის სახით), ძალიან გაქვს ბეჭდვის პროცესს, ოღონდ მცირედი ცვლილებებით. ამ შემთხვევაშიც უნდა იყოს გათვალისწინებული ბლიდები, ფერთა მოდელი (CMYK), ფაილის გარჩევადობა და ა.შ. (იხ. ახალი დოკუმენტის შექმნა (ბეჭდვის გათვალისწინებით).

როდესაც დოკუმენტი ყველანაირად მზად იქნება და გადაწყვიტავთ მის pdf-ის სახით შენახვას, გამოიძახეთ ბრძანება Save As – File -> Save As (Shift + Ctrl + S).

გამოსულ ფანჯარაში (იხ. სურ. 2.6-5. PDF-ის მომზადების ფანჯარა.), აყენებთ ფაილის თვისებებს. დაყენებული თვისებები, შეგიძლიათ შეინახოთ შაბლონის სახით და შემდეგში სწრაფად გამოიძახებთ სასურველ მახასიათებლებს.

**Presets** - ირჩევთ შენახულ შაბლონებს, რის მიხედვითაც მოხდება ფაილის მომზადება. სტანდარტულად აყენია [Illustrator Default]. ჩამოსაშლელი მენიუდან შეგიძლიათ აირჩიოთ სხვა შაბლონებიც. მაგ., თუ გინდათ ფაილი გააგზავნოთ მეილით, აირჩიეთ [Smallest Files Size]. ამ დროს მცირე ზომის ფაილი გამოდის. თუმცა არის შემთხვევები, როდესაც ამ მეთოდითაც მომზადებული ფაილი საკმაოდ დიდი მოცულობის გამოდის.

**Standard** - უთითებთ PDF-ის სტანდარტებს

**Compatibility** - თავსებადობა. უთითებთ Acrobat-ის მინიმალურ ვერსიას, რომელშიც გაიხსნება ეს ფაილი კორექტულად.

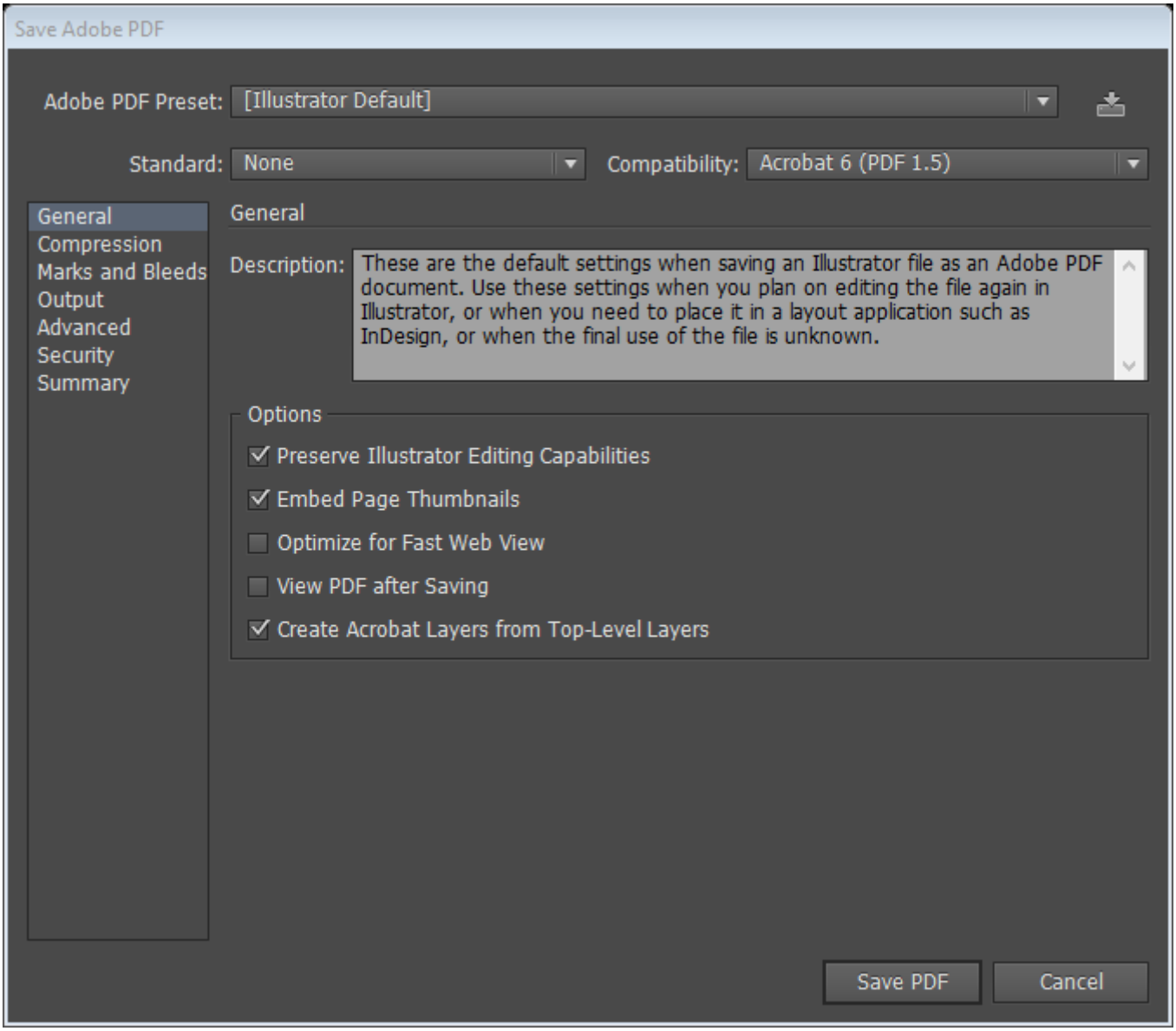
ეს სამი მახასიათებელი შეგიძლიათ ნებისმიერ დროს აირჩიოთ, რომელ ჩანართზეც არ უნდა იმყოფებოდეთ.

სხვა ფუნქციები კი ჩამოთვლილია მარცხენა მხარეს:

**General** - ზოგადი ინფორმაცია ფაილის შესახებ.

**Compression** - გამოსახულებების კომპრესიის მართვა. სამი სახის რასტრული გამოსახულების კომპრესიაზეა საუბარი - ფერადი, ნაცრისფრის ტონალობის და მონოქრომული. შეგიძლიათ დააყენოთ გარჩევადობის ზღვარი და წერტილების ოპტიმალური რაოდენობა. თუ გამოსახულების გარჩევადობა,

დაწესებულ ზღვარზე მაღალი იქნება, მოხდება მისი შემცირება (ამას Downsample ეწოდება) ოპტიმალურამდე.




სურ. 2.6-5. PDF-ის მომზადების ფანჯარა.

**Marks and Bleeds** – ეთითება დამატებითი მახასიათებლებს, მაგ., ქონდეს თუ არა ჭრის ხაზები, ბლიდების ზომა (თუ ის განსხვავდება დოკუმენტში დაყენებულისგან, შეგიძლიათ აქვე შეცვალოთ), გვერდის შესახებ ინფორმაცია.

**Output** - ფერების პროფილის მართვა. ავტომატურად ყენდება ის ფერთა პროფილი, რომელიც თქვენ სამუშაოდ გაქვთ არჩეული (იხ. ფერების ერთიანი პროფილის გამოყენება). საჭიროების შემთხვევაში შეგიძლიათ აირჩიოთ სხვა (სამიზნე მოწყობილობის შესაბამისი).

**Advanced** - ეთითება დამატებითი მახასიათებლები, მაგალითად, როგორი ხარისხით უნდა დაიბეჭდოს გამჭვირვალე ელემენტები.

**Security** - ეთითება დაცვის მახასიათებლები. შეგიძლიათ დოკუმენტზე სხვადასხვა შეზღუდვების დაყენება, მაგ., გრაფიკული ინფორმაცია დაიბეჭდოს დაბალი სახით ან საერთოდ არ დაიბეჭდოს და ა.შ.

**Summary** - აქ თავმოყრილია ყველა ის თვისება, რომელიც თქვენ დააყენეთ. შეგიძლიათ თვალი გადაავლოთ და თუ რამე ცვლილება დაგჭირდებათ, გადახვალთ შესაბამის ფუნქციაზე და შეიტანთ ცვლილებებს.

## სავარჯიშო

1. შექმენით სხვადასხვა სახის საბეჭდი სტილები: სხვადასხვა ფორმატისთვის, ჭრის ხაზების გათვალისწინებით.
2. შექმენით pdf-ის სტილები - მაღალი და დაბალი ხარისხის ფაილების მისაღებად.

# 3. ვებ ინტერფეისის გრაფიკული დიზაინის მომზადება

## 3.1. მოსამზადებელი სამუშაოები

### 3.1.1. კითხვარის (ბრიფის) შედგენა

სწავლის შედეგის შესაბამისი თემატიკა

- ბრიფის ფორმატის გაცნობა
- ბრიფის შემადგენელი ნაწილების განსაზღვრა
- პროექტის ბრიფის შედგენა

“კრეატიული ბრიფი” ან “დიზაინის ბრიფი” არის დოკუმენტი, რომლიც ასახავს დამკვეთის მოთხოვნებს და ორიენტირებულია პროექტის შედეგზე. ამ დოკუმენტის ფორმატი გამოიყენება სხვადასხვა კრეატიულ ინდუსტრიაში; გრაფიკულ დიზაინში, მარკეტინგში, პროდუქტის დიზაინში, არქიტექტურაში და მათ შორის ვებ დიზაინშიც.

ბრიფი როგორც ინსტრუმენტი, ამარტივებს დამკვეთის და შემსრულებლის შორის ურთიერთობას, ვინაიდან მასში ასახულია პროექტის მნიშვნელოვანი დეტალები. ზოგ შემთხვევაში დამკვეთს ბრიფი გამზადებული აქვს, სხვა შემთხვევაში კი, დიზაინერი ეხმარება დამკვეთს ბრიფის შედგენაში. ორივე შემთხვევაში ეს დოკუმენტი ძალიან სასარგებლოა ნაყოფიერი თანამშრომლობისთვის.

ეფექტურად შედგენილი ბრიფი ზოგავს დროს დამკვეთისთვის და შემსრულებლისთვის. ეფექტური ბრიფის შედგენა არის უნარი, რომელიც საჭიროებს პრაქტიკას და გამოცდილებას. ამ თავში ჩვენ

განვიხილავთ ბრიფის სტრუქტურას და ძირითად ელემენტებს, რომელიც აუცილებელია ეფექტური ბრიფის შედგენისთვის.

ვებ დიზაინის შემთხვევაში, ბრიფის სტრუქტურა უნდა შეიცავდეს შემდეგ ინფორმაციას:

1. დამკვეთი
    1.1. ორგანიზაციის დასახელება
    1.2. საკონტაქტო პირის სახელი
    1.3. ტელეფონის ნომერი
    1.4. ელ. ფოსტის მისამართი
    1.5. მოკლე ინფორმაცია ორგანიზაციის საქმიანობის შესახებ
2. პროექტი
    2.1. შესავალი / პრობლემის აღწერა
    2.2. მიზანი / სასურველი შედეგი
    2.3. ანალოგი ვებ საიტების მაგალითები
    2.4. სამიზნე აუდიტორია
3. დავალება
    3.1. ვებ საიტის გვერდები / სტრუქტურა
    3.2. დიზაინის სტილი
    3.3. მისაღები შედეგი
4. პირობები
    4.1. გრაფიკი
    4.2. ბიუჯეტი
    4.3. სხვა პირობები / შეზღუდვები

მაგალითისთვის, განვიხილოთ დამკვეთის მიერ შევსებული ბრიფი, რომელიც შევსებულია ელექტრონული კომერციის ტიპის ვებ საიტის დამზადების მიზნად.

1. **დამკვეთი**

    1.1. **ორგანიზაციის დასახელება**
    სათამაშოების მაღაზიათა ქსელი "პეპელა"

    1.2. **საკონტაქტო პირის სახელი**
    ნოდარ ტაბატაძე

    1.3. **ტელეფონის ნომერი**
    0322 22 65 51

    1.4. **ელ. ფოსტის მისამართი**
    info@pepela.ge

### 1.5. მოკლე ინფორმაცია ორგანიზაციის საქმიანობის შესახებ

სათამაშოების მაღაზიათა ქსელი „პეპელა“, საქართველოს ბაზარზე გაჩნდა 2006 წლიდან. „პეპელა“ მომხმარებელს სთავაზობს ევროპული და საერთაშორისო ბრენდების სათამაშოების ფართო ასორტიმენტს, ბიჭებისათვის და გოგონებისათვის 1 თვიდან 12 წლის ჩათვლით. ქსელში ამჟამად შედის, თბილისის სხვადასხვა უბნებში გახსნილი 5 მაღაზია.

## 2. პროექტი

### 2.1. შესავალი / პრობლემის აღწერა

მიუხედავად მრავალწლიანი გამოცდილების, კომპანიას დღემდე არ გააჩნია ვებ საიტი. ქსელში შემოსული ახალი კოლექციები და ინფორმაცია მიმდინარე ფასდაკლებების შესახებ ვრცელდება მხოლოდ არსებული Facebook-ის გვერდის მეშვეობით, რომელსაც საკმაოდ დიდი აუდიტორია ჰყავს.

მომხმარებელი ითხოვს ვებ საიტს, სადაც შეეძლება სათამაშოების სრული კატალოგის დათვალიერება და პროდუქციის ელექტრონულად ყიდვის საშუალება.

### 2.2. მიზანი / სასურველი შედეგი

“პეპელა”-ს ახალი ვებ საიტის განიხილება როგორც ერთ-ერთი ქსელის ფილიალი, სადაც მომხმარებელს ექნება საშუალება დაათვალიეროს საწყობში არსებული პროდუქციის კატალოგი, შეუკვეთოს სასურველი პროდუქტი და მიიღოს თავისი შენაძენი ადგილზე მიტანით.

ვებ საიტი ასევე შეასრულებს ფასდაკლებებზე და ახალ კოლექციებზე ინფორმაციის გავრცელების ფუნქციას და მომხმარებელს მიაწვდის სხვა საინტერესო და სასარგებლო ინფორმაციას სათამაშოების ინდუსტრიის შესახებ.

### 2.3. ანალოგი ვებ საიტების მაგალითები

საქართველოს ბაზარზე უკვე არსებობს რამდენიმე კომპანია, რომელიც სათამაშოების გაყიდვას ახორციელებს ვებში. მათი გამოცდილება შესაძლოა სასარგებლო იყოს “პეპელა”-ს ახალი ვებ საიტის დამზადებაში.

http://chita.ge/
http://bana.ge/
http://kubiki.ge/

### 2.4. სამიზნე აუდიტორია

ვინაიდან “პეპელა” მოღვაწეობს მხოლოდ საქართველოს ბაზარზე, ვებ საიტი უნდა იყოს გათვლილი ადგილობრივ მოსახლეობაზე.

სათამაშოების ძირითადი მომხმარებელი არის 12 წლამდე ბავშვები, თუმცა სათამაშოების მყიდველი ხშირ შემთხვევაში არის მშობელი ან უფროსი, რომელიც საჩუქარს ეძებს. ვებ საიტის უნდა ითვალისწინებდეს ბავშვების ინტერესს კატალოგის დათვალიერების

პროცესში და ამავდროულად მორგებული უნდა იყოს ონლაინ გაყიდვებზე.

# 3. დავალება

## 3.1. ვებ საიტის გვერდები / სტრუქტურა

- **ზოგადი ელემენტები**
  ვებ საიტის ყველა გვერდზე უნდა ჩანდეს შემდეგი ზოგადი ელემენტები:

  - ძიების ველი
  - ცხელი ხაზის ტელეფონის ნომერი
  - კალათის მანიშნებელი
  - კატალოგის სანავიგაციო მენიუ
  - ვებ საიტის სანავიგაციო მენიუ
  - სოციალური ქსელების პიქტოგრამები

- **მთავარი გვერდი**
  გარდა ზოგადი ელემენტებისა, მთავარ გვერდზე უნდა იყოს გამოქვეყნებული:

  - ქსელში არსებული სხვადასხვა აქციები
  - ახალი სათამაშოები
  - სეზონურად რჩეული პროდუქცია

- **კატეგორიის გვერდი**
  კატალოგიდან რომელიმე კატეგორიის არჩევის შემთხვევაში, მომხმარებელი უნდა ხედავდეს:

  - კატეგორიაში შესული პროდუქციის სიას, დიდი სურათებით
  - პროდუქციის ფასით, ანბანით და პოპულარობით სორტირების საშუალებას
  - კონკრეტული პროდუქტის კალათაში დამატების საშუალებას
  - კონკრეტული პროდუქტის რჩეულებში დამატების საშუალება
  - გვერდების გადართვის საშუალებას

- **პროდუქციის გვერდი**
  კონკრეტული პროდუქტის გვერდზე უნდა ჩანდეს:

  - პროდუქტის ერთი ან მეტი სურათი
  - დასახელება
  - ფასი
  - აღწერა
  - პროდუქტის კოდი
  - კალათაში დამატების საშუალება
  - რჩეულებში დამატების საშუალება

- სოციალურ ქსელებში გაზიარების ღილაკები

- **კალათა**
  კალათის გვერდზე უნდა ჩანდეს კალათაში დამატებული პროდუქციის ჩამონათვალი შემდეგი დეტალებით:

  - პროდუქტის სურათი
  - პროდუქტის დასახელება
  - პროდუქტის კოდი
  - ერთეულის ფასი
  - რაოდენობა (რედაქტირების საშუალებით)
  - პროდუქციის ამოღების საშუალება
  - შერჩეული პროდუქციის ჯამური ღირებულება

- **ძიების შედეგი**
  მომხმარებელს უნდა შეეძლოს საკვანძო სიტყვით ან პროდუქტის კოდით მოახდინოს ძიება კატალოგში. ძიების შედეგის გვერდზე უნდა გამოჩნდეს:

  - კატალოგში ნაპოვნი პროდუქციის ჩამონათვალი, დიდი სურათებით
  - რამდენი შედეგი იყო ნაპოვნი
  - საკვანძო სიტყვის რედაქტირების საშუალება
  - კატალოგის კონკრეტული კატეგორიის მითითების საშუალება
  - პროდუქციის ფასით, ანბანით და პოპულარობით სორტირების საშუალებას

- **მომხმარებლის კაბინეტი**
  იმისათვის რომ მომხმარებელმა შეიძინოს პროდუქცია ვებ საიტზე, საჭიროა რეგისტრაციის გავლა. რეგისტრაციის პროცესში მომხმარებელი უთითებს თავის ავტორიზაციის მონაცემებს და ასევე ადგილზე მიტანის მისამართს.

  რეგისტრირებული მომხმარებლის კაბინეტში ჩანს კალათაში და რჩეულებში დამატებული და შენახული პროდუქცია. ასევე ჩანს მომხმარებლის მიერ წინა შესყიდვები და მათი დეტალები.

- **ტექსტური გვერდები**
  ვებ საიტის სანავიგაციო მენიუში უნდა იყოს დამატებული რამდენიმე ტექსტური სახის გვერდი, როგორიც არის:

  - კომპანიის შესახებ
  - ადგილზე მიტანის მომსახურების პირობები
  - ვებ საიტის გამოყენების პირობები

- **კონტაქტი**
  ამ ეტაპზე კომპანია “პეპელა”-ს აქვს 5 მაღაზია თბილისში. საკონტაქტო ინფორმაციაში უნდა იყოს განთავსებული ყველა მაღაზიის საკონტაქტო

ტელეფონი, მისამართი და მათი ადგილმდებარეობა უნდა იყოს დატანილი Google-ის რუკაზე.

ასევე უნდა ჩანდეს ადმინისტრაციული საკონტაქტო ინფორმაცია, პოტენციური პარტნიორების და პროდუქციის მომწოდებლებისთვის.

### 3.2. დიზაინის სტილი

კომპანია "პეპელა"-ს გააჩნია ბრენდირების სახელმძღვანელო, რომელშიც მოცემულია კომპანიის ლოგო, მისი გამოყენების წესები და ფერთა პალიტრა, რომელიც აუცილებლად უნდა იყოს გათვალისწინებული ვებ საიტის დიზაინის შემუშავების პროცესში.

ამის გარდა, ასევე არსებობს რამდენიმე სასარგებლო ვიდეო რგოლი, პლაკატი და ბანერი, რომლებსაც კომპანია წარმატებულად იყენებდა სხვადასხვა სარეკლამო კამპანიებში. ვებ საიტის დიზაინის სტილი უნდა ითვალისწინებდეს და სარგებლობდეს არსებული დიზაინის ელემენტებს.

### 3.3. მისაღები შედეგი

ვებ საიტის დიზაინის შეთანხმება მოხდება ეტაპობრივად. შემსრულებლისგან მოსალოდნელია შემდეგი შედეგი, ეტაპების მიხედვით.

- ეტაპი პირველი - ვებ საიტის სტრუქტურული ჩონჩხი (wireframe), სადაც ნაჩვენები იქნება შემოთავაზებული ინფორმაციის განლაგების გადაწყვეტილება.

- ეტაპი მეორე - შეთანხმებული ჩონჩხის მიხედვით, დიზაინის სტილის და სხვა პირობების გათვალისწინებით შემუშავებული ვებ საიტის მთავარი გვერდის დიზაინის მაკეტი.

- ეტაპი მესამე - შეთანხმებული მაკეტის მიხედვით, ვებ საიტის ყველა განსხვავებული გვერდის მაკეტი მოწოდებული PSD ფაილებად.

- ეტაპი მეოთხე - ვებ საიტის აწყობის პროცესში პროგრამული ჯგუფის კონსულტაცია, მხარდაჭერა და საჭირო ცალკეული ელემენტების დამზადება.

## 4. პირობები

### 4.1. გრაფიკი

საახალწლო და საშობაო პერიოდი არის ყველაზე მნიშვნელოვანი სათამაშოების ინდუსტრიაში. ამის გამო ვებ საიტი უნდა იყოს დასრულებული და ონლაინ გაყიდვებისათვის მომზადებული დეკემბრის 1 რიცხვამდე.

### 4.2. ბიუჯეტი

პროექტის მაქსიმალური ბიუჯეტი არის 10,000 ლარი, რომელშიც უნდა შედიოდეს ვებ საიტის დიზაინი, პროგრამირება და ელ. კომერციის ფუნქციონალის ინტეგრაცია.

### 4.3. სხვა პირობები / შეზღუდვები

- ვებ საიტის იქნება განთავსებული www.pepela.ge მისამართზე.
- ვებ საიტი უნდა ითვალისწინებდეს კომპანიის არსებულ ბრენდის სტილს.
- ვებ საიტის კატალოგი ავტომატურად უნდა ივსებოდეს არსებული 1C საწყობის მართვის სისტემიდან, შესაბამისად კონკრეტული პროდუქციის გვერდი უნდა ითვალისწინებდეს 1C ბაზაში არსებულ ველებს.
- ელექტრონული კომერციის ფუნქცია უნდა იყოს მორგებული TBC ბანკზე. პლასტიკური ბარათით გადახდის გვერდი უნდა იყოს გაფორმებული ვებ საიტის საერთო სტილში.

მოცემული მაგალითი არის მორგებული ელექტრონული კომერციის ტიპის ვებ საიტს და დამკვეთის კონკრეტულ მოთხოვნებს. ცხადია, სხვადასხვა დამკვეთს სხვადასხვა მოთხოვნები იქნება. აქედან გამომდინარე ბრიფი უნდა იყოს მორგებული ამოცანის სპეციფიკას.

იმ შემთხვევაში თუ დამკვეთს არ აქვს მომზადებული ბრიფი, სასურველია რომ შემსრულებელმა დიზაინერმა თვითონ შეავსოს პროექტისთვის საჭირო ინფორმაცია ბრიფის ფორმატში და შეათანხმოს მიღებული შედეგი დამკვეთთან. ამ შემთხვევაში ბრიფის შევსება ღებულობს კითხვარის / ინტერვიუს სახეს სადაც დიზაინერი დამკვეთს უსვამს წინასწარ მომზადებულ კითხვებს და მიღებული პასუხებით ავსებს ბრიფის ფორმას.

ბრიფი შესაძლოა იყოს გამოყენებული, როგორც შეკვეთის საბუთი. ოფიციალურ ხელშეკრულებასთან ერთად, ბრიფი გამოიყენება როგორც დანართი, სადაც აღწერილია შესასრულებელი სამუშაოს სპეციფიკა და მოსალოდნელი პროექტის შედეგი.

**სავარჯიშო**

მოცემული მაგალითის მიხედვით შეადგინეთ ბრიფი თქვენი შემდეგი პროექტისთვის.

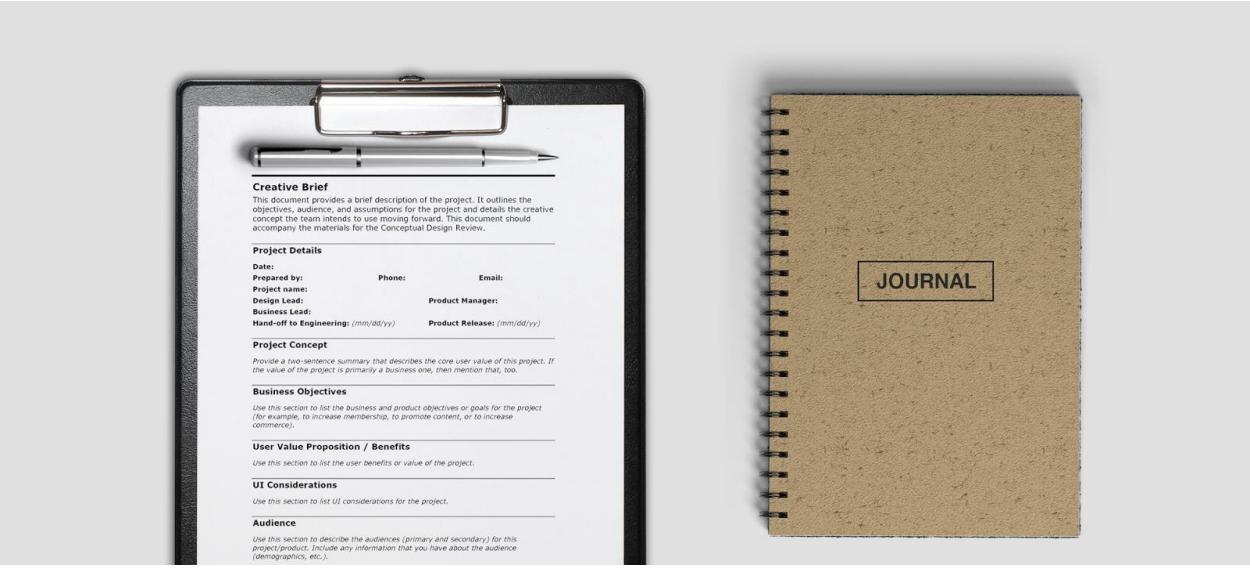
## 3.2. დამკვეთისგან მიღებული ბრიფის ანალიზი

სწავლის შედეგის შესაბამისი თემატიკა

- შევსებული ბრიფის ანალიზი
- მომხმარებლის მოთხოვნების გაცნობა
- კონკურენტების სპეციფიკის გაცნობა

ბრიფის ანალიზი არის პროცესი, რომელიც საჭიროებს ყურადღებას და კონცენტრაციას. გამოტოვებული წინადადება ან ფრაზა შესაძლოა უარყოფითად აისახოს მთლიან პროექტზე. მიუდექით ანალიზის პროცესს, როგორც კომუნიკაციის შესაძლებლობას. დასვით კითხვები, დააზუსტეთ დეტალები, მიიღეთ ყველა საჭირო ინფორმაცია, რომელიც ნათლად არ არის ასახული ბრიფში.

თუ ბრიფის დანიშნულება არის თქვენთვის ინფორმაციის მიწოდება, ბრიფის ანალიზი თქვენთვის უნდა იყოს მიღებული ინფორმაციის ვალიდურობის გადამოწმება და პროექტის შესახებ ყველა შესაძლო კითხვაზე პასუხის მიღება.




**მომხმარებელი**

დიზაინერისთვის, პირველ რიგში მნიშვნელოვანი უნდა იყოს ინფორმაცია თუ ვისთვის და რა დანიშნულებისთვის არის მოფიქრებული კონკრეტული პროექტი. მიუხედავად ბრიფის მოცემული სტრუქტურისა, ეცადეთ პირველ რიგში წარმომიდგინოთ სამიზნე აუდიტორია და შეხედეთ პროექტის სასურველ შედეგს მათი გადმოსახედიდან. დაუსვით თქვენ თავს შემდეგი კითხვები:

- **ვინ არის მომხმარებელი?**
  მომხმარებლის მიახლოებული ასაკი, სქესი, ადგილმდებარეობა დაგეხმარებათ უფრო რეალურად მოარგოთ თქვენი პროდუქტი მომხმარებლის მოთხოვნებს. მომხმარებლის კონტექსტში ასევე საინტერესოა კონკურენცია. მაგალითად, აქვს თუ არა პოტენციურ მომხმარებელს კონკურენტის ან ანალოგი ტიპის ვებ საიტი გამოყენებული და რა გამოცდილება მიიღო?
  შეადგინეთ პოტენციური მომხმარებლის ფსიქოლოგიური პორტრეტი და დიზაინის ყველა

გადაწყვეტილებაზე დაუსვით თქვენ თავს კითხვა თუ რამდენად გასაგები, მისაღები, სასიამოვნო, სწორი იქნება თქვენი გადაწყვეტილება პოტენციური მომხმარებლისთვის.

- **რა გარემოში და ვითარებაში იქნება გამოყენებული მოცემული ვებ საიტი?**
  ვებ საიტის სპეციფიკიდან გამომდინარე, წარმოიდგინეთ მისი მოხმარების პირობები. ეს ინფორმაცია მოგეხმარებათ ფერების, განლაგების არჩევანში და ასევე წარმოდგენას შეგიქმნით მისი მომხმარებლის კონკრეტულ საჭიროებებზე.
  მომხმარებელი სახლშია თუ სამსახურში? პერსონალური კომპიუტერით არის შემოსული ვებ საიტზე თუ პლანშეტური კომპიუტერით? დღის რომელ მონაკვეთშია ვებ საიტის გამოყენების ყველაზე მაღალი ალბათობა?

- **როგორია სრული ვებ საიტის გამოყენების პროცესი?**
  წარმოიდგინეთ ვებ საიტის დანიშნულება როგორც პროცესი. სასურველი შედეგის მისაღწევად რა საფეხურებს გადის მომხმარებელი? მაგალითად, ელექტრონული კომერციის ტიპის ვებ საიტის მომხმარებლის საიტთან ინტერაქციის/ურთიერთობის პროცესი"პროცესი არის:

  1. პროდუქციის მოძებნა / შერჩევა
  2. პროდუქციის ყიდვა
  3. პროდუქციის მიღება

  პროცესის ყველა საფეხურზე წარმოიდგინეთ ვებ საიტის რომელ ელემენტებთან აქვს მომხმარებელს შეხება, აქვს თუ არა გავლილი მსგავსი პროცესები სხვა ვებ საიტებზე და რა გამოცდილება აქვს მომხმარებელს მიღებული.

- **რა არის საბოლოო პროდუქტი?**
  რა ელემენტებისგან შედგება ვებ საიტის, როგორც სრული პროდუქტი და რომელ ელემენტებზე უნდა გაკეთდეს ძირითადი აქცენტი? ეცადეთ დაინახოთ პროდუქტის საერთო სურათი ისე როგორც ამას დაინახავს მომხმარებელი. რა შესაძლებლობები და შეზღუდვები აქვს მას?

**კონკურენცია**

კონკურენტების ან/და ანალოგი ტიპის ვებ საიტების ანალიზი არის კიდევ ერთი მნიშვნელოვანი საფეხური. თუ ბრიფში მოცემული არის კონკურენტების ანა ანალოგი ტიპის ვებ საიტების ბმულები, პირველ რიგში შეისწავლეთ ის მაგალითები.

ვებ სივრცეში გამოქვეყნებულია მილიარდობით სხვადასხვა ვებ საიტი. დიდი ალბათობით არსებობს სხვა ვებ საიტებიც, რომელიც არ იყო მოცემული ბრიფში. გახდით მომხმარებელი, მოძებნეთ ასეთი ვებ საიტები და შეისწავლეთ მათი გადაწყვეტილებები. ჩაინიშნეთ ყველაფერი რაც საინტერესოდ მოგეჩვენებათ. ყველა წვრილმანი დეტალი მნიშვნელოვანია საერთო ანალიზის დროს.

### დამკვეთი

თუ ბრიფი მიღებული გაქვს ახალი ან პოტენციური დამკვეთისგან, გამოიყენეთ ეს შესაძლებლობა და შეისწავლეთ დამკვეთის ორგანიზაციის ან/და პროექტის ისტორია. ნახეთ იგივე ორგანიზაციის მიერ სხვა შესრულებული პროექტები.

### პირობები

გარდა პროექტის ამოცანისა, ბრიფში ასევე მოცემულია პროექტის ვადები და ბიუჯეტი. იმ შემთხვევაში თუ დამკვეთი თქვენგან ელის პროექტის ღირებულების (ფასის) განსაზღვრას/დადგენას. ეს ორი ფაქტორი არის ძალიან მნიშვნელოვანი თქვენთვის, რათა სწორად სწორედ შეაფასოთ და განსაზღვროთ შესასრულებელი სამუშაოს მოცულობა და განხორციელების პროცესი შესასრულებელი სამუშაო.

ბრიფში მითითებული ვადები, ხშირ შემთხვევაში, არის დაკავშირებული სხვა მოვლენებთან. მოცემული ვადის გადაცილება თქვენს მხრიდან, პირდაპირ კავშირში იქნება ყველა სხვა პროცესებთან და შესაბამისად აუცილებლად უარყოფითად აისახება თქვენს ურთიერთობაზე დამკვეთთან. თუ მოცემული გრაფიკის პირობები მიუღებელია თქვენთვის, გაესაუბრეთ დამკვეთის წარმომადგენელს ამ საკითხთან დაკავშირებით და თუ პირობები არ/ვერ შეიცვლება, ნუ აიღებთ იმ პროექტის შესრულების პასუხისმგებლობას.


უმეტეს შემთხვევაში, პროექტი ბიუჯეტი არ იქნება მითითებული ბრიფში, ვინაიდან დამკვეთი ყოველთვის ცდილობს შემსრულებლისგან მიიღოს პროექტის ბიუჯეტის შეფასება და შემდგომ მიღებული ინფორმაციის მიხედვით იღებს თანამშრომლობის გადაწყვეტილებას. ასეთ შემთხვევებში თქვენ უნდა განსაზღვროთ თანამშრომლობის შესახებ პროექტის ბიუჯეტი და უნდა შესთავაზოთ დამკვეთს თქვენი წინადადება.

ნებისმიერი დამატებითი პირობა, რომელიც შესაძლოა იყოს ნახსენები ბრიფში უნდა იყოს გათვალისწინებული როგორც დამატებითი სამუშაო და დამატებითი ადამიანური რესურსის გამოყენების საჭიროება. გაითვალისწინეთ ეს ფაქტორის ბრიფის ანალიზის დროს.

**სავარჯიშო**

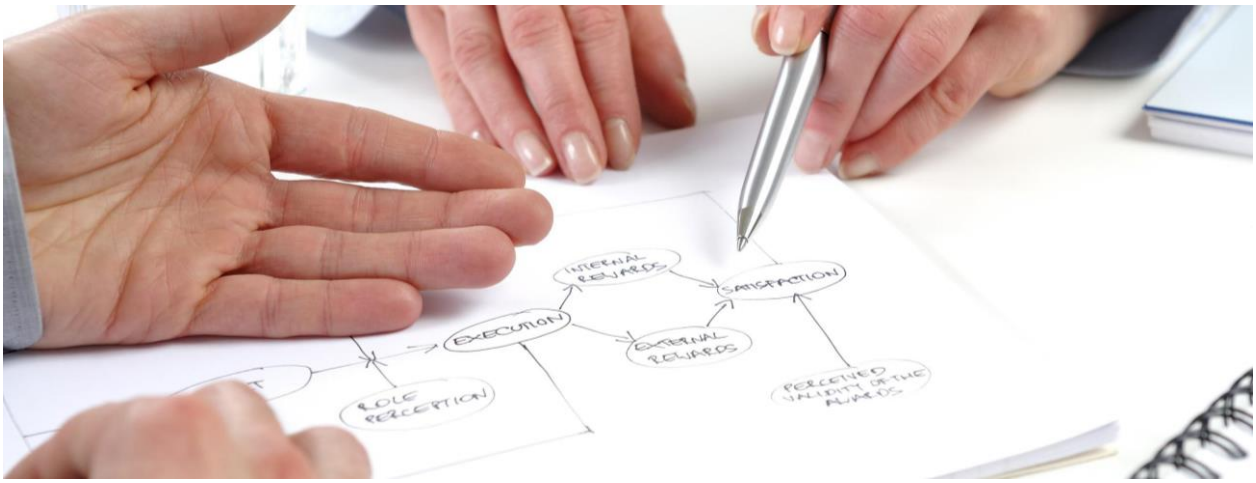
მოახდინეთ წინა დავალებაში შედგენილი ბრიფის ანალიზი.

## 3.3. სამუშაო ეტაპების დაგეგმარება

სწავლის შედეგის შესაბამისი თემატიკა

- პროექტის შესრულებისთვის მასალების განხილვა
- სტრუქტურის / პროტოტიპის შემუშავება
- მაკეტის შემუშავება
- პროდუქტის შეფუთვა

სამუშაო პროცესი ბევრად უფრო ეფექტურია და სწრაფი, თუ წინასწარ გაქვს დაგეგმილი ყველა საფეხური და ყველა საჭირო მასალა მომზადებული გაქვს. საბოლოო შედეგიც უფრო ხარისხიანი გამოდის, ვინაიდან დაგეგმილ პროცესში ძალიან დაბალია იმის ალბათობა, რომ რაიმე გამოგრჩეთ ან დაგავიწყდეთ. ამის გარდა, ნებისმიერი ტიპის დიდი ამოცანის შესრულება ბევრად უფრო მარტივი ხდება თუ ამოცანას პატარა ქვე-ამოცანებად დავყოფთ.




1. **დავალების ანალიზი**
   ბრიფიდან მიღებული ინფორმაციის საფუძველზე ვებ დიზაინერს აქვს ჩასატარებელი საანალიზო და კვლევითი სამუშაო. ამ ეტაპზე დიზაინერი შეისწავლის და განიხილავს დამკვეთთან ბრიფში მითითებულ დეტალებს. ამასთან ერთად იკვლევს და შეისწავლის ანალოგი ტიპის ვებ საიტებს და არსებულ გადაწყვეტილებებს.

   გაანალიზებული ინფორმაციისა და ჩანიშნული კომენტარების საფუძველზე ვებ დიზაინერს უკვე შექმნილი აქვს დაახლოებითი წარმოდგენა თუ რა ტიპის სამუშაო აქვს შესასრულებელი და შეუძლია წარადგინოს მისთვის საჭირო მასალების ნუსხა.

2. **საჭირო მასალები**
   ვებ დიზაინერის ძირითადი მოვალეობა არის, უკვე არსებული მასალების ეფექტურად და ესთეტიკურად განლაგება. იმისათვის რომ ვებ დიზაინერმა შეასრულოს თავისი სამუშაო ხარისხიანად, მას ხელთ უნდა ჰქონდეს ყველა საჭირო მასალა. ამ მასალების ნაწილი ვებ

დიზაინერს უნდა მიაწოდოს დამკვეთი. მაგალითად:

1. საიტის რუკა / გვერდების ჩამონათვალი
2. ორგანიზაციის ლოგო
3. ორგანიზაციის ფერთა პალიტრა
4. ორგანიზაციის პირადი შრიფტი
5. ვებ საიტზე გამოსაქვეყნებელი სხვადასხვა ტექსტური ინფორმაცია

არის შემთხვევები როდესაც დამკვეთს არ აქვს მომზადებული სურათები და საჭიროა მათი შეძენა. არის შემთხვევები, როდესაც ვებ საიტი ასევე საჭიროებს რამდენიმე პარტნიორი ორგანიზაციის ლოგოების გამოქვეყნებას და დამკვეთს არ აქვს მათი ლოგოები. ყველაზე ხშირი არის შემთხვევები, როდესაც დამკვეთს არ აქვს მომზადებული ტექსტური ინფორმაცია, რომელიც ვებ საიტის გვერდებზე უნდა იყოს გამოქვეყნებული.

დაგეგმარების ერთ-ერთი მნიშვნელოვანი ელემენტი არის საჭირო მასალების დროული იდენტიფიცირება და მათი მოპოვება. გაითვალისწინეთ, მასალების მოპოვება, არის დროსთან დაკავშირებული პროცესი. წინასწარ გაანალიზებული მასალების ნუსხა დაგიზოგავთ დროს და არ შეგაფერხებთ მუშაობის პროცესში.

3. **სტრუქტურის / პროტოტიპის შემუშავება**

სტრუქტურა (wireframe) არის ვებ საიტის რომელიმე კონკრეტული გვერდის შავ-თეთრი მონახაზი. ამ თემას სიღრმისეულად გავივლით მომდევნო თავში. სამუშაო პროცესში კი, სტრუქტურის შემუშავება არის საჭირო იმისათვის, რომ სწრაფად მოხდეს ყველა საჭირო ელემენტის გათვალისწინება და ასახვა მონახაზში და ამ განლაგების შეთანხმება დამკვეთთან

ერთი შეხედულებით, ეს ეტაპი შესაძლოა მოგვეჩვენოს როგორც ზედმეტი საფეხური, მაგრამ სამუშაო პროცესში, სადაც რამდენიმე ადამიანი არის ჩარეული ჩონჩხი ასრულებს სწორი

ვიზუალური კომუნიკაციის და დროის დაზოგვის ფუნქციას. სამუშაო დაგეგმარებისას, გაითვალისწინეთ ამ საფეხურისთვის საჭირო დრო და საერთო ჯამში პროექტი ბევრად უფრო ეფექტურად შესრულდება.

4. **მაკეტის შემუშავება**
   მიღებული მასალების და შეთანხმებული ჩონჩხის საფუძველზე, ვებ დიზაინერი აწყობს საბოლოო მაკეტს. მაკეტი არის ვებ საიტის ვიზუალური გამოსახულება, რომელიც ჯერ არ ფუნქციონირებს, უბრალოდ ვიზუალურად ასახავს იმას თუ როგორ გამოიყურება ვებ საიტი მთლიანობაში.

   მაკეტის შემუშავება არის შრომატევადი პროცესი. ზუსტად ამ ეტაპის გამარტივების გამო ჩვენ ვასრულებთ ყველა მოსამზადებელ სამუშაოს. ვაგროვებთ ყველა საჭირო მასალას, ვამზადებს მონახაზებს და ვსწავლობთ ბრიფში მოცემულ ინფორმაციას. მაკეტში ცვლილებების შეტანაც ძალიან რთულია, ამიტომ ვებ დიზაინერმა უნდა ეცადოს მაკეტი ააწყოს რაც შეიძლება სწორად რომ დამკვეთმა არ მოითხოვოს ბევრი დეტალის შეცვლა.

5. **პროდუქტის შეფუთვა**
   ნაშრომის გადაცემა არის სამუშაო პროცესის ერთ-ერთი ეტაპი, ვინაიდან ვებ დიზაინის შემთხვევაში დიზაინის ფაილი არ არის საკმარისი იმისათვის რომ ვებ საიტის ფუნქციონირებდეს. თქვენი დიზაინის მიხედვით კიდევ ბევრი საფეხური არის გასავლელი, მანამ მომხმარებელი შემოვა დასრულებულ ვებ საიტზე და დაიწყებს მის გამოყენებას.

   დამკვეთის ან/და სხვა სამუშაო გუნდის წევრების პროცესების გათვალისწინებით, ვებ დიზაინერმა უნდა გადასცეს თავისი ნაშრომი მაქსიმალურად ეფექტურად, რათა არ დაირღვეს და არ შეფერხდეს პროექტის შემდგომი საფეხურების პროცესი.

   ყველა პროექტი განსხვავებულია და ჩაბარების მოთხოვნები შესაბამისად შესაძლოა რომ განსხვავებული იყოს, მაგრამ ელემენტარულ დონეზე ვებ დიზაინერმა სწორად უნდა დაახარისხოს ფაილები დასახელებები, დიზაინის ფაილებში სწორად გადაალაგოს და დააჯგუფოს ფენები. საჭიროების შემთხვევაში მოამზადოს ინტერფეისის სტილის ნაკრები (UI Kit).

6. **მხარდაჭერა**
   უმეტეს შემთხვევაში როცა ვებ საიტის დიზაინი არის დასრულებული და გადაცემული დამკვეთისთვის ან სამუშაო გუნდის სხვა წევრებისთვის, ჩნდება წვრილმანები რომელიც საჭიროებს დაზუსტებას ან დიზაინის დეტალები რომელიც აკლია საერთო სურათს.

   ვებ დიზაინერმა უნდა დაგეგმოს დამატებითი დრო, რომელიც აუცილებელია მთლიანობაში შემუშავებული პროექტის მაღალი ხარისხის მისაღწევად. ამ დროში დიზაინერი ეხმარება პროგრამისტების ჯგუფს სხვადასხვა საკითხების დაზუსტებაში და პარალელურად ამზადებს ვებ საიტის პატარა დიზაინის ელემენტებს, რომელების საჭიროება სამუშაო პროცესში გაჩნდა.

   ვებ დიზაინერის სამუშაო დასრულებულია მხოლოდ მაშინ, როცა ვებ საიტის შექმნა ბოლომდე დამთავრებულია, დამკვეთმა გადაამოწმა შედეგი და კმაყოფილია მიღებული შედეგით.

**სავარჯიშო**

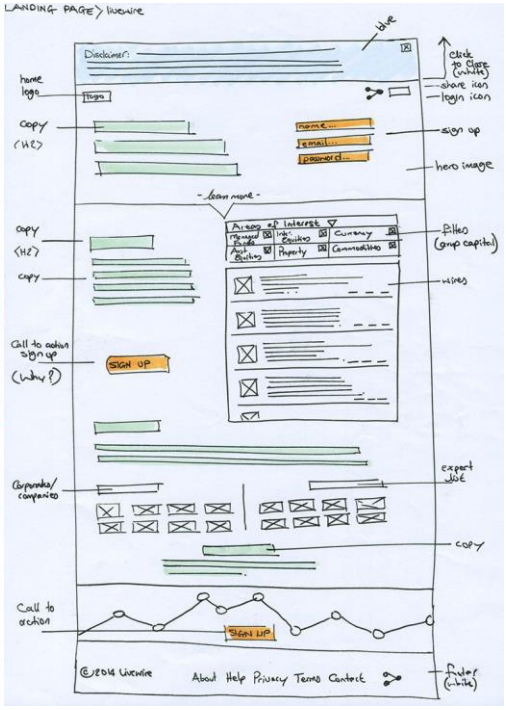
მოახდინეთ წინა დავალებებში შედგენილი და გაანალიზებული ბრიფის საფუძველზე პროექტის დაგეგმარება.

## 3.4. . სტრუქტურა და პროტოტიპი

### 3.4.1. ინფორმაციის არქიტექტურა

<u>სწავლის შედეგის შესაბამისი თემატიკა</u>

- მომხმარებლის მოთხოვნების გათვალისწინება
- პერსონების ჩამოყალიბება
- სცენარების ჩამოყალიბება
- გვერდის ტიპების ჩამოყალიბება
- მომხმარებლის დინების პროცესის ჩამოყალიბება


ვებ სივრცეში პოპულარულია გამოთქმა “კონტენტი მეფეა!”. მომხმარებელს არ სჭირდება უბრალოდ ლამაზი ვებ საიტი - მომხმარებელს სჭირდება მისთვის სასარგებლო ინფორმაცია (ე.წ. კონტენტი), მიწოდებული მაქსიმალურად ეფექტური ხერხით.

ვებ საიტი მომხმარებლის მიზნებზე უნდა იყოს ორიენტირებული. ამიტომ, მანამ დავიწყებთ ვებ საიტის დიზაინის აწყობას, პირველ რიგში უნდა დავგეგმოთ და დავალაგოთ მომხმარებლისთვის მისაწოდებელი ინფორმაცია.

როდესაც მომხმარებელი ხვდება ვებ საიტის რომელიმე გვერდზე, მას უჩნდება ოთხი ძირითადი კითხვა. ჩვენი ამოცანა იქნება, ეფექტურად გავცეთ ამ კითხვებზე პასუხები:

1. სწორ ადგილას მოვხვდი?

2. არის თუ არა აქ ის, რასაც ვეძებდი?

3. არის თუ არა აქ რაიმე უკეთესი?

4. რა არის ჩემი შემდეგი ნაბიჯი?

ინფორმაციის არქიტექტურის შემუშავებისას ჩვენ სწორად დავალაგებთ ვებ საიტზე გამოსაქვეყნებელ ინფორმაციას და მაქსიმალურად დავაახლოებთ მომხმარებლის მიზნებს, ვებ საიტის მფლობელის მიზნებთან.

არქიტექტურის შემუშავების პროცესში, ასევე მივიღებთ პასუხებს წვრილმან კითხვებზე, მაგალითად: ძიების შედეგის გვერდზე, პროდუქცია ანბანის მიხედვით უნდა დალაგდეს თუ ფასის მიხედვით?

ვებ საიტის არქიტექტურა შედგება სხვადასხვა ელემენტებისგან, როგორიც არის ვებ საიტის რუკა, სტრუქტურა, დიზების დიაგრამები და სხვა. ამ თავში დეტალურად განვიხილავთ ამ ელემენტებს.

### მომხმარებელი

ინფორმაციის არქიტექტურის შემუშავების პროცესში, კრიტიკულია ვიცოდეთ პოტენციური მომხმარებლის შესახებ რაც შეიძლება მეტი დეტალი. წინა თავში ჩვენ განვიხილეთ ბრიფის შედგენის და ანალიზის საკითხი, სადაც ნახსენები იყო სამიზნე აუდიტორია. გარდა ბრიფიდან მიღებული ინფორმაციისა, ჩვენ ასევე გვაინტერესებს პასუხები შემდეგ კითხვებზე:

- ვინ არის ჩვენი ვებ საიტის პოტენციური მომხმარებელი?
- რა პრობლემას უგვარებს ჩვენი ვებ საიტი პოტენციურ მომხმარებელს?
- რა მიზნები ან სარგებელი აქვს მომხმარებელს ჩვენი საიტის გამოყენებისას?
- აქვს თუ არა მომხმარებელს გამოცდილება ანალოგი საიტების გამოყენებაში?

მოძებნეთ თქვენს გარემოში პოტენციური მომხმარებელი და გამოკითხეთ. მიიღეთ პასუხები თქვენს კითხვებზე და ჩაინიშნეთ. გამოკითხვის შედეგად მიღებული პასუხები გამოგვადგებათ პერსონების ჩამოყალიბებაში.

### პერსონები (Personas)

ვებ საიტი კონკრეტული ტიპის მომხმარებლისთვის არის და არა ყველა ტიპის მომხმარებლისთვის. მომხმარებლების გამოკვლევის პროცესში, დიდი ალბათობით, თქვენ აღმოაჩენთ პოტენციური მომხმარებლების რამდენიმე ჯგუფს. ყველა ჯგუფს საერთო მიზნები აქვთ, თუმცა რაღაცით მაინც განსხვავდებიან ერთმანეთისაგან.

პერსონები, ვებ საიტის პოტენციური მომხმარებელთა ჯგუფების ფიქტიური წარმომადგენლები არიან. ფიქტიური პერსონების ვიზუალიზაცია და მათი პროფილების ჩამოყალიბება დაგეხმარებათ

მომხმარებლების მოთხოვნების უკეთ გათვალისწინებაში და ასევე გუნდურად პროექტის სხვადასხვა დეტალების განხილვაში.

მაგალითისთვის, განვიხილოთ პერსონის პროფილი, რომელიც ელექტრონული კომერციის ტიპის ვებ საიტისთვის იყო შედგენილი.

1. **სახელი, გვარი და ფოტო**

ნათია ჯალაბაძე
მომხმარებლის სახელი და ფოტო სასარგებლოა ასოცირებისათვის. სამუშაო ჯგუფში პერსონას სახელით როცა ახსენებთ, განხილვის საკითხი უფრო პირადი ხდება და ქვეცნობიერად ყველა ჯგუფის წევრი ცდილობს უკეთესი გადაწვეტილება მოუფიქროს მომხმარებელს.
2. **მომხმარებლის ტიპი** დედა

*ვინაიდან საბოლოო შედეგად ჩვენ რამდენიმე პერსონა გვეყოლება, სასურველია ყველა პერსონას ჰქონდეს მომხმარებლის ჯგუფის ტიპი მითითებული. ელექტრონული კომერციის ტიპის ვებ საიტის შემთხვევაში, მაღაზია "პეპელა"-ს, რომელიც სათამაშოებით ვაჭრობს, შესაძლოა რამდენიმე ტიპის მომხმარებელი ესტუმროს. მათ შორის "დედა" არის ერთ-ერთი ყველაზე პოპულარული ტიპის მომხმარებელი, რომელსაც თავისი მოთხოვნები გააჩნია.*

3. **თანამდებობა, შემოსავალი**
ბანკის ფილიალის მენეჯერი, 2000 ლარი თვეში

*მომხმარებლის თანამდებობა და შემოსავალი გვეხმარება მომხმარებლის უკეთ გაცნობაში და შემდეგი ტიპის კითხვებზე პასუხების მიღებაში: რამდენად დაკავებულია მომხმარებელი კვირის სამუშაო დღეებში და საათებში? აქვს თუ არა წვდომა კომპიუტერთან დრის განმავლობაში? რამდენად გამოცდილი მომხმარებელი ინტერნეტით პროდუქციის შეძენაში? ისარგებლებს თუ არა ადგილზე მიტანის მომსახურებით დამატებითი გადასახადის შემთხვევაში? და ა. შ.*

4. **მოტივაცია**
ნათია 5 და 7 წლის ბიჭების დედაა. ყოველ საღამოს, სამსახურიდან სახლში როცა ბრუნდება ბიჭები კარებთან ხვდებიან და ერთიდაიგივე კითხვას უსვამენ "რა მოგვიტანე?". ნათია დღის განმავლობაში დაკავებულია და ვერ ახერხებს მაღაზიებში გასვლას, ამიტომ ყოველ პარასკევს ყიდულობს საბავშვო წიგნებს და სხვა სათამაშოებს ვებ საიტის მეშვეობით. სარგებლობს ადგილზე მიტანის მომსახურებით ხან სამსახურში და ხან სახლში.

*მომხმარებლის მოტივაციის ფაქტორებიდან ძალიან სასარგებლო ინფორმაცია მივიღეთ. გარდა სხვა ფაქტორებისა, შეგვიძლია გავითვალისწინოთ, რომ ვებ საიტის გამოყენების პროცესში, მომხმარებლისთვის ძალიან სასარგებლო იქნება პროდუქციის ფილტრაცია სქესის და ასაკის მიხედვით. ასევე, შესაძლოა სასარგებლო იყოს პროდუქციის ძიება ფასის ლიმიტის მითითებით.*

5. **მიზნები**

- პროდუქციის ხელმისაწვდომი ფასები
- სხვადასხვა მისამართებზე პროდუქციის მიტანის მომსახურება
- ლოიალურობის ფასდაკლებები

*კვლევის და მოტივაციის ანალიზის შედეგად ვღებულობთ ფაქტორებს, რომლის მიღწევისას გვეყოლება ბედნიერი მომხმარებელი.*

6. **შემაფერხებელი ფაქტორები**

- პროდუქციის მწირი აღწერა
- პროდუქციის კატალოგში არჩევანის სიმცირე
- სურათების დაბალი ხარისხი

*პროდუქციის მაღალი ხარისხი და დეტალური აღწერა მნიშვნელოვანია ამ მომხმარებლისთვის, ვინაიდან ფიზიკურად ვერ ხედავს პროდუქტს და შესაბამისად სწრაფად ვერ იღებს გადაწყვეტილებას. ასევე, ვინაიდან ხშირად უწევს პროდუქციის შეძენა, ყოველთვის სიახლის ძებნის პროცესშია.*

პერსონის ჩამოყალიბების პროცესში შესაძლოა დამატებით იყოს მითითებული პროფილის სხვა პარამეტრებიც, რომელიც იქნება აქტუალური თვენი კონკრეტული პროექტისათვის. მაგალითად ადგილმდებარეობა, იმ შემთხვევებისთვის როცა მომხმარებელი ბათუმიდან არის, მაღაზიას კი ადგილზე მიტანის მომსახურება მხოლოდ თბილისზე აქვს მორგებული.

მეტი ეფექტურობისთვის, ამობეჭდეთ პერსონების პროფილები და გამოაკარით კედელზე. სამუშაო პროცესში გამოგადგებათ ეს ინფორმაცია და გუნდის წევრებთან ხშირად განიხილავთ პერსონების კონკრეტულ მოთხოვნებს.

**სცენარები**

ჩვენ უკვე გვაქვს ინფორმაცია პოტენციური მომხმარებლის შესახებ და ამ ინფორმაციაზე საფუძველზე, გვაქვს შემუშავებული პერსონები. პერსონების პროფილებიდან მიღებული ინფორმაციის საფუძველზე უნდა შევიმუშაოთ მოკლე სცენარები, რომელიც დაგვეხმარება კონკრეტული მაგალითების ჩამოყალიბებაში, იმაზე თუ როგორ სარგებლობს მომხმარებელი ჩვენი ვებ საიტით.

მაგალითისთვის, განვიხილოთ რამდენიმე სცენარი, რომელიც ელექტრონული კომერციის ტიპის ვებ საიტისთვის იყო შედგენილი.

**სცენარი 1**

ქეთი ალასანიას შვილი დაპატიჟეს მეზობლის დაბადების დღეზე. ზეიმი ხვალ არის და ქეთი ვერ ასწრებს საჩუქრის ყიდვას. მეგობარმა ურჩია სათამაშოების მაღაზიის ვებ საიტიდან საჩუქრის გამოწერა. ქეთი პირველად შედის www.pepela.ge-ზე და 30 ლარის ფარგლებში ეძებს საჩუქარს 7 წლის გოგოსთვის.

*ამ სცენარიდან გავიგეთ, რომ ქეთი ჯერ არ არის ჩვენი ვებ საიტის მომხმარებელი. ის პირველ რიგში ესტუმრება ვებ საიტის მთავარ გვერდს და მისი პირველი მიზანი არის პროდუქციის კატალოგის დათვალიერება იქნება. პროდუქციის კატალოგში ქეთი პირველ რიგში მოძებნის პროდუქციის ფილტრაციის საშუალებას სქესის და ასაკის მიხედვით. ქეთის შეზღუდული ბიუჯეტი აქვს, ამიტომ ფილტრაციის შედეგად აინტერესებს მხოლოდ ის პროდუქცია რომელის ღირებულებაც არ აღემატება 30 ლარს.*

**სცენარი 2**

ნათია ჯალაბაძე “პეპელა”-ს ხშირი მომხმარებელია. მას ელ. ფოსტაზე მოუვიდა შეტყობინება, რომ ელექტრონულ მაღაზიაში ახალი წიგნები გამოქვეყნდა დიდი ფასდაკლებით. ნათია ოჯახთან ერთად ბაკურიანში ისვენებს და ინტერნეტთან წვდომა მხოლოდ მობილურით აქვს.

*ნათიას კონკრეტული მიზანი აქვს, რომელის დაჭერაზე ის პირდაპირ გადავა “პეპელა”-ს ვებ საიტის იმ გვერდზე სადაც ფასდაკლებული წიგნები არის გამოქვეყნებული. ის ამას ტელეფონით აკეთებს, რომელსაც პატარა ეკრანი აქვს და ვინაიდან მობილური ინტერნეტით სარგებლობს, შეერთება არ არის ძალიან სწრაფი. ნათიას არასდროს არ გამოუწერია პროდუქცია თბილისის გარეთ და არც იცის აქვს თუ არა “პეპელა”-ს ბაკურიანში პროდუქციის მიტანის მომსახურება.*

სცენარების ჩამოყალიბების პროცესში, იკვეთება ვებ საიტის გამოყენების სხვადასხვა კონკრეტული დეტალი. დაგეგმარების პროცესში ყველა რეალისტური დეტალის გათვალისწინება იქნება სასარგებლო საბოლოო შედეგის მიღწევაში.

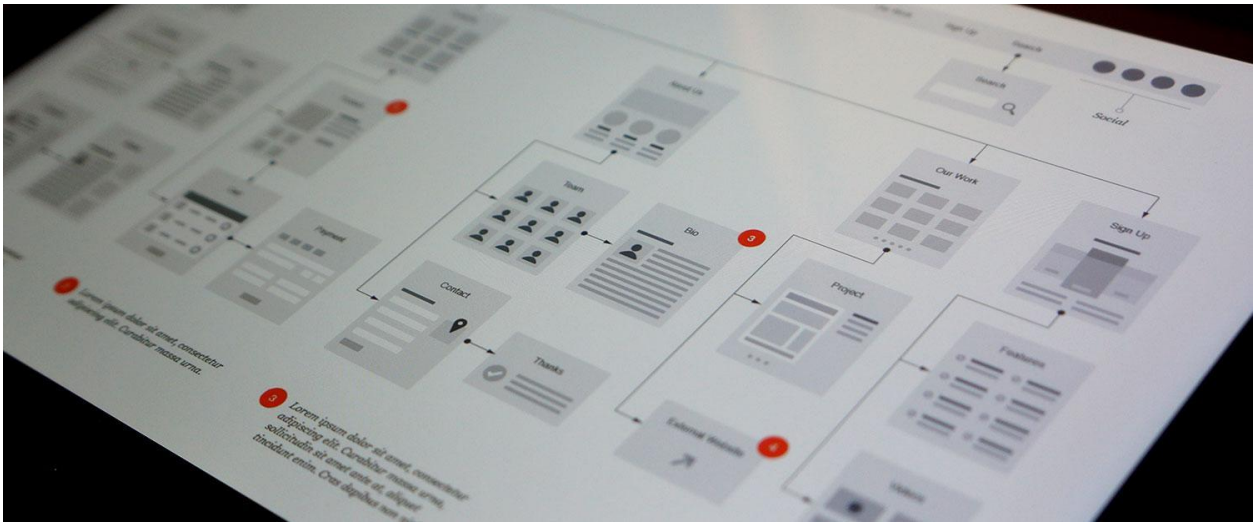
### გვერდები

ვებ საიტი არის ვებ გვერდების ნაკრები. თითოეული ვებ გვერდი თავის, კონკრეტულ ფუნქციას ასრულებს. ვინაიდან გვერდები ერთმანეთში გადაბმულია, ზოგი ფუნქცია ერთი გვერდიდან მეორეზე გადადის და ამით უკვე პროცესად იქცევა.

ინფორმაციის არქიტექტურის შემუშავებისას უნდა ჩამოვაყალიბოთ ვებ საიტის გვერდების რუკა და კონკრეტული გვერდის შესაძლებლობები ავსახოთ. მაგალითისთვის განვიხილოთ სათამაშოების მაღაზიის ვებ საიტის გვერდების რუკა:

ვებ საიტის რუკა არის იერარქიული. პირველ რიგში შევადგინოთ გვერდების იერარქია:

- მთავარი
- ჩვენ შესახებ
- კატალოგი
    - ცხოველები
    - წიგნები
    - თოჯინები
    - მანქანები
    - იარაღები
    - ეზოს სათამაშოები
    - და ა.შ.
- კალათა
- ახალი ამბები
- მომხმარებლის გვერდი
    - რეგისტრაცია
    - ავტორიზაცია
        - ჩემი შეკვეთები
        - ჩემი ფავორიტები
- ხშირად დასმული კითხვები
- საკონტაქტო ინფორმაცია
    - მაღაზიის ფილიალები
    - ცენტრალური ოფისი
- მომსახურების პირობები
    - შეკვეთა
    - ანგარიშსწორება
    - პროდუქციის მიწოდება
    - უსაფრთხოება და კონფიდენციალურობა




შემდეგ, შეგვიძლია გავწეროთ კონკრეტული გვერდის დანიშნულება და ფუნქციონალი. ამ ეტაპზე არ ვუთითებთ ისეთ საერთო ელემენტებს რომლების ყველა გვერდზე ჩანს, მაგალითად ვებ საიტის ლოგო, ძიების ველი და ა.შ.

- მთავარი

  *კატალოგიდან ჩანს ახალი დამატებული და პოპულარული პროდუქცია. ასევე ჩანს სპეციალური შემოთავაზებები.*

- ჩვენ შესახებ

  *ტექსტური ინფორმაცია კომპანიის ისტორიის, საქმიანობის და მომსახურების შესახებ.*

- კატალოგი

  *კატალოგის ყველა კატეგორიის გვერდი იდენტურია.*

  - ცხოველები

    *კატეგორიის გვერდზე ჩანს პროდუქციის ჩამონათვალი. ერთ გვერდზე ჩანს 30 პროდუქტი და არსებობს გვერდების გადაფურცვლის ფუნქცია. თითოეული პროდუქტის ბლოკში ჩანს სურათი, დასახელება, ფასი, კალათაში და ფავორიტებში დამატების საშუალება. შესაძლებელია პროდუქციის დალაგება ანბანის ან ფასის მიხედვით.*

  - წიგნები
    - მოთხრობები 4 წლის ბავშვებისათვის

      *კონკრეტული პროდუქტის გვერდზე ჩანს დიდი ერთი სურათი და სხვა სურათებზე გადართვის შესაძლებლობა. ჩანს პროდუქტის დასახელება, ფასი, პროდუქციის კოდი. არის რაოდენობის მითითების, კალათაში და ფავორიტებში*

*დამატების შესაძლებლობა. არის სოციალურ ქსელებში გაზიარების შესაძლებლობა.*

- თოჯინები
- მანქანები
- იარაღები
- ეზოს სათამაშოები
- და ა.შ.

- კალათა

*მომხმარებლის მიერ კალათაში დამატებული პროდუქციის ჩამონათვალი, სურათით, დასახელებით, ფასით და რაოდენობით. არის კალათიდან პროდუქციის ამოღების, პროდუქციის რაოდენობის რედაქტირების შესაძლებლობა. შესაძლებელია ფასდაკლების კოდის შეყვანა და შეკვეთის გაფორმებაზე გადასვლა.*

- ახალი ამბები

*კომპანიის ახალი ამბების ჩამონათვალი, თარიღით, სურათით, დასახელებით და მოკლე აღწერით. ერთ გვერდზე ჩანს ბოლო 10 სიახლე და გვაქვს გვერდების გადასართველი არქივის სანახავად.*

- მომხმარებლის გვერდი
  - რეგისტრაცია

    *ჩანს მომხმარებლის სარეგისტრაციო ფორმა. ველები: სახელი, გვარი, ელ. ფოსტა, პაროლი. ყველა ველის შევსება აუცილებელია.*

  - ავტორიზაცია

    *ჩანს მომხმარებლის ავტორიზაციის ფორმა. ველები: ელ. ფოსტა, პაროლი. არსებობს პაროლის დამახსოვრების და შეხსენების ფუნქცია. არსებობს ბმული რეგისტრაციის ფორმაზე გადასასვლელი.*

    - ჩემი შეკვეთები

      *რეგისტრირებულ მომხმარებელს უჩანს ყველა წინა გაფორმებული შეკვეთის ჩამონათვალი. ჩანს შეკვეთის ნომერი და ჯამური ღირებულება. ნომერის დაჭერაზე ჩანს შეკვეთის დეტალური პროდუქციის ჩამონათვალი და ადგილზე მიტანის მისამართი.*

    - ჩემი ფავორიტები

*ეს გვერდი არის კალათის გვერდის ანალოგი*

- ხშირად დასმული კითხვები

  *ჩანს ხშირად დასმული კითხვების ჩამონათვალი. კონკრეტულ კითხვაზე დაჭერისას ჩნდება იმ კითხვაზე პასუხი. შესაძლებელია ერთდროულად ორი ან მეტი პასუხის ნახვა.*

- საკონტაქტო ინფორმაცია

  *ჩანს ძირითადი საკონტაქტო ტელეფონის ნომერი, ელ. ფოსტის მისამართი და ცენტრალური ოფისის მისამართი. ოფისის მდებარეობა ასევე ასახულია Google Maps რუკაზე ჭიკარტის სახით.*

  - ფილიალები

    *ჩანს მაღაზიების ჩამონათვალი, სურათით, მისამართით და ტელეფონით. ფილიალები ასევე ასახულია Google Maps რუკაზე ჭიკარტების სახით.*

- მომსახურების პირობები

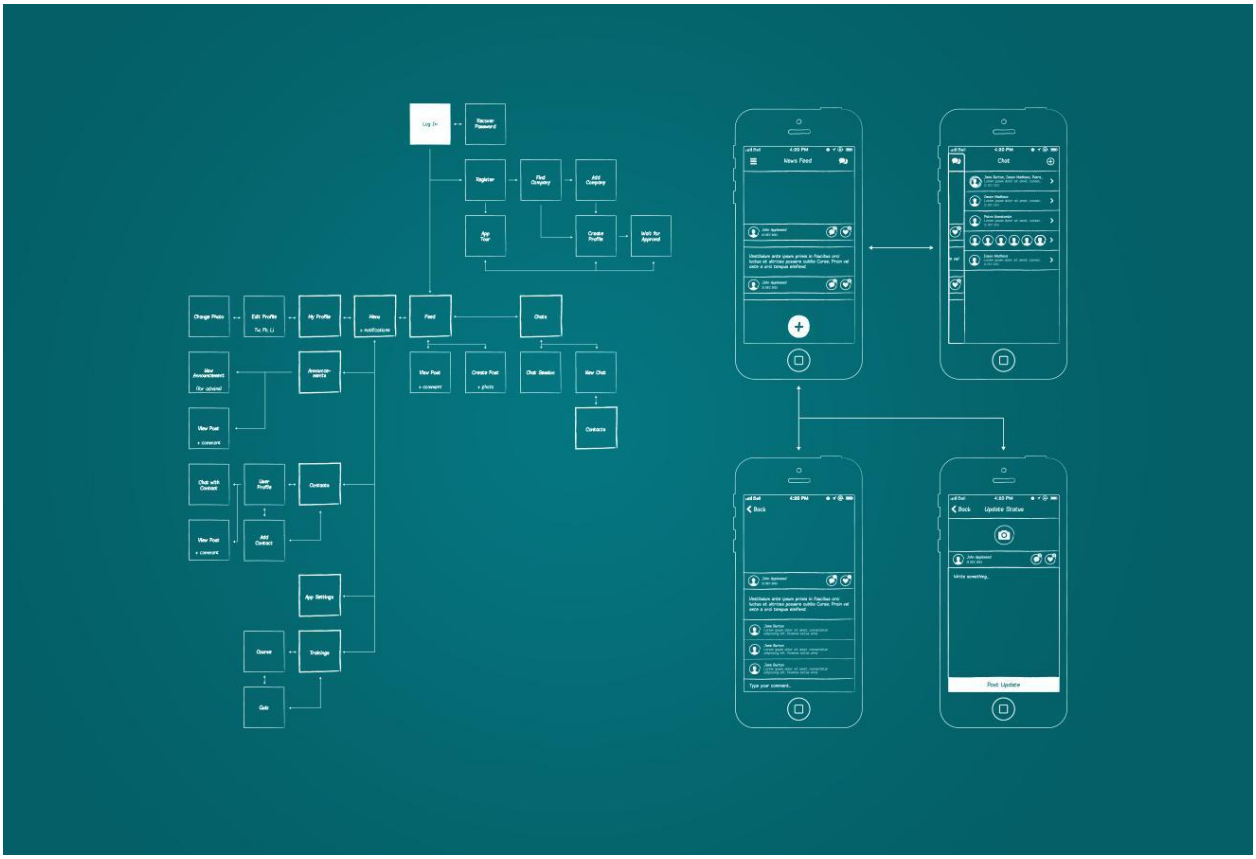
  *პირობების ყველა გვერდის არის ტექსტური. არის ამობეჭდვის შესაძლებლობა და ინფორმაციის გაზიარების შესაძლებლობა სოციალურ ქსელებში.*

  - შეკვეთა
  - ანგარიშსწორება
  - პროდუქციის მიწოდება
  - უსაფრთხოება და კონფიდენციალურობა

ამ ეტაპზე ყველა ძირითადი გვერდი ჩამოყალიბებულია და თითოეული გვერდის შესაძლებლობებიც გვაქვს ასახული. ვებ საიტის რუკის ჩამოყალიბების პროცესში საჭიროა აქტიური კომუნიკაცია პროექტში ჩართულ სამუშაო ჯგუფის წევრებთან, რათა შეათანხმოთ გვერდების შესაძლებლობების დეტალები.

**დიზების პროცესი**

სასურველი შედეგის მისაღწევად, ხშირად მომხმარებლისთვის არ არის საკმარისი ვებ საიტის რომელიმე ერთი გვერდის გახსნა. სათამაშოების მაღაზიის შემთხვევაში, მომხმარებელს მოუწევს რამდენიმე გვერდიდან გვერდზე გადასვლა იმისათვის რომ სასურველი სათამაშო შეიძინოს.



ჩვენ უკვე გვაქვს პერსონა, გვაქვს სცენარი და გვაქვს ვებ საიტის რუკა. ამ ინფორმაციით შეგვიძლია შევადგინოთ კონკრეტული მომხმარებლის დინების პროცესი და დავაკვირდეთ თუ პროცესში ყველა საჭირო ელემენტი გვაქვს გათვალისწინებული.

ქეთი ალასანიას სცენარის მიხედვით, მისი შესაძლო დინების პროცესი იქნება შემდეგი:

1. ქეთი, სახლის კომპიუტერიდან შედის www.pepela.ge-ზე
2. ათვალიერებს ვებ საიტის მთავარ გვერდზე განთავსებულ პროდუქციას და ვერ ხედავს მისთვის სასურველ პროდუქციას
3. ვინაიდან ქეთი არ იცის კონკრეტულად რას არჩევს საჩუქრად, ის ვერ გამოიყენებს ძიების ფუნქციას და შესაბამისად მისი ერთადერთი ალტერნატივა არის კატალოგის გვერდზე გადასვლა
4. კატალოგის გვერდზე ის ხედავს პროდუქციის ფილტრაციის შესაძლებლობას კატეგორიის მიხედვით და ჩამონათვალში პოულობს თოჯინების კატეგორიას
5. თოჯინების გვერდზე ხედავს სხვადასხვა პროდუქციას სხვადასხვა ფასებით და ალაგებს პროდუქციას ფასების მიხედვით
6. მიღებული შედეგიდან არჩევს თოჯინას 30 ლარის ფარგლებში და აჭერს სურათს
7. კითხულობს პროდუქციის დეტალურ ინფორმაციას და ამატებს პროდუქციას კალათაში
8. გადადის კალათის გვერდზე და აჭერს “შეკვეთის გაფორმებას”
9. ვინაიდან ეს ქეთის პირველი შეკვეთა არის, ის ირჩევს ახალი ანგარიშის რეგისტრაციას და ავსებს სარეგისტრაციო ფორმას
10. რეგისტრაციის შედეგად მას უწევს ანგარიშის დადასტურება ელ. ფოსტაზე მიღებული ბმულით.
    a. ქეთი ამოწმებს ფოსტას
    b. ადასტურებს ანგარიშის რეგისტრაციას და ბრუნდება “პეპელა”-ს ვებ საიტზე
11. ახალი რეგისტრირებული ანგარიშით გადის ავტორიზაციას

12. ადასტურებს შეკვეთის გაფორმებას და ავსებს ადგილზე მიტანის მისამართს
13. გადადის გადახდის გვერდზე და ავსებს პლასტიკური ბარათის მონაცემებს
14. ბრუნდება ვებ საიტის გვერდზე და ხედავს შეკვეთის წარმატებით გაფორმების შეტყობინებას. ვებ საიტის ატყობინებს რომ შეკვეთა იქნება მიტანილი მითითებულ მისამართზე 24 საათში.

დეტალური პროცესის ჩამოყალიბება გვეხმარება, ქეთის მიერ გასავლელი დინების წარმოდგენაში და ყველა საფეხურზე პროცესის გამარტივების შესაძლებლობების მოფიქრებაში.

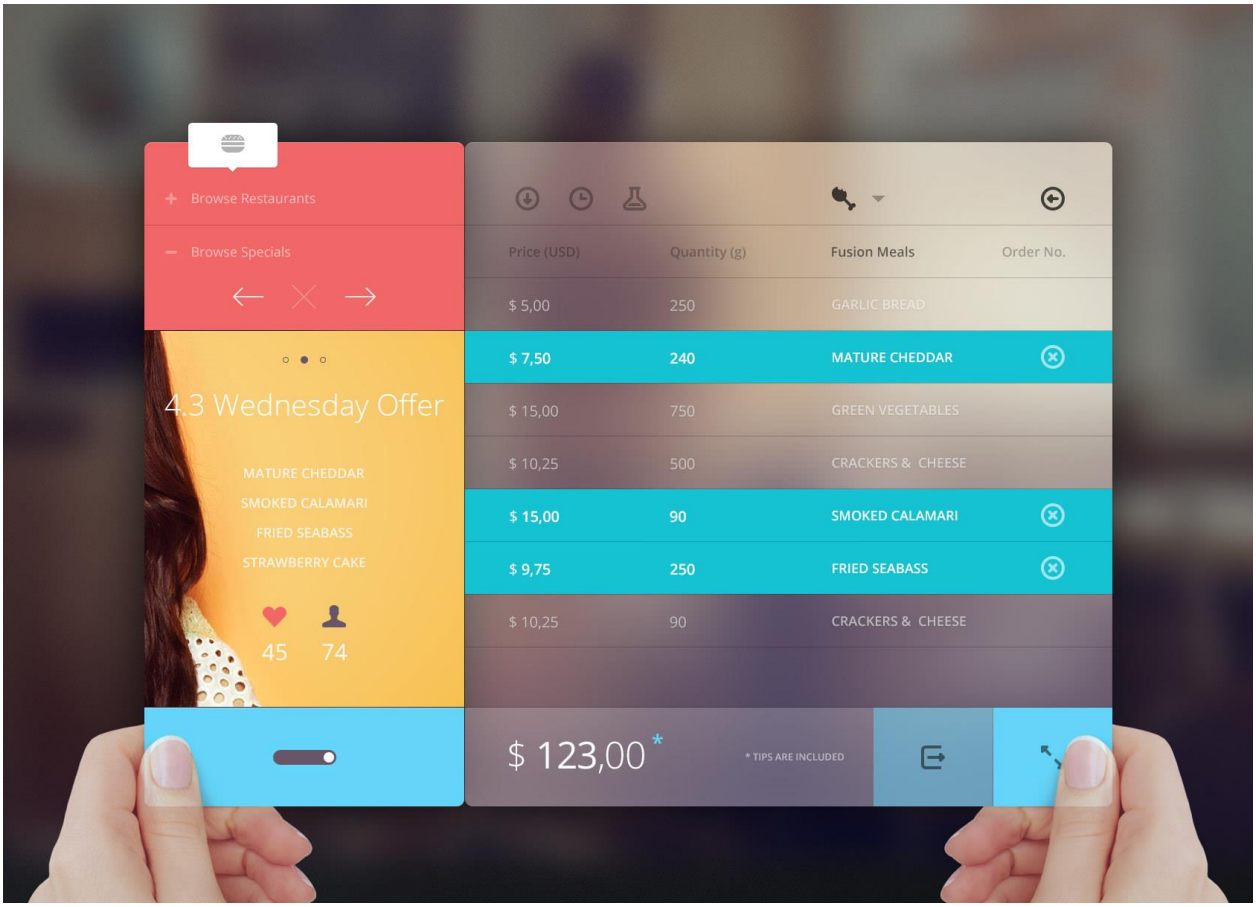
სამაგალითო დინების პროცესი არის მოცემული მხოლოდ ქეთის სცენარის შემთხვევაში, წარმოიდგინეთ სხვა შემთხვევები. რა მოხდება იმ შემთხვევაში, როდესაც მომხმარებელს ორი სხვადასხვა შეკვეთა აქვს რომელიც ორ სხვადასხვა მისამართზე არის მისატანი. როგორც იცვლება დინება ამ შემთხვევაში და რა როგორ შევძლებთ მომხმარებლის სურვილების დაკმაყოფილებას.

### ნავიგაცია

უფრო ხშირია შემთხვევები, როდესაც მომხმარებელი საძიებო სისტემიდან ან კონკრეტული ბმულიდან პირდაპირ ვებ საიტის რომელიმე შიდა გვერდზე ხვდება და არა ვებ საიტის მთავარი გვერდიდან. ამის გამო აუცილებელია ვებ საიტის ნავიგაციაში გაითვალისწინოთ

ისეთი სტრუქტურა, რომლის გამოყენებით მომხმარებელი შეძლებს ნებისმიერი გვერდიდან ნებისმიერ გვერდზე გადასვლას.

ვებ საიტის ნავიგაცია არის ერთ-ერთი ყველაზე მნიშვნელოვანი ნაწილი ვებ საიტის დიზაინში, ვინაიდან თუ ნავიგაციის გამო მომხმარებელი დაიკარგა საიტის გვერდებში და ვერ ახერხებს თავის მიზნების მიღწევას, მომხმარებელი დატოვებს ვებ საიტს და დიდი ალბათობით არასდროს დაბრუნდება უარყოფითი გამოცდილების გამო.




არსებობს ნავიგაციის რამდენიმე ძირითადი ელემენტი, რომელიც შეგიძლიათ გაითვალისწინოთ ვებ საიტის ინფორმაციის არქიტექტურის ჩამოყალიბებაში:

- **ტერმინოლოგია**
  სფეროს ტერმინოლოგია მხოლოდ იმ სფეროს წარმომადგენლებისთვის არის გასაგები. რიგითი მომხმარებელი უნდა მიხვდეს მენიუს პუნქტის დასახელებიდან, თუ რა ტიპის ინფორმაცია ელის იმ გვერდზე, სადაც ის მენიუს პუნქტი გადაიყვანს.

  მაგალითად: სათამაშოების მაღაზიის კატალოგში, კატეგორიის დასახელება “თოჯინები” ან “სუპერ გმირები” უფრო გასაგებია, ვიდრე “პლასტმასის სათამაშოები”

- **სტატუსი**
  ვებ საიტის შიდა გვერდებზე, მომხმარებელი უნდა ხედავდეს თუ სად იმყოფება ის ნებისმიერ დროს. განვიხილოთ რამდენიმე მნიშვნელოვანი მაგალითი:

  - თუ მომხმარებელმა კატალოგიდან რომელიმე კატეგორია აირჩია, ის კატეგორია უნდა გამოირჩეოდეს დანარჩენებისგან.
  - თუ მომხმარებელი მენიუს პუნქტიდან ირჩევს პუნქტს “ჩვენ შესახებ”, მაშინ ის პუნქტი უნდა იყოს გამოკვეთილი სხვა მენიუს პუნქტებიდან.
  - იმ შემთხვევაში თუ მომხმარებელი დაიკარგა, მას უნდა შეეძლოს მარტივად ვებ საიტის მთავარ გვერდზე გადასვლა ლოგოს დაჭერით მეშვეობით.
  - როდესაც ვებ საიტის რუკის იერარქია რამდენიმე საფეხურიდან შედგება, ყოველთვის სასურველია მეორე საფეხურიდან დაწყებული მომხმარებელს ვაჩვენოთ კონკრეტული

გვერდის ადგილმდებარეობა. მაგალითად: მთავარი > კატალოგი > ახალი პროდუქცია > თოჯინები > მოლაპარაკე თოჯინები. ამას ინდუსტრიაში პურის ნამცეცებს ეძახიან და ეს მომხმარებელს დამატებით სანავიგაციო საშუალებას აძლევს.

- **ძიება**
  Google-ის პოპულარიზაციის შედეგად, რიგითი მომხმარებელი ელის ყველა ვებ საიტის ძიების შესაძლებლობა რომ მუშაობდეს იგივე ნაირად როგორც Google-ის საძიებო სისტემა მუშაობს. ძიების შედეგის გვერდი არის მომხმარებლისთვის ალტერნატიული ნავიგაციის შესაძლებლობა თუ მანდ ვერ იპოვა სასურველი ინფორმაცია სტანდარტული მენიუს ნავიგაციის საშუალებით.

  ძიების შედეგის გვერდზე დაალაგეთ ინფორმაცია მიახლოებით ისე, როგორც Google-ის შედეგის გვერდი არის განლაგებული. აჩვენეთ კონკრეტული შედეგის "პურის ნამცეცები", ეს ეხმარება მომხმარებელს მიღებული შედეგის აღქმაში.

**შეჯამება**

გეგმის გარეშე პროექტი არ იქნება წარმატებული. წარმოიდგინეთ შენობის მშენებლობის პროცესი, არქიტექტურული გეგმის გარეშე. ინფორმაციის არქიტექტურის ელემენტები, თქვენი პროექტის დაგეგმარების პროცესის ნაწილებია. დიდ პროექტებში ინფორმაციის არქიტექტურის გარეშე წარმოუდგენელია პროექტის დასრულება.

დაგეგმეთ თქვენი პროექტი განხილული ინფორმაციის არქიტექტურის ელემენტების საშუალებით:

- აქციეთ მომხმარებლები პერსონებად
- შეადგინეთ სცენარები პერსონებისთვის
- გაატარეთ პერსონები დინების პროცესში, სცენარის მიხედვით
- შეადგინეთ საჭირო გვერდების დეტალური აღწერა გავლილი დინების საფუძველზე
- შეადგინეთ ნავიგაციის მოხერხებული საშუალებები

კლასიკური ინფორმაციის არქიტექტურა ბევრად მეტია, ვიდრე ის ელემენტები, რაც ამ თავში განვიხილეთ. განხილული ელემენტები გაძლევთ ამ დისციპლინის საბაზისო შესავალს და გასწავლით მათ გამოყენებას პროექტის შესრულების სასარგებლოდ. ინფორმაციის არქიტექტურაზე ბევრი კარგი წიგნი არსებობს, რომელიც გაგაცნობთ სხვა ელემენტებს და მათი გამოყენების სპეციფიკას.

**სავარჯიშო**

მოახდინეთ თქვენს მიერ დაგეგმილი პროექტისთვის პერსონების და მათი სცენარების ჩამოყალიბება.

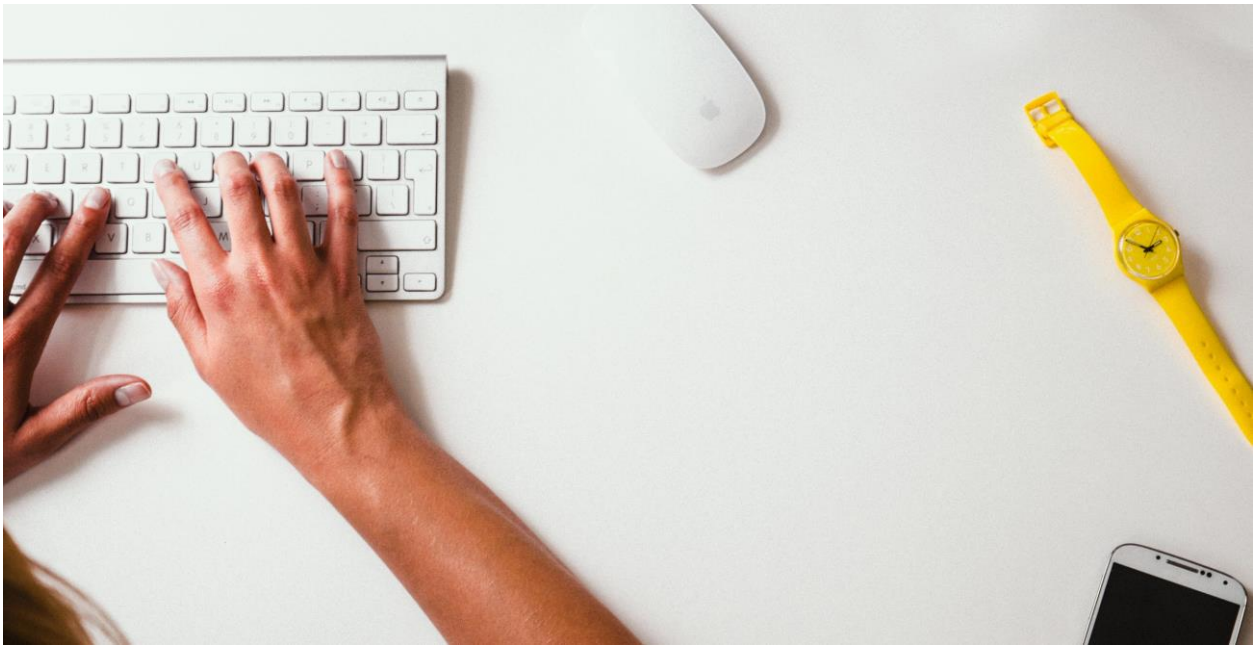
### 3.4.2. კონკურენტების ანალიზი

სწავლის შედეგის შესაბამისი თემატიკა

- კონკურენტების იდენტიფიცირება
- კონკურენტების შედარების კრიტერიუმების შედგენა
- სასარგებლო ხელსაწყოების განხილვა

ყოველდღე ათასობით ახალი ვებ საიტი ეშვება. დიდი ალბათობით, უკვე არსებობს ვებ საიტები იმ თემატიკის, რა თემატიკაზეც თქვენ იმუშავებთ, თქვენი შემდეგი პროექტის ფარგლებში. ამ თავში ჩვენ განვიხილავთ, არსებული ვებ საიტების ანალიზის პროცესს, რომელიც დაგეხმარებათ თქვენი კონცეფციის შემუშავებაში.

ანალიზის პროცესში, არ არის აუცილებელი "კონკურენტი" იყოს კონკრეტული ბიზნესის პირდაპირი კონკურენტი, რომელიც იგივე ბაზარზე მოღვაწეობს. ამ კონტექსტში, "კონკურენტი" არის ნებისმიერი ვებ საიტი, მსოფლიოს ნებისმიერ ქვეყანაში, რომელიც იგივე ან მონათესავე ინდუსტრიას ეხება, რომელიც თქვენ შეგიძლიათ გამოიყენოთ ანალიზის პროცესში.



კონკურენტების შესწავლის პროცესი, შესაძლოა ზედმეტად მოგეჩვენოთ, ვინაიდან შესწავლილმა დეტალებმა შესაძლოა მოახდინოს ზეგავლენა თქვენ შემოქმედებაზე და ვებ საიტის აწყობის პროცესში გექნებათ გამოყენებული დაკოპირებული გადაწყვეტილებები და დაკოპირებული ვიზუალური ელემენტები.

გამოიყენეთ ეს პროცესი, როგორც ინსტრუმენტი, რომელიც გაძლევთ საშუალებას შეისწავლოთ კონკრეტული ინდუსტრია, გაეცნოთ საჯაროდ გამოყენებულ ტერმინოლოგიას და გაითვალისწინოთ ის შეცდომები, რაც კონკურენტებს აქვთ დაშვებული. კონკურენციის ანალიზი არის უფრო ტექნიკური და შინაარსობრივი ანალიზის პროცესი, ვიდრე ვიზუალური შთაგონების მოძიების.

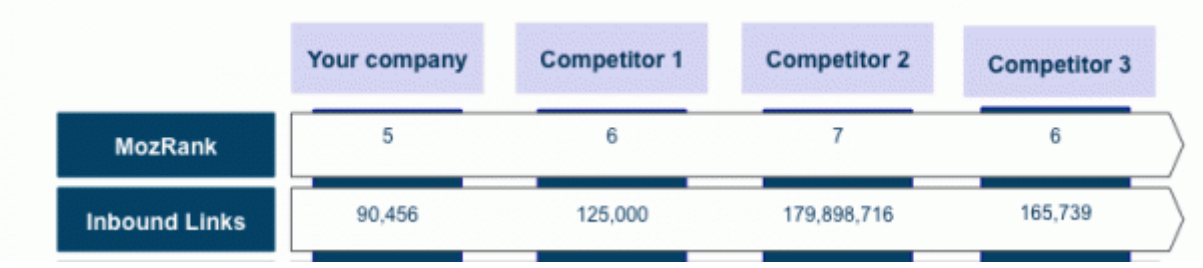
**იდენტიფიცირება**

პირველ რიგში, თქვენ დაგჭირდებათ რამდენიმე ვებ საიტის მისამართი, რომელსაც გამოიყენეთ ანალიზის პროცესში. ეფექტური შედეგისთვის, მოიძიეთ 10 ან მეტი, განსხვავებული ვებ საიტი. რამდენიმე ვებ საიტის მისამართი მოცემული იქნება პროექტის ბრიფში. კითხეთ დაკვეთს, მოძებნეთ Google-ში, კითხეთ პოტენციურ მომხმარებლებს რომელ სხვა რესურსებთან ჰქონდათ შეხება. მოძებნეთ ორგანიზაციის ზუსტი ანალოგი სხვა ქვეყნებში, მოძებნეთ მონათესავე ინდუსტრიის წარმომადგენლები.

გადაახარისხეთ ყველა მიღებული მისამართი. განსაკუთრებული ყურადღება მიაქციეთ იმ ვებ საიტებს, რომელის მისამართი გამეორდა ძიების პროცესში. ანალიზისთვის აირჩიეთ პოპულარულო ორგანიზაციები, ორგანიზაციები რომლებიც დიდი ხანია მოღვაწეობენ ბაზარზე. ნუ შეარჩევთ მხოლოდ ვიზუალური მხარის მიხედვით. თუ თქვენ არ მოგწონთ როგორ გამოიყურება კონკრეტული ვებ საიტი, ეს იმას არ ნიშნავს რომ მანდ არ იქნება თქვენთვის სასარგებლო ინფორმაცია.

გამოიყენეთ Google-ის აპლიკაცია Chrome ბრაუზერისთვის, სახელით Google Similar Pages. ამ აპლიკაციის დახმარებით აღმოაჩენთ სხვა ვებ საიტებს, რომელიც Chrome-ში გახსნილი კონკრეტული ვებ საიტის მსგავსი იქნება.


შეადგინეთ კონკურენტების ცხრილი. ცხრილის სვეტებში ჩამოწერეთ კონკურენტების სახელები და მწკრივებში ჩამოწერეთ შედარების კრიტერიუმები. მაგალითისთვის, შეადარეთ 4 განსხვავებული ვებ საიტის სტანდარტული ტექსტისთვის შერჩეული ფონტის სახელი, ფონტის ფერი და ფონტის ზომა.



| | Your company | Competitor 1 | Competitor 2 | Competitor 3 |
|---|---|---|---|---|
| MozRank | 5 | 6 | 7 | 6 |
| Inbound Links | 90,456 | 125,000 | 179,898,716 | 165,739 |

**შედარების კრიტერიუმები**

კონკურენტების ანალიზის ცხრილში ჩამოწერილი შედარების კრიტერიუმები დიდი ალბათობით

განსხვავებული იქნება, გამომდინარე იქიდან თუ რა ტიპის პროექტზე მუშაობთ და რის აღმოჩენას ცდილობთ ანალიზის შედეგად.

შემდეგი რამდენიმე შედარების კრიტერიუმები, გამოგადგებათ ყველა ტიპის ვებ საიტის ანალიზის პროცესში.

- **ძიების შედეგი**
  რა პოზიცია უკავია კონკურენტს Google-ის საძიებო სისტემის შედეგის გვერდზე, კონკრეტული საკვანძო სიტყვის მიხედვით?

- **მობილური ვერსია**
  ოპტიმიზირებულია თუ არა კონკურენტის ვებ საიტი მობილური მოწყობილობებისათვის?

- **ჩატვირთვის სისწრაფე**
  რამდენ წამში იხსნება კონკურენტის მთავარი გვერდი და თუ ის შედარებით ნელია, რა არის ამის მიზეზი?

- **ფერთა პალიტრა**
  ვებ საიტის ღია თუ მუქ ფერებში არის გადაწყვეტილი. ინტერნეტ კაზინოების შემთხვევაში, დიდი ალბათობით უმრავლესობა იქნება მუქ ფერებში და ეს არ იქნება შემთხვევითი.

ანალიზის პროცესში, თქვენ შეამჩნევთ სხვადასხვა ვებ საიტებს აქვთ ბევრი საერთო ფუნქცია. მაგალითად, თუ იკვლევთ სადაზღვეო მომსახურების კომპანიების ვებ საიტებს, დიდი ალბათობით უმრავლესობას ექნება დაზღვევის კალკულატორის ფუნქცია, რომელიც მომხმარებელს აძლევს “თვითმომსახურების” საშუალებას.

ჩაინიშნეთ ცალკე აღმოჩენილი მომხმარებლისთვის სასარგებლო ფუნქციები, რომელიც შედარების ცხრილში არ გაქვთ შეყვანილი და გაითვალისწინეთ ისინი თქვენი კონცეფციის შემუშავების პროცესში.

გაითვალისწინეთ, რომ თქვენი პოტენციური მომხმარებელი შესაძლოა იყოს თვენი კონკურენტის არსებული მომხმარებელი და უკვე მიჩვეულია კონკურენტის ვებ საიტის გამოყენებას. აქედან გამომდინარე, თქვენი გადაწყვეტილება არ უნდა იყოს რადიკალურად განსხვავებული ანალიზის შედეგად მიღებული საერთო ნორმებიდან.

### დამატებითი სერვისები

კონკურენტების შესწავლის პროცესში, შესაძლოა თქვენთვის სასარგებლო იყოს კონკრეტული ვებ საიტის მომხმარებელთა სტატისტიკური მაჩვენებლები, რომელიც არ არის პირდაპირ კონკურენტის ვებ საიტიდან ხელმისაწვდომი. ამისთვის არსებობს რამდენიმე სერვისი, რომელიც დაგეხმარებათ ამ ინფორმაციის აღმოჩენაში:

- Wayback Machine

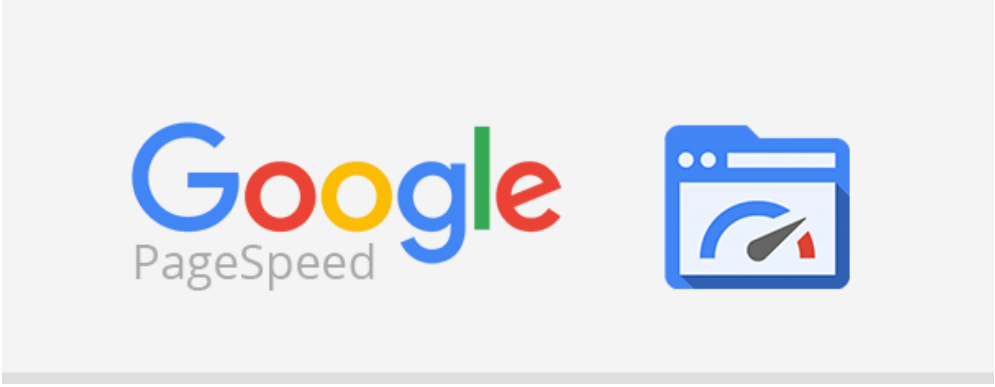

ეს არის ინტერნეტის არქივის სერვისი, რომელი ინახავს ვებ საიტის გამოსახულებას. თქვენ შეგიძლიათ ნახოთ როგორ გამოიყურებოდა კონკურენტის საიტი წლების წინ და როგორ ვითარდებოდა ის დროთა განმავლობაში.

- SimilarWeb


ამ სერვისის საშუალებით თქვენ შეგიძლიათ შეისწავლოთ კონკურენტი ვებ საიტის მომხმარებელს სტატისტიკა და გაიგოთ:

  - რამდენი მომხმარებელი ესტუმრა ვებ საიტს წინა თვეში
  - რამდენი ხანი ჩერდება მომხმარებელი მოცემულ ვებ საიტზე
  - რომელი ქვეყნებიდან შემოდის მომხმარებლების უმრავლესობა
  - რომელი ვებ საიტებიდან გადმოდიან მომხმარებლები მოცემულ ვებ საიტზე
  - და უამრავი სხვა სასარგებლო ინფორმაცია

- PageSpeed Insights



ეს არის Google-ის უფასო სერვისი, რომელიც დაგეხმარებათ კონკრეტული ვებ საიტის ჩატვირთვის სიჩქარის ანალიზში.

- SEMrush


საძიებო სისტემების ოპტიმიზაციის საკმაოდ დეტალური ანალიზის სერვისი.

**შეჯამება**

კონკურენტების ანალიზის პროცესში, თქვენ შეისწავლით კონკრეტული ინდუსტრიის შესახებ სასარგებლო ინფორმაციას. შეხედავთ სხვადასხვა ორგანიზაციების ვებ საიტების მომხმარებლის თვალით. ნახავთ როგორ ვითარდებოდა ინდუსტრიის ლიდერების ვებ საიტები დროთა განმავლობაში და რას ითვალისწინებდნენ ისინი ვებ საიტის დახვეწის პროცესში.

ანალიზის პროცესში, ყველა მიღებული ინფორმაცია სასარგებლოა. შეადგინეთ ანალიზის საერთო ანგარიში, ცხრილით და თქვენი პირადი აზრებით. განიხილეთ ანგარიში გუნდის წევრებთან ან/და დამკვეთთან. საჭიროების შემთხვევაში შეიტანეთ ცვლილებები პროექტის ბრიფში.

**სავარჯიშო**

მოახდინეთ კონკურენტების ანალიზი, წინა დავალებებში განხილული პროექტის საფუძველზე.

### 3.4.3. სტრუქტურის შემუშავება (Wireframing)

სწავლის შედეგის შესაბამისი თემატიკა

- სტრუქტურის დანიშნულების განხილვა
- ქაღალდის სტრუქტურის შემუშავება
- სასარგებლო ხელსაწყოების განხილვა

*ინსტრუმენტს, რომელსაც ამ თავში განვიხილავთ ჰქვია wireframe. ქართულად ამას ზოგი ჩონჩხს ეძახის, ზოგი სტრუქტურას და ზოგი ნახაზს. სიმარტივის გამო ჩვენ ავირჩიეთ ტერმინი "სტრუქტურა", თუმცა ინდუსტრიაში უფრო ხშირად wireframe-ს იყენებენ.*

სტრუქტურის შემუშავება არის ძალიან მნიშვნელოვანი ეტაპი ყველა ტიპის ინტერფეისის შემუშავების პროცესში. ის გაძლევთ საშუალებას, ვიზუალურად განალაგოთ ინფორმაცია და წარმოიდგინოთ როგორ მიიღებს ამ ინფორმაციას მომხმარებელი.

სტრუქტურის შემუშავების პროცესში, ფერი ხელს შეგიშლით კონცენტრირებაში. თქვენი ძირითადი ამოცანა უნდა იყოს ინფორმაციის სწორი განლაგება და არა სტრუქტურის სილამაზეზე. ამიტომ, სასურველია რომ სტრუქტურა იყოს შავ-თეთრი.

**დანიშნულება**

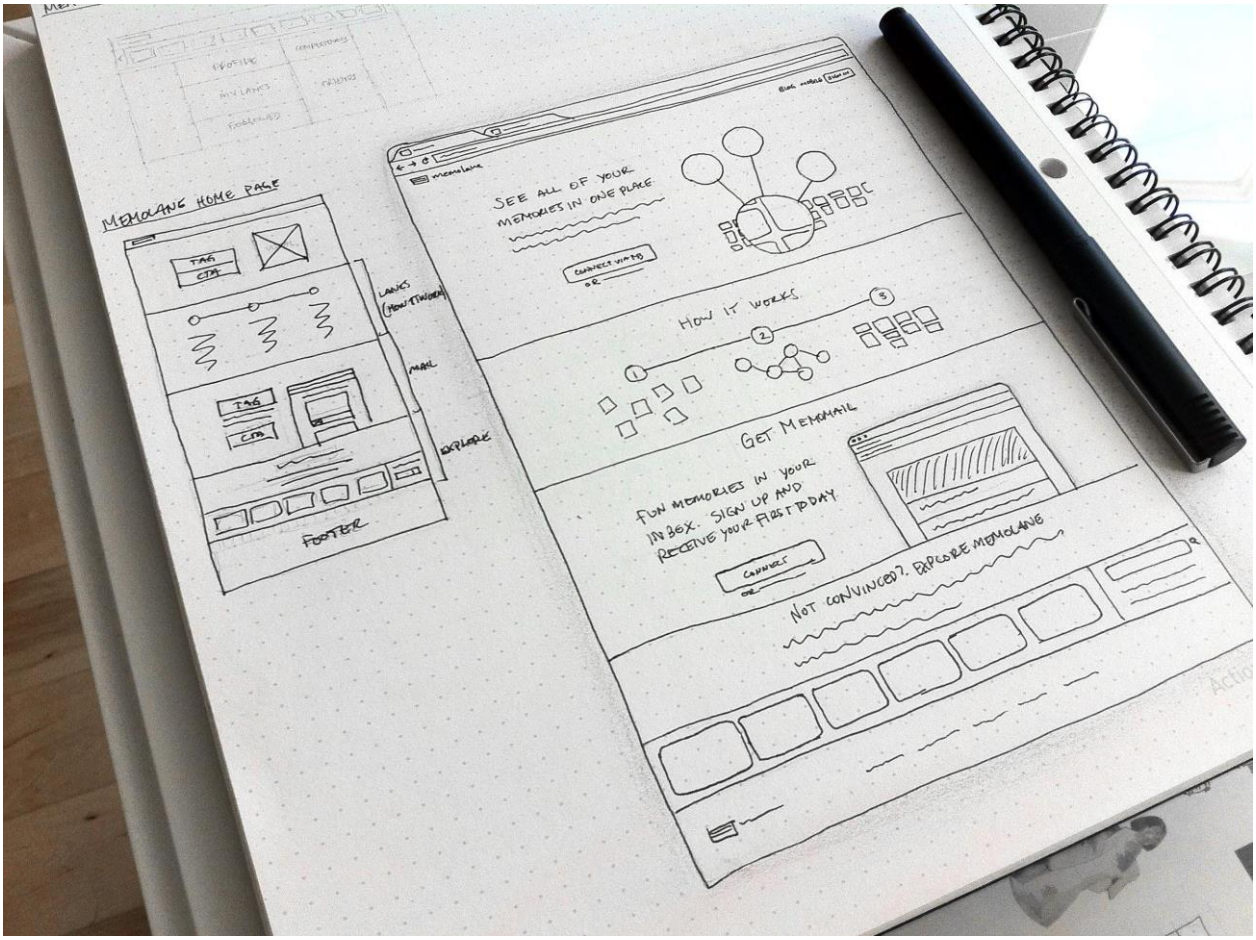
თქვენს მიერ შემუშავებული სტრუქტურა უნდა ასრულებდეს თავის ძირითად დანიშნულებას, რომელიც არის ინფორმაციის ეფექტური განლაგება და თქვენი ხედვის გაზიარება გუნდის წევრებთან ან/და დამკვეთთან.

ინფორმაციის არქიტექტურის დაგეგმარების პროცესების შედეგად, თქვენ გექნებათ ზოგადი წარმოდგენა, თუ რა ელემენტებისგან უნდა შედგებოდეს ვებ საიტის კონკრეტული გვერდი. სტრუქტურა მოგცემთ საშუალებას დაალაგოთ ეს ელემენტები და გვერდს შეუდგინოთ იერარქიულად სტრუქტურირებული კომპოზიცია.

სტრუქტურის შემუშავება დაგიზოგავთ დროს, ვინაიდან ელემენტების განლაგების პროცესში უფრო ნათელი გახდება რომელი ელემენტი, სად უნდა იყოს განლაგებული და ეს პროცესი ხშირ შემთხვევაში მოითხოვს ბევრ ცვლილებას. სტრუქტურის გარეშე ცვლილებები შეტანა მოგიწევთ პირდაპირ ვებ საიტის დიზაინში, რაც საკმაოდ შრომატევადია.

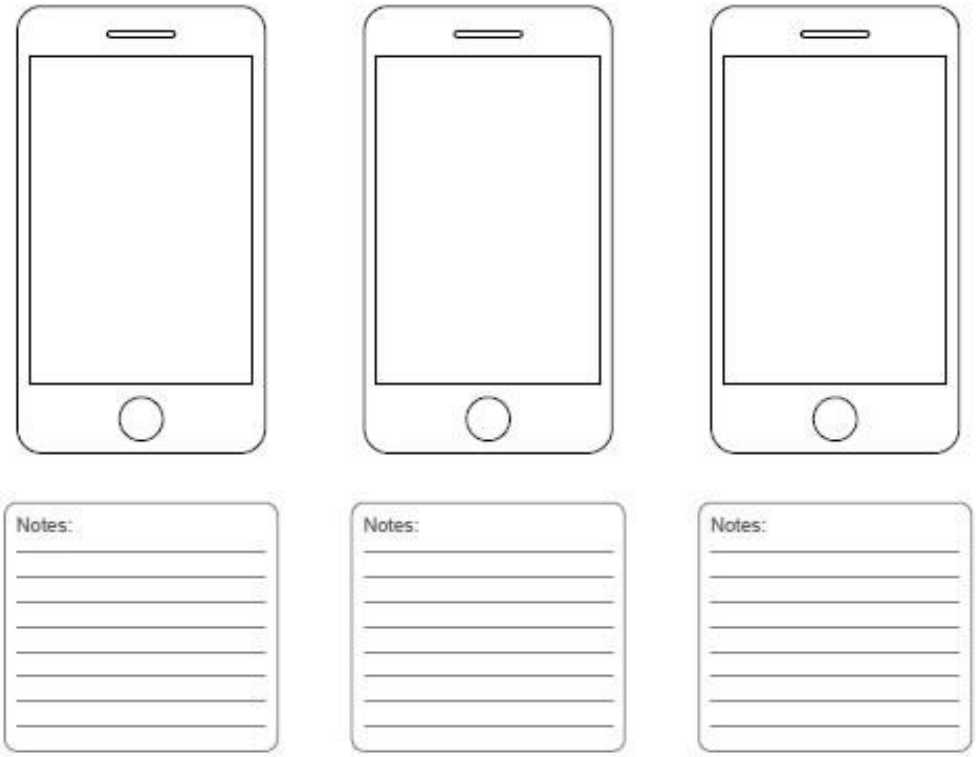
### ქაღალდის სტრუქტურა

სტრუქტურების შემუშავების სხვადასხვა ხერხი არსებობს. ყველაზე მარტივი და სწრაფი არის სტრუქტურის აგება ქაღალდზე. ამისთვის თქვენ არ გჭირდებათ არც კომპიუტერი და არც სპეციალური პროგრამები. მარტივად ჩამოაყალიბეთ სტრუქტურა ქაღალდზე ან დაფაზე და მომენტალურად მიიღეთ თქვენი ხედვის ვიზუალური ინტერპრეტაცია. ვინაიდან სტრუქტურაში მოგიწევთ ცვლილებები შეტანა, მარტივი იქნება კონკრეტული ელემენტების წაშლა ან სრულიად ახალი სტრუქტურის აგებაც.




ქაღალდის მეთოდის არჩევის შემთხვევაში, ისარგებლეთ შემდეგი რჩევებით:

- ცვლილებების გამარტივების მიზნად, გამოიყენეთ ფანქარი და საშლელი
- პროცესის აჩქარების მიზნად, ნუ გამოიყენებთ სახაზავს ან სხვა ხაზვის დამხმარე საშუალებას
- არ არის აუცილებელი ხატვა იცოდეთ რომ მოახერხოთ სტრუქტურის აგება
- შეიძინეთ და თან იქონიეთ ბლოკნოტი, რომელშიც ნებისმიერ დროს მოახერხებთ სტრუქტურის აგებას. ბლოკნოტი, გარკვეულ წილად, შეასრულებს ძველი სტრუქტურების არქივს.
- არსებობს უამრავი უფასო ქაღალდის შაბლონი, რომელიც კომპიუტერის ეკრანის ან მობილური მოწყობილობის ჩარჩოს გაძლევთ ქაღალდზე. შაბლონის შესარჩევად მოძებნეთ "wireframing PDF templates" Google-ში.



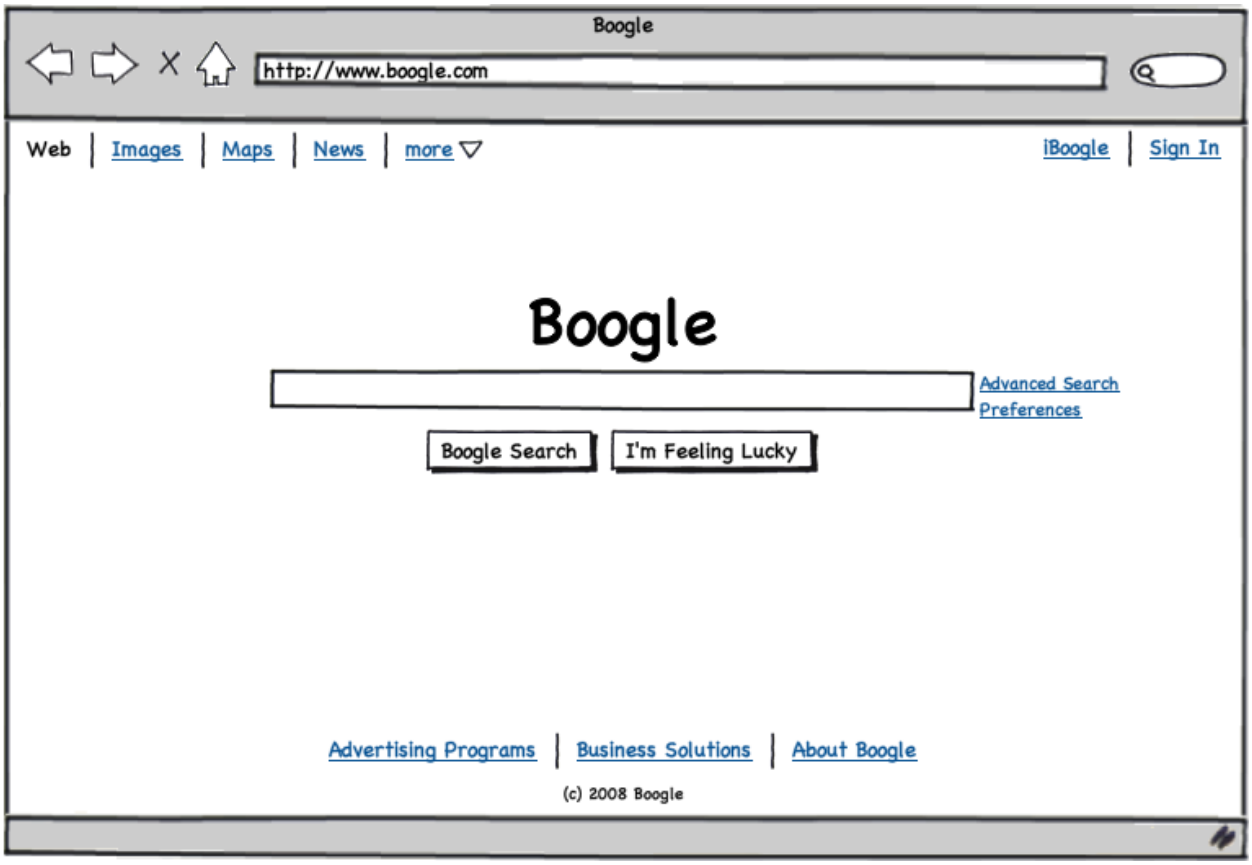

**პროგრამული უზრუნველყოფა**

არსებობს არაერთი პროგრამა, რომელიც მოგეხმარებათ სტრუქტურების აგებაში. ზოგ შემთხვევაში, განსაკუთრებით თუ დიდ და გრძელვადიან პროექტზე მუშაობთ, უმჯობესია რომ რომელიმე პროგრამა გამოიყენოთ. პროგრამა ზოგი პროცესის ავტომატიზაციას მოახდენს და ამით გაგიმარტივებთ საქმეს.

სტრუქტურების აგების პროგრამებში ასევე არსებობს წინასწარ შედგენილი სხვადასხვა ხშირად გამოყენებული სტრუქტურის ელემენტები, როგორიც არის ღილაკები, ტექსტი, ფორმის ელემენტები და სხვა. თქვენ აღარ მოგიწევთ ყველა ელემენტის თავიდან მომზადება და მარტივად შეგიძლიათ დააკოპიროთ და მართოთ ისინი.

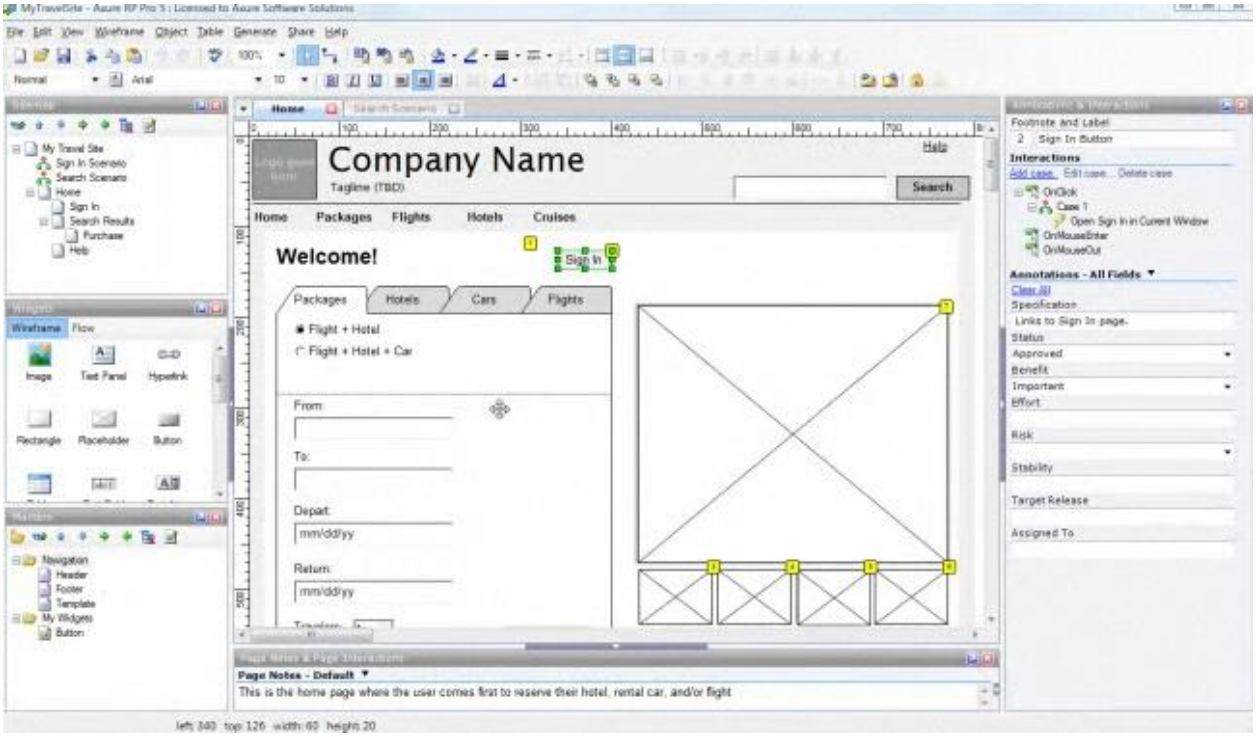
**Balsamiq - www.balsamiq.com**

სურათზე არის ნაჩვენები სტრუქტურა, რომელიც აგებულია ამ პროგრამაში. როგორც ხედავთ ის სპეციალურად გამოიყურება ისე თითქოს ქაღალდზე იყო შემუშავებული. ეს არის Balsamiq-ის ყველაზე დიდი უპირატესობა. მას ასევე აქვს წინასწარ შექმნილი თითქმის ყველა სტრუქტური ელემენტი, რომელიც შეიძლება დაგჭირდეთ სამუშაო პროცესში. სტრუქტური აგება ხდება drag-and-drop საშუალებით. პროგრამა ხელმისაწვდომია Windows და Mac პლატფორმებზე და ასევე ვებ აპლიკაციის სახით.




**Axure - www.axure.com**

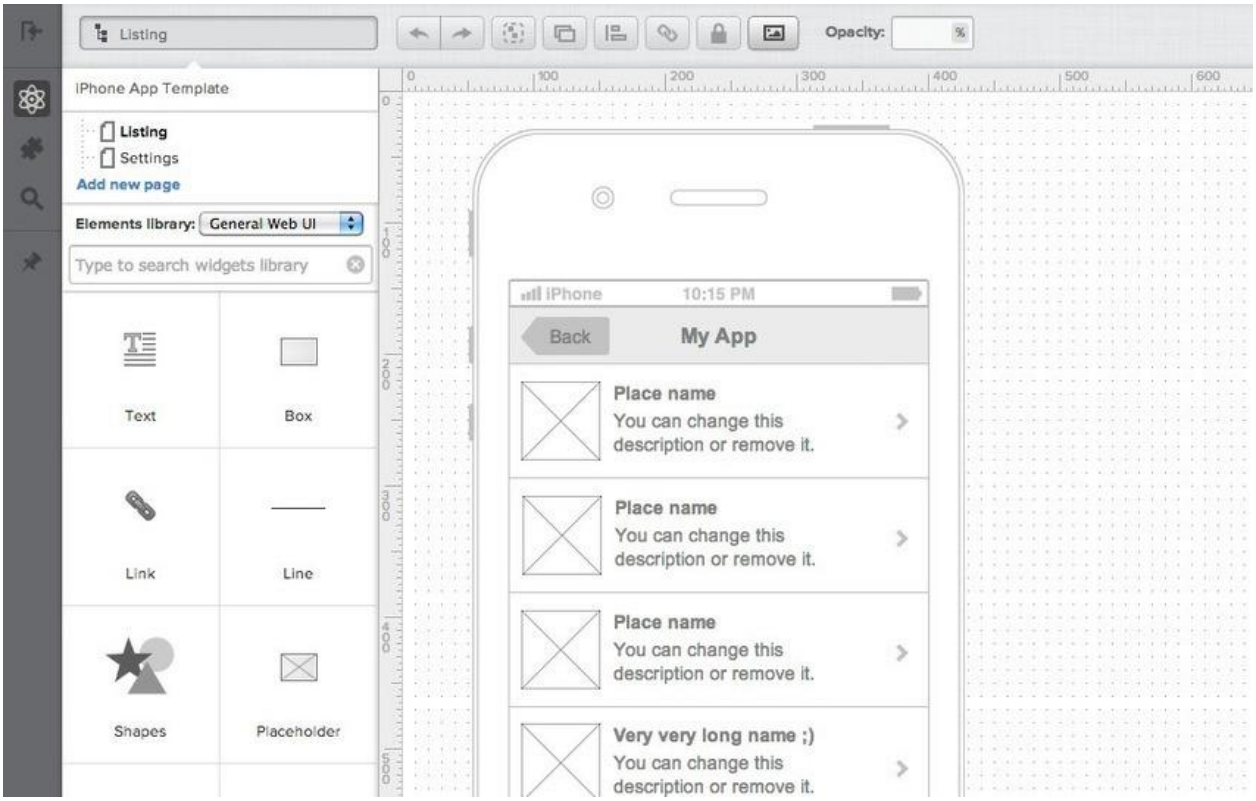
ეს არის ერთ-ერთი ყველაზე პირველი პროგრამა, რომლის დანიშნულება არის სტრუქტურების და პროტოტიპების აგება. Axure ითვლება ინდუსტრიის სტანდარტად და პროფესიონალებისთვის საუკეთესო ინსტრუმენტია. ამ პროგრამასაც აქვს გამზადებული ელემენტები, მაგრამ სამუშაო პროცესი უფრო მორგებულია დიდ პროექტებზე და პროტოტიპების შექმნაზე (პროტოტიპებს შემდეგ თავში განვიხილავთ). პროგრამა ხელმისაწვდომია Windows და Mac პლატფორმებზე.




**UXPin - www.uxpin.com**

ეს არის ვებ აპლიკაცია. ანუ პროგრამა, რომელიც ხელმისაწვდომია ნებისმიერი პლატფორმიდან ბრაუზერის მეშვეობით. აპლიკაციის გამოსაყენებლად უნდა გაიაროთ რეგისტრაცია. აქვთ უფასო საცდელი პერიოდი, რომლის განმავლობაში შეგიძლიათ გამოსცადოთ აპლიკაციის შესაძლებლობები.

ვებ აპლიკაციის ყველაზე დიდი უპირატესობა არის ის რომ ფაილები არის შენახული “ღრუბელში” და არა თქვენს ლოკალურ კომპიუტერში. შესაბამისად, თქვენ შეგიძლიათ იმუშაოთ ნებისმიერ კომპიუტერზე და ყოველთვის გექნებათ წვდომა თქვენი პროექტის ფაილებთან. ასევე, აპლიკაციას აქვს საშუალება სტრუქტურების გაზიარებისა, რომელიც გაძლევთ საშუალებას აჩვენოთ თქვენი სტრუქტურა გუნდის წევრს ან/და დამკვეთს ინტერნეტის მეშვეობით და იქვე მიიღოთ კონსტრუქციული კომენტარი.




ჩვენ განვიხილეთ მხოლოდ ყველაზე პოპულარული ინსტრუმენტი. რა თქმა უნდა, არსებობს სხვა საშუალებებიც, რომელიც შეგიძლიათ გამოიყენოთ თვენი პროექტისთვის. ყველა საშუალებას ერთი ძირითადი პრინციპი აქვს - ინტერფეისის სტრუქტურის მარტივად აგების საშუალება.

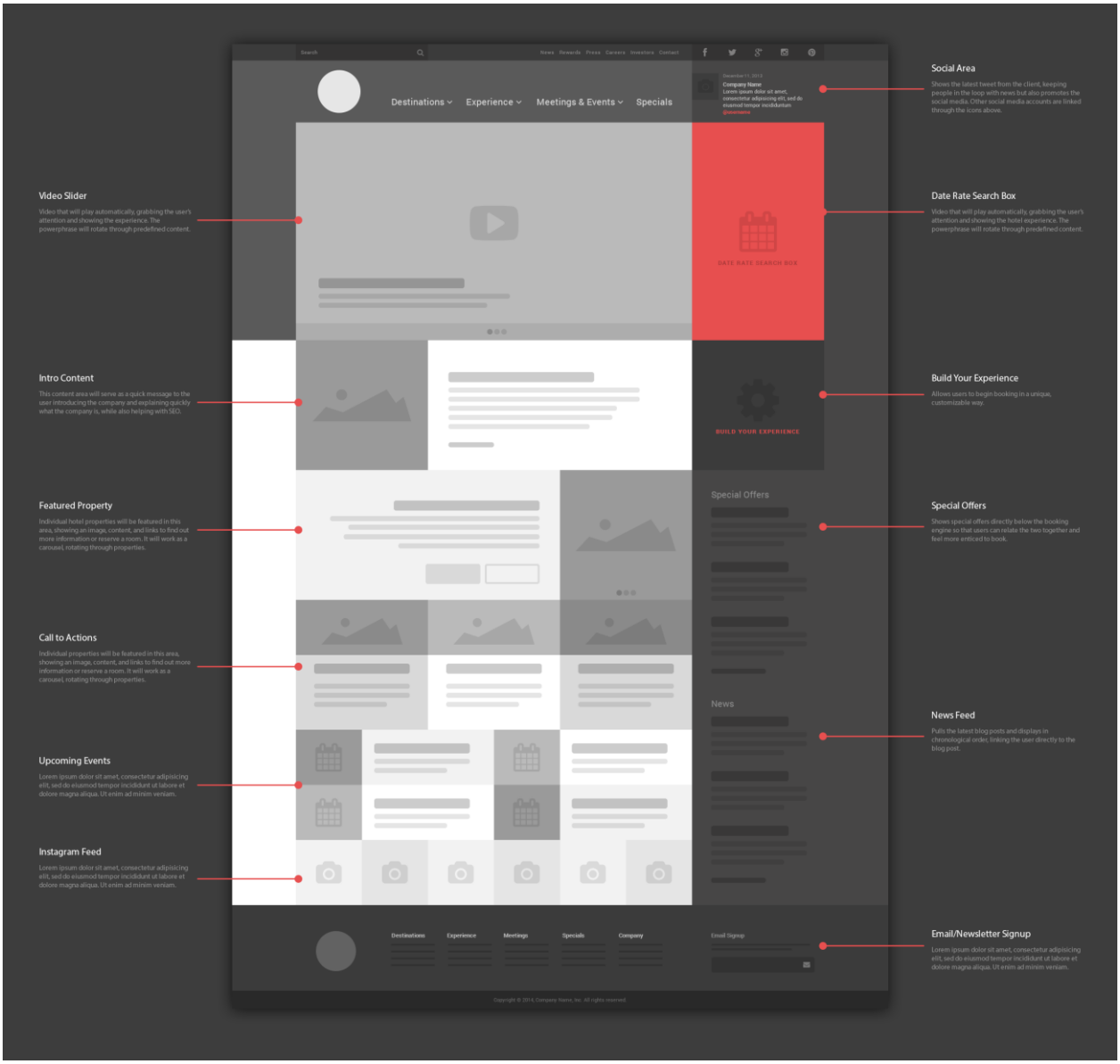
მიუხედავად იმისა თუ რომელ საშუალებას აირჩევთ თქვენი პროექტისთვის, გაითვალისწინეთ შემდეგი დამატებითი რჩევები:

- არ არის აუცილებელი სტრუქტურა ლამაზად გამოიყურებოდეს, მთავარია ყველა დაინტერესებული პირისთვის გასაგები იყოს ელემენტების განლაგება.

- სტრუქტურაში უნდა იყოს მხოლოდ ის ელემენტები, რაც უნდა გამოჩნდეს ვებ საიტის გვერდებზე. თუ გრჩებათ თავისუფალი ადგილი, ნუ შეავსებთ “რაღაცით”.

- შემუშავებული სტრუქტურა სხვას უნდა აჩვენოთ, ამიტომ ეცადეთ მნიშვნელოვანი დეტალები აღნიშნოთ კომენტარებით

- დამკვეთთან ურთიერთობაში: სანამ სტრუქტურას აჩვენებთ, ახსენით მისი დანიშნულება და წინასწარ გააფრთხილეთ რომ ის არ არის საბოლოო დიზაინის მაკეტი. სტრუქტურა უნდა გამოიყურებოდეს “უხეშად”.




**სავარჯიშო**

შეადგინეთ თქვენი ვებ საიტის სტრუქტურა ქაღალდზე.

### 3.4.4. პროტოტიპი

სწავლის შედეგის შესაბამისი თემატიკა

- პროტოტიპის დანიშნულების განხილვა
- პროტოტიპის სახეობების განხილვა
- მომხმარებლის ტესტირება

ჩვენ უკვე შევისწავლეთ მომხმარებელი და მომხმარებლის მოთხოვნები. შევიმუშავეთ პოტენციური ვებ საიტის გამოყენების სცენარები. მოვამზადეთ ინფორმაციული არქიტექტურა და ავაწყვეთ გვერდების სტრუქტურები.

ჩვენ შეგვიძლია გადავიდეთ ვებ საიტის დიზაინის აწყობის სტადიაზე, ვებ საიტიც ავაწყოთ და მივაწოდოთ მომხმარებელს. მაგრამ…

- შესაძლებელია რომ გვერდის სტრუქტურები სუბიექტური გამოგვივიდა და გუნდის წევრების გარდა დანარჩენებისთვის გაუგებარი იქნება?

- შესაძლებელია რომ ტერმინოლოგია რომელიც გამოყენებული გვაქვს არის სფეროს ექსპერტებისთვის გასაგები და სხვებისთვის გაუგებარი იქნება?

- შესაძლებელია რომ დამკვეთის მიერ შემუშავებულ ტექსტებში შეცდომებია?

პასუხი ყველა ამ კითხვაზე არის - დიახ, შესაძლებელია და დიდი ალბათობით ფაქტია. ამიტომ, არსებობს პროტოტიპების შემუშავების ეტაპი, სადაც გვეძლევა საშუალება შევიმუშაოთ ვებ საიტის ან მისი რომელიმე ნაწილის დროებითი, სამაგალითო ვერსია და გადავამოწმოთ ჩვენ პოტენციურ მომხმარებლებთან, თუ ჩვენი ყველა ჰიპოთეზა გამართლებულია.

**რა არის ვებ საიტის პროტოტიპი**

ვებ საიტის პროტოტიპი არის ქაღალდზე ან რომელიმე პროგრამაში შემუშავებული ვებ საიტის ინტერაქტიული ნიმუში, რომელიც კონკრეტულად ასახავს ვებ საიტის რომელიმე ფუნქციის დანიშნულებას ან/და მაქსიმალურად რეალისტურად ასახავს ვებ საიტის ყველა შესაძლებლობას. პროტოტიპი გამოიყენება მომხმარებლის შესასწავლად, რათა დაგეგმარების პროცესში მიღებული ყველა გადაწყვეტილება გადამოწმდეს რეალურ გამოყენებაში.

სწორად შემუშავებული პროტოტიპი და მომხმარებლის ტესტირების პროცესი მოგცემთ საშუალებას მიიღოთ მომხმარებლების სხვადასხვა პოტენციური ჯგუფებიდან, მნიშვნელოვანი ინფორმაცია, ვებ საიტის დასახვეწი დეტალების შესახებ.

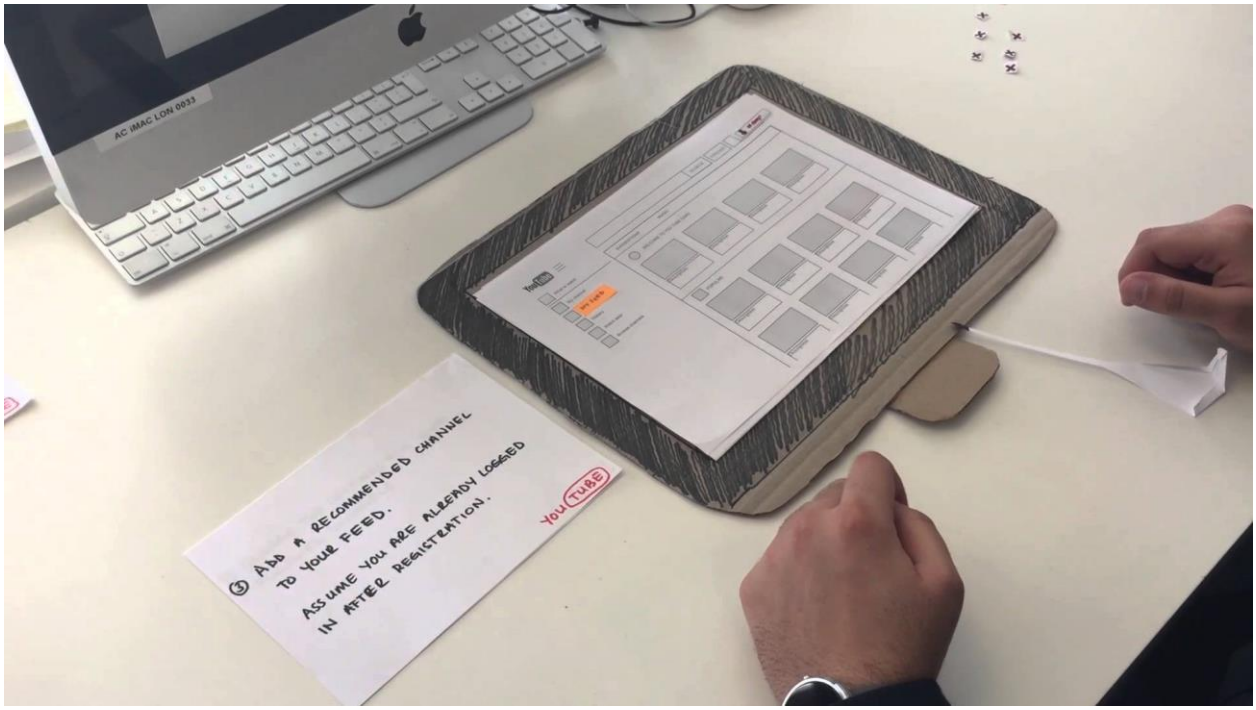
ერთის მხრივ, პროტოტიპის შემუშავება შესაძლოა მოგეჩვენოთ როგორც დამატებითი სამუშაო პროცესი, რომელიც საჭიროებს დროს. განსაკუთრებით რთულია ამ პროცესის დანერგვა დაკვეთით მიღებულ პროექტზე, რადგან შესრულების ვადა შეზღუდულია და პროექტის ბიუჯეტი შესაძლოა არ იძლეოდეს ამ სამუშაოების შესრულების შესაძლებლობას.

პროტოტიპების შემუშავება ყველა ტიპის პროექტზე არის რეკომენდირებული, ვინაიდან ხარვეზების აღმოჩენა და გასწორება პროექტის დასრულების და გამოქვეყნების შემდგომ, ბევრად უფრო მეტ დროის მოითხოვს და შესაბამისად ფინანსურ ხარჯებსაც. ამიტომ, შეეცადეთ პროტოტიპის ტესტირება მოახდინოთ ყველა ტიპის პროექტზე, უბრალოდ შეარჩიეთ ტესტირების სხვადასხვა მეთოდიკა, პროექტის სირთულიდან და მასშტაბიდან გამომდინარე.

**პროტოტიპების სახეობები**

არსებობს სამი ძირითადი პროტოტიპის სახეობა: ქაღალდის, HTML პროტოტიპი და ინტერაქტიული პროტოტიპი, რომელიც შემუშავებულია სხვადასხვა პროგრამული უზრუნველყოფის მეშვეობით.

- **ქაღალდის პროტოტიპი**
  ეს ყველაზე მარტივი და სწრაფი მეთოდია. ქაღალდზე ან რომელიმე პროგრამაში შემუშავებული და შემდგომ ამობეჭდილი გვერდების სტრუქტურები ეძლევა მომხმარებელს გადასამოწმებლად..




- **HTML პროტოტიპი**
  არ არის საჭირო რომ ვებ საიტი სრულად აწყობილი გქონდეთ იმისათვის რომ მოახდინოთ მისი გადამოწმება. საკმარისია მხოლოდ საჭირო გვერდების ესკიზები გადააბათ ერთმანეთს HTML ბმულების მეშვეობით. ეს მეთოდი უფრო რეალისტურია მომხმარებლისთვის, ვინაიდან დავალებას კომპიუტერთან ასრულებს, ანუ იმ გარემოში სადაც რეალურად გამოიყენებდა თქვენს ვებ საიტს.

- **ინტერაქტიული პროტოტიპი**
  ინტერაქტიული პროტოტიპი ტესტირების პროცესიც და შედეგიც HTML პროტოტიპის ანალოგია. სხვაობა მხოლოდ იმაშია რომ ინტერაქტიული პროტოტიპის შემუშავებისთვის არ არის საჭირო HTML გვერდების შემუშავება. საკმარისია პროტოტიპის შემუშავების რომელიმე პროგრამა გამოიყენოთ რათა მიიღოთ სასურველი ინტერაქციის შედეგი. ზევით ნახსენები Axure (www.axure.com) იძლევა ამის შესაძლებლობას. მობილურ მოწყობილობებზე შეგიძლიათ გამოიყენოთ POP (www.popapp.in) აპლიკაცია.

**მომხმარებლის ტესტირების პროცესი**

მიუხედავად შერჩეული პროტოტიპის სახეობისა, ტესტირების პროცესში ყოველთვის ფიგურირებს მომხმარებელი, რომელიც გვაწვდის პროტოტიპის შესახებ სასარგებლო ინფორმაციას.

წინასწარ მომზადებული სცენარის საფუძველზე, მომხმარებელი, რომელიც პოტენციური მომხმარებლის რომელიმე ჯგუფს უნდა წარმოადგენდეს, ასრულებს სცენარში მითითებულ დავალებას პროტოტიპის გამოყენებით.

ამ პროცესში ჩვენ ვთხოვთ მომხმარებელს:

- **ნუ მოგვაქცევს ყურადღებას, როგორც პროცესის დამმკვიდრებლებს და ეცადოს დავალება შეასრულოს დამოუკიდებლად, კითხვების გარეშე.**

  *ამით ჩვენ რეალობის სიმულირებას ვაკეთებთ. რეალური მომხმარებელი ხშირ შემთხვევაში მარტოა კომპიუტერის წინაშე და ჩვენ გვაინტერესებს, რამდენად გასაგები იქნება ჩვენი გადაწყვეტილება მომხმარებლისთვის დახმარების გარეშე.*

- **იფიქროს ხმამაღლა**

  *ჩვენ გვაინტერესებს რას ფიქრობს მომხმარებელი როცა ხედავს ჩვენ გადაწყვეტილებას. გასაგებია თუ არა მისთვის ვებ საიტის კონკრეტული ტექსტური ნაწილი ან ღილაკის წარწერა. რატომ იღებს ის კონკრეტულ გადაწყვეტილებას რომ დააჭიროს რომელიმე ღილაკზე.*

- **შეასრულოს კონკრეტული დავალება, სცენარის მიხედვით.**

  *თვენი ვებ საიტის შესაძლოა ძალიან საინტერესო იყოს მომხმარებლისთვის, მაგრამ ტესტირების ძირითადი დანიშნულება არის პრობლემების აღმოჩენა, რომელიც დაკავშირებულია კონკრეტულ პროცესებთან. თქვენი პრიორიტეტი ყოველთვის უნდა იყოს სცენარი. მიეცით მომხმარებელს ცოტა თავისუფლება და ჩაინიშნეთ ზოგადი კომენტარები, მაგრამ შეახსენეთ რა დავალებას ასრულებს ის და გააგრძელეთ ტესტირება.*

მომხმარებლის და პროტოტიპის ტესტირების პროცესში, გაითვალისწინეთ შემდეგი რჩევები:

- რეკომენდირებულია ოთახში იყოს მხოლოდ მომხმარებელი და თქვენი გუნდის ერთი წარმომადგენელი. დამკვიდრებელი უნდა ინიშნავდეს მომხმარებლისგან ყველა მნიშვნელოვან კომენტარს.

- თუ არის შესაძლებელი, პროცესი შეგიძლიათ ჩაწეროთ ვიდეოზე, ვინაიდან ტექსტის შედეგები მნიშვნელოვანი უნდა იყოს ყველა სხვა გუნდის წევრისთვის და ჩანიშვნებით, შესაძლებელია ვერ გადასცეთ მომხმარებლის ემოცია.

- თუ მომხმარებელი გაჩერდა და თავისით ვერ ხვდება რა ნაბიჯი უნდა გადადგას პროცესში, ჩაინიშნეთ, პროცესის რომელ საფეხურზე გაჩერდა მომხმარებელი და მიეხმარეთ, რათა არ გააღიზიანოთ მომხმარებელი.

- ხარვეზების 80-90% აღმოჩენისთვის, საკმარისია ხუთი განსხვავებული ტიპის მომხმარებლის ტესტირება. ამასობაში შეამჩნევთ რომ მომხმარებლები ერთ და იგივე ხარვეზებს პოულობენ.

- ნუ დააკავებთ მომხმარებელს დიდი ხნით. 15 წუთის შემდეგ მომხმარებლის მოტივაცია იკლებს და მიიღებთ არასასურველ შედეგს. იყავით რეალისტური. რამდენი ხნით გაჩერდება რეალური მომხმარებელი თქვენს ვებ საიტზე?

- თუ მომხმარებელი კითხვით მოგმართავთ, ეცადეთ კითხვით უპასუხოთ. მაგალითად “ამ გვერდზე რა უნდა გავაკეთო?” - “თქვენი აზრით ამ გვერდზე რა უნდა გააკეთოთ?”.

- ყველა მომხმარებელი სხვადასხვა სისწრაფით მუშაობს. ნუ აჩქარებთ ან/და ნუ ანელებთ მომხმარებლის სიჩქარეს. თუ მომხმარებელი ნელა ასრულებს თქვენ დავალებას, მოითმინეთ. ის აუცილებლად მოგაწვდით პროცესში სასარგებლო კომენტარს.

- თქვენ თვითონ გადაამოწმეთ პროტოტიპი დავალების მიხედვით, მანამ მომხმარებელს აჩვენებთ. ძალიან უხერხულია როცა პროტოტიპი არ იძლევა იმის შესაძლებლობას რაც სცენარში გიწერიათ.

**სავარჯიშო**

მოახდინეთ თქვენს მიერ შემუშავებული ქაღალდის სტრუქტურის ტესტირება მომხმარებელთან.

## 3.5. კონცეფციის შემუშავება

### 3.5.1. ინტერფეისის ელემენტები

სწავლის შედეგის შესაბამისი თემატიკა

- ინტერფეისის ელემენტების განხილვა
- ვებ ტიპოგრაფიის შესავალი
- UI Kit ელემენტების ნაკრების გაცნობა

ვიზუალური ინტერფეისი შედგება სხვადასხვა განსხვავებული ელემენტებისგან. ვებ დიზაინერის საბოლოო პროდუქტი, ვებ საიტის მაკეტი, გამოირჩევა ინტერფეისის სხვადასხვა ელემენტების ესთეტიკურად გამოყენებაში და გამოყენებით ეფექტური საერთო ინტერფეისის აგებით ამ თავში ჩვენ განვიხილავთ ძირითად ელემენტებს და მათ სპეციფიკას.

**ლოგო**

ყველა ორგანიზაციას აქვს თავისი ლოგო ან სხვა ტიპის მახასიათებელი ნიშანი. ეს სიმბოლო არის იმ ორგანიზაციის სახე, რითიც მას ცნობენ საზოგადოებაში. აქედან გამომდინარე, ვებ საიტის სტილი უნდა იყოს დაფუძნებული ლოგოს სტილისტიკაზე.

მაგალითად, Coca-Cola-ს ლოგო არის წითელი და თეთრი. ლოგოში გამოყენებულია ხელნაწერი ფონტი. ლოგოს აქვს თავისი გამოყენების წესებიც და დამკვიდრებული ტრადიცია.


ლოგოს ხშირ შემთხვევაში მოყვება თავისი გამოყენების წესები. წესებში მითითებული იქნება ლოგოს გამოყენების სხვადასხვა დასაშვები ვარიაცია, ზომები და ფერები. ასევე მითითებული იქნება ლოგოს გამოყენების დაუშვებელი ვარიაციები.

ვებ დიზაინერმა აუცილებლად უნდა გაითვალისწინოს ლოგოს გამოყენების წესები. ვებ დიზაინში, დიზაინერს არ აქვს უფლება შეცვალოს ან გადააკეთოს დამკვეთი ორგანიზაციის ლოგო. ნებისმიერი ცვლილება, რომელიც საჭიროა აუცილებლად უნდა შეთანხმდეს ლოგოს ავტორთან.

თქვენ შეგიძლიათ დამატებით მოითხოვოთ ლოგოს გამოყენების მაგალითები. დიდი ალბათობით დამკვეთმა უკვე გამოიყენა ლოგო სხვადასხვა ტიპის მარკეტინგულ პროდუქციაზე, პლაკატებზე, მაისურებზე. ზოგ შემთხვევაში ლოგოს გამოყენების მაგალითები მოყვება ინსტრუქციას.


.

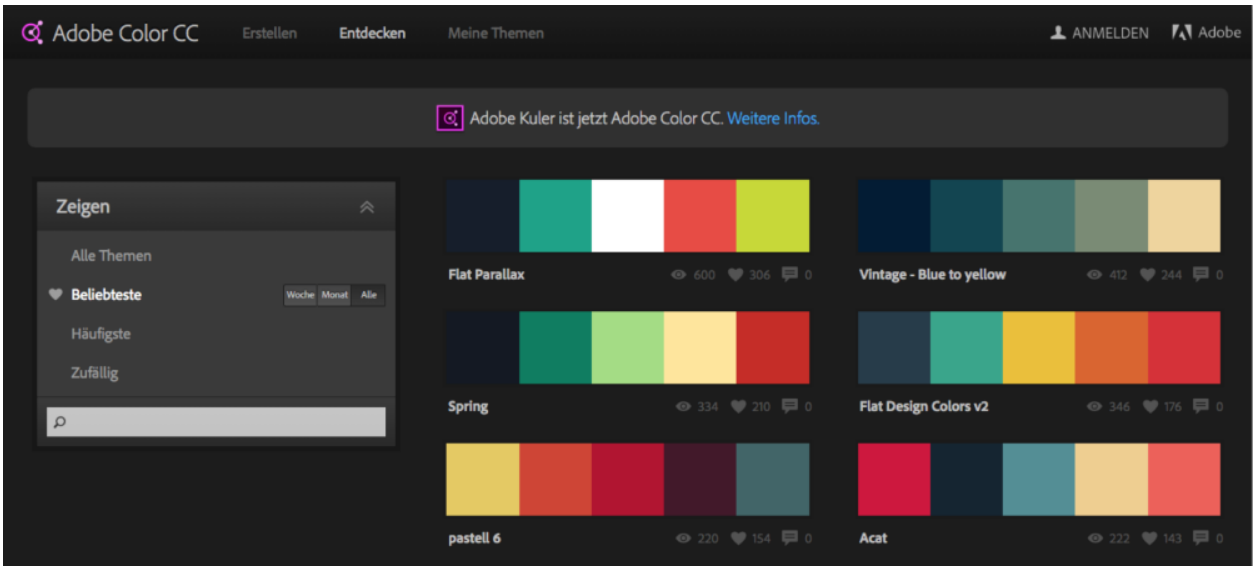
**ფერთა პალიტრა**

თუ დამკვეთის ორგანიზაციას შემუშავებული აქვს კორპორატიული სტილი, ე.წ. “Brand Book”, იმ სტილის ფარგლებში იქნება მითითებული ორგანიზაციის კორპორატიული ფერები. ხშირ შემთხვევაში, მითითებული იქნება ძირითადი ერთი ფერი და რამდენიმე დამხმარე ფერი, რომელიც თქვენ შეგიძლიათ გამოიყენოთ ვებ საიტის ინტერფეისის აგებაში.


იმ შემთხვევაში, თუ დამკვეთს არ აქვს შემუშავებული კორპორატიული სტილი, ფერები შეგიძლიათ აიღოთ ორგანიზაციის არსებული ლოგოს და არსებული მარკეტინგული მასალებიდან. დამატებით, შეგიძლიათ გამოიყენოთ შემდეგი ფერების რესურსები:

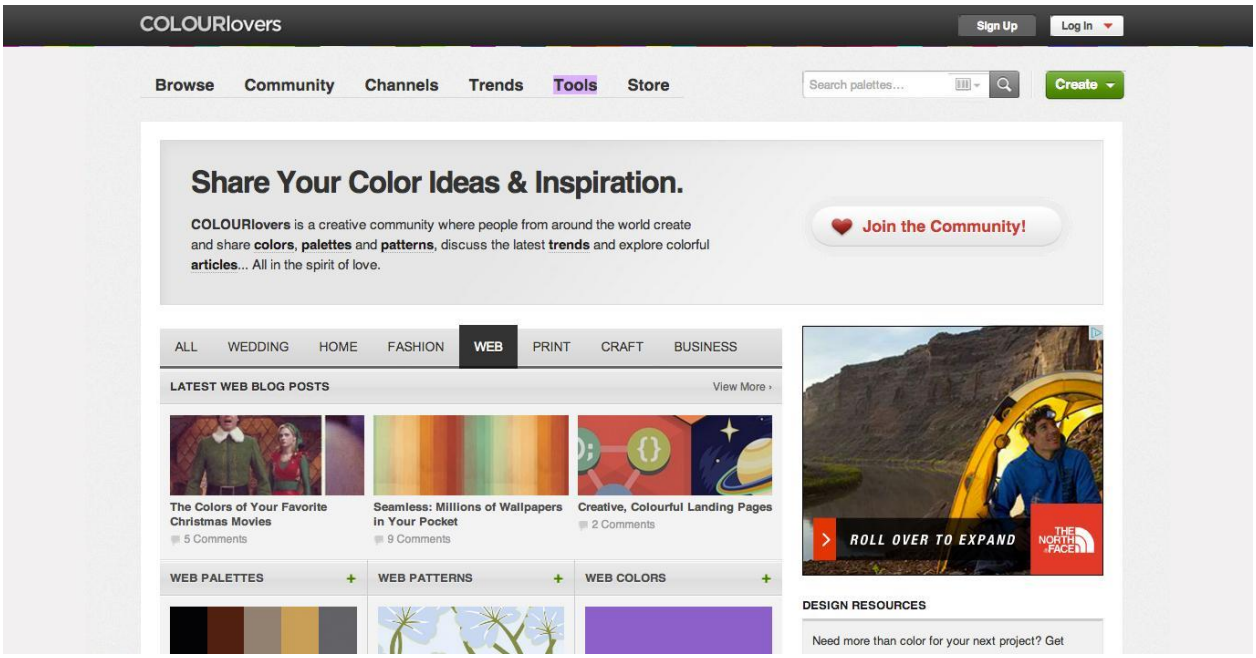
[Adobe Color]

ეს არის კომპანია Adobe-ს უფასო რესურსი, რომელიც გაძლევთ ფერთა პალიტრებთან მუშაობის საშუალებას. ძირითადი ფერის მითითებით თქვენ შეგიძლიათ აღმოაჩინოთ სხვა დამხმარე ფერები, რომელიც გამოგადგებათ ინტერფეისის სხვადასხვა ელემენტებისათვის.




[COLOURlovers]

ეს არის ფერზე ორიენტირებული სოციალური რესურსი, რომელიც გაძლევთ საშუალებას შექმნათ და გააზიაროთ თქვენი ფერთა პალიტრები. ნახოთ რა ფერები არის მოდაში და სხვა.




ფერებთან მუშაობის პროცესში, გაითვალისწინეთ შემდეგი სამი ძირითადი პრინციპი:

1. გამოყენებული ფერები უნდა უხდებოდეს ერთმანეთს. ამიტომ გამოიყენეთ ზევით მითითებული რესურსები ფერების შერჩევის პროცესში

2. კონტრასტი მნიშვნელოვანია. ყვითელი ტექსტი არ იკითხება კარგად თეთრ ფონზე. გამოიყენეთ კონტრასტის კალკულატორი და დაადგინეთ რამდენად მისაღებია თქვენს მიერ არჩეული კონტრასტი.
3. ფერები ადამიანის ემოციებზე მოქმედებს. შეისწავლეთ შემდეგი დიაგრამა და განიხილეთ ჯგუფში


**ტიპოგრაფია**

ვების დიდი ნაწილი ტექსტისგან შედგება. ამიტომ, ვებ დიზაინერმა სწორად უნდა შეარჩიოს გამოსაყენებელი ფონტი, რათა დაწერილი ინფორმაცია მომხმარებლისთვის მაქსიმალურად ხელმისაწვდომი იყოს. თუ დამკვეთის კორპორატიულ სტილში მითითებული არის გამოსაყენებელი ფონტი, გამოიყენეთ მხოლოდ ის რაც მითითებულია. სხვა შემთხვევაში თავისუფალი ხართ შეარჩიოთ ფონტი თქვენი გემოვნების მიხედვით.


გაითვალისწინეთ, ქართული დამწერლობის სპეციფიკიდან გამომდინარე, სტანდარტულად, ბრაუზერს შეუძლია გამოაჩინოს მხოლოდ UTF-8 კოდირების, ანუ უნიკოდის ფონტი, მაგალითად Sylfaen. ვინაიდან ეს არის ერთადერთი ფონტი, რომელსაც აქვს ქართული დამწერლობის მხარდაჭერა და ის სტანდარტულად მოყვება თითქმის ყველა თანამედროვე მოწყობილობას. სხვა შემთხვევებში, თქვენს მიერ შერჩეული ფონტი არ გამოჩნდება იმ კომპიუტერების ბრაუზერებში, სადაც ის ფონტი არ არის წინასწარ ინსტალირებული და ნებისმიერი სხვა ფონტი ავტომატურად იქნება შეცვლილი Sylfaen-ით.

თუმცა, არსებობს გადაწყვეტილება სახელით web fonts, რომელიც CSS-ის გამოყენებით გაძლევთ საშუალებას გამოიყენოთ ნებისმიერი ფონტი. ამ გადაწყვეტილებით თავისუფალი ხართ აირჩიოთ ნებისმიერი ქართული ფონტი, რომლის მოძიებაში შეგიძლიათ გამოიყენოთ www.fonts.ge.

ტექსტთან მუშაობისას, გაითვალისწინეთ შემდეგი რჩევები:

1. ტექსტის და ფონის ფერის მკვეთრი კონტრასტი უნდა იყოს. გამოიყენეთ კონტრასტის კალკულატორი და დაადგინეთ რამდენად მისაღებია თქვენს მიერ არჩეული კონტრასტი.
2. ტექსტის Justify განლაგება მიუღებელია ვებში, ვინაიდან სიტყვებს შორის არაპროპორციულ დაშორებას აჩენს.
3. მომხმარებელს აქვს ტექნიკური საშუალება ბრაუზერში გაზარდოს ტექსტის ზომა, გაითვალისწინეთ რომ თქვენს მიერ მითითებული ზომა შესაძლოა შეიცვალოს.
4. ეცადეთ არ გამოიყენოთ ინდუსტრიაში გავრცელებული Lorem Ipsum ტექსტი მაკეტების შემუშავებისას. რეალური ტექსტი ყოველთვის რადიკალურად განსხვავებულ შედეგს მოგცემთ.
5. მომხმარებელი ეკრანზე ტექსტს არ კითხულობს, არამედ ასკანირებს.
6. Photoshop-ში გაფორმებული ტექსტი ყოველთვის განსხვავდება HTML და CSS-ში აწყობილ ტექსტისგან (Photoshop-ში უფრო ლამაზად გამოიყურება).
7. ისწავლეთ ტრადიციული ტიპოგრაფიის საფუძვლები.

**UI Kit**

UI Kit-ი არის ინტერფეისის ელემენტების ნაკრები. ღილაკები, ფორმის ველები და სხვა ტიპის ელემენტები, რომლებიც ხშირად გხვდება სხვადასხვა ინტერფეისებში. თქვენ შეგიძლიათ მოძებნოთ და გამოიყენოთ სხვა დიზაინერის მიერ შემუშავებული UI Kit-ი ან შექმნათ თვენი პირადი ნაკრები.

UI Kit-ის დანიშნულება არის სამუშაო პროცესის ასწრაფება, რათა თქვენ არ მოგიწიოთ ყველა ელემენტის ინდივიდუალურად მომზადება და შემდგომ მათი კოპირება ინტერფეისის სხვადასხვა ნაწილებში. UI Kit-ის გამოყენება Photoshop-ში გაძლევთ ნებისმიერი ელემენტის გამოყენების საშუალებას, მარტივი drag & drop მეთოდით.

ინტერნეტში გამოქვეყნებულია ათასობით სხვადასხვა სტილის UI Kit-ი, რომელიც თქვენ შეგიძლიათ გამოიყენოთ თქვენს პროექტებში. არსებობს ასევე ცნობილი ინტერფეისების UI Kit-ები, მაგალითად iOS-ის ან/და Android-ის სტანდარტული ინტერფეისის ელემენტებით.


**თვითშეფასება**

- საჭიროების შემთხვევაში, შესაძლებელია თუ არა დამკვეთის ორგანიზაციის ლოგოს რედაქტირება?
- რა პარამეტრები არის საჭირო კონტრასტის კალკულატორისთვის?
- სად შეიძლება მოვიპოვოთ ქართული ფონტები?

### 3.5.2. მაკეტის შემუშავება

სწავლის შედეგის შესაბამისი თემატიკა

- Grid სისტემების გაცნობა
- მაკეტის შემუშავების სასარგებლო რჩევები

ვებ საიტის დიზაინი ბევრი სხვადასხვა ელემენტისგან შედგება. იმისათვის რომ პროექტში ჩართული ყველა მხარე ინფორმირებული იყოს თუ რა იქნება ვიზუალური ინტერფეისის საბოლოო სახე, საჭიროა დიზაინის მაკეტის შემუშავება.

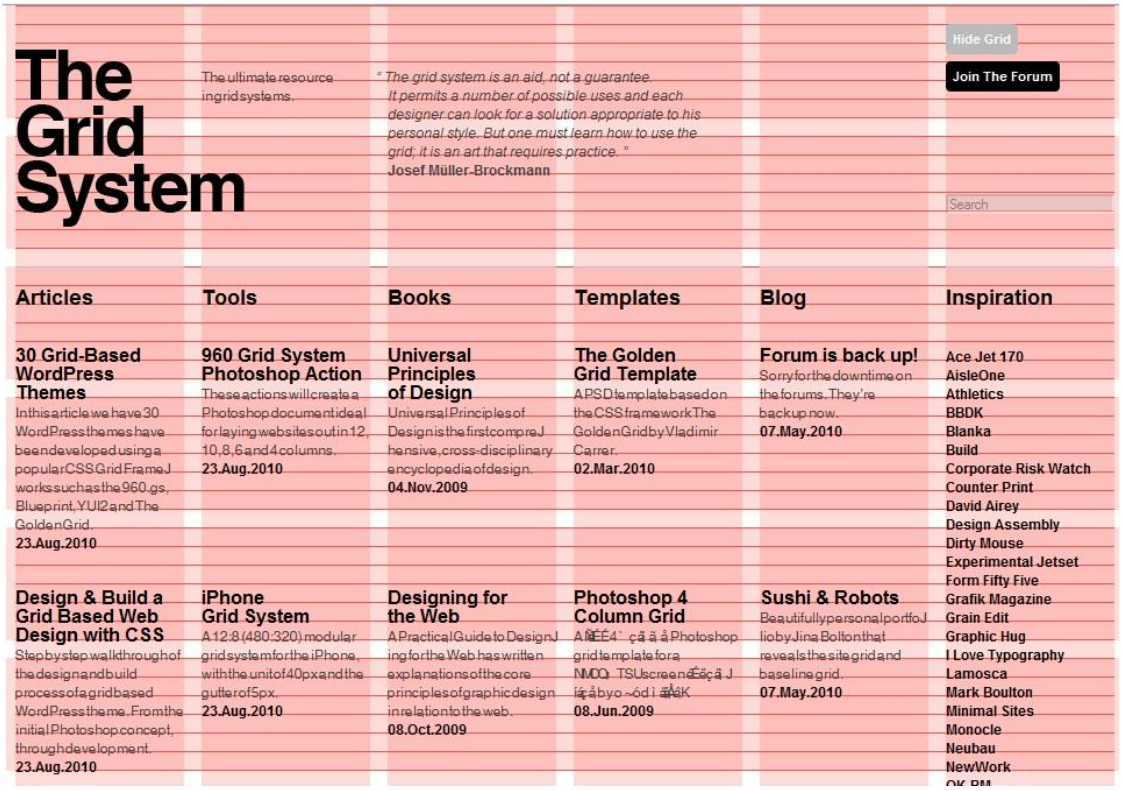
მაკეტის აწყობა ხშირ შემთხვევაში ხდება Photoshop-ში, მაგრამ თქვენ შეგიძლიათ გამოიყენოთ ნებისმიერი სხვა რასტრული ან ვექტორული რედაქტორი. მნიშვნელოვანია მხოლოდ საბოლოო შედეგი. მაკეტმა ნათლად უნდა ასახოს დიზაინის ყველა ვიზუალური გადაწყვეტილება.

მაკეტის შემუშავების პროცესში გაითვალისწინეთ, შემდეგი მნიშვნელოვანი დეტალები:

**გამოიყენეთ გრიდის (Grid) სისტემა**

გრიდი არის ვიზუალური ელემენტების განაწილების კონსტრუქცია. გრიდის გამოყენებით თქვენ გაქვთ საშუალება, განალაგოთ ვიზუალური ელემენტები პიქსელების სიზუსტით. საბოლოო ჯამში, თქვენი ვიზუალური ინტერფეისი Photoshop-იდან HTML/CSS-ში უნდა გადავიდეს, გრიდი ასევე გეხმარებათ ამ პროცესში.

Photoshop-ში მაკეტის შემუშავების დაწყებამდე, პირველ რიგში აირჩიეთ თქვენთვის სასურველი გრიდის სისტემა. როგორც ქვედა სურათზე არის ნაჩვენები, გრიდი ვარდისფერი ხაზებია, რომლის უკან ჩანს ინტერფეისის ელემენტები. გრიდი იზომება სვეტების რაოდენობით. ყველაზე პოპულარული არის 12 სვეტიანი გრიდი.



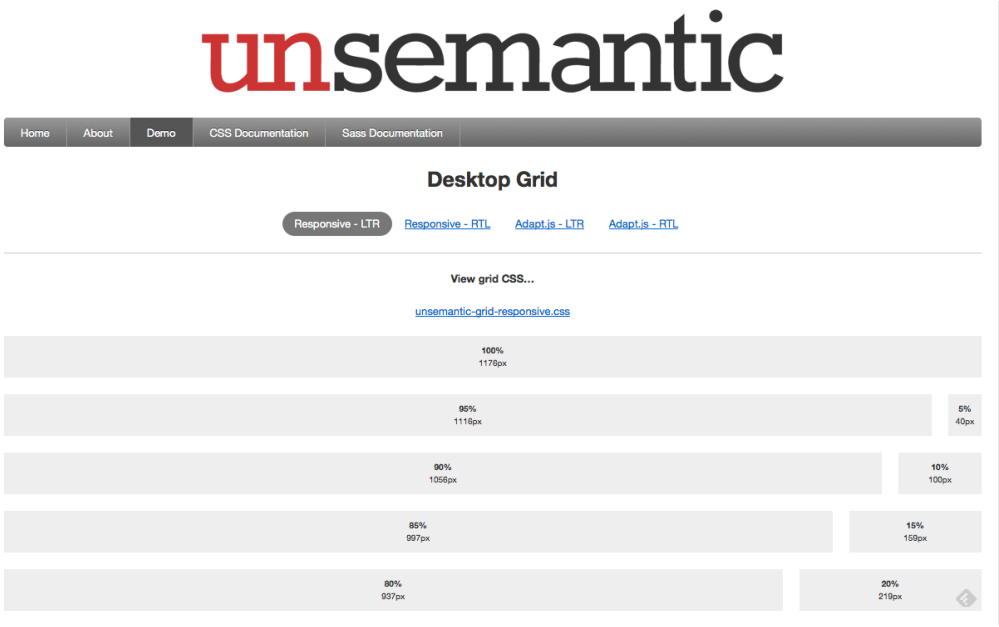

არსებობს რამდენიმე პოპულარული გრიდის სისტემა, რომელიც გამოყენებაც შეგიძლიათ. ყველას თავისი უპირატესობები აქვს, ამიტომ თქვენ შეგიძლიათ აირჩიოთ ის სისტემა, რომელიც თქვენთვის არის უფრო მისაღები.

- 960

  ფიქსირებული სიგანე, მორგებულია 960 პიქსელზე. აქვს 12 და 16 სვეტიანი კონსტრუქცია. აქვს გადმოსაწერი შაბლონები CSS-თვის, Photoshop-თვის და უამრავი სხვა რედაქტორისთვის.

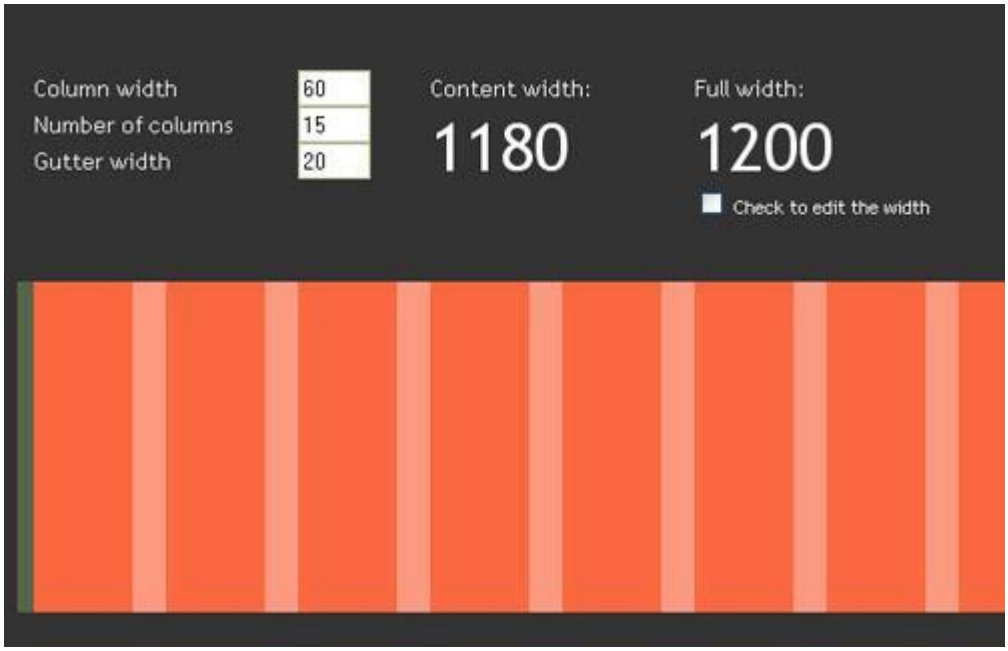

- unsemantic

  ამ სისტემას აქვს დინამიური სიგანე, რომელიც მორგებულია რესპონსიული დიზაინისთვის. გრიდის კონსტრუქცია დინამიურად იცვლება იმის და მიხედვით თუ რა სიგანის ბრაუზერი აქვს მომხმარებელს.



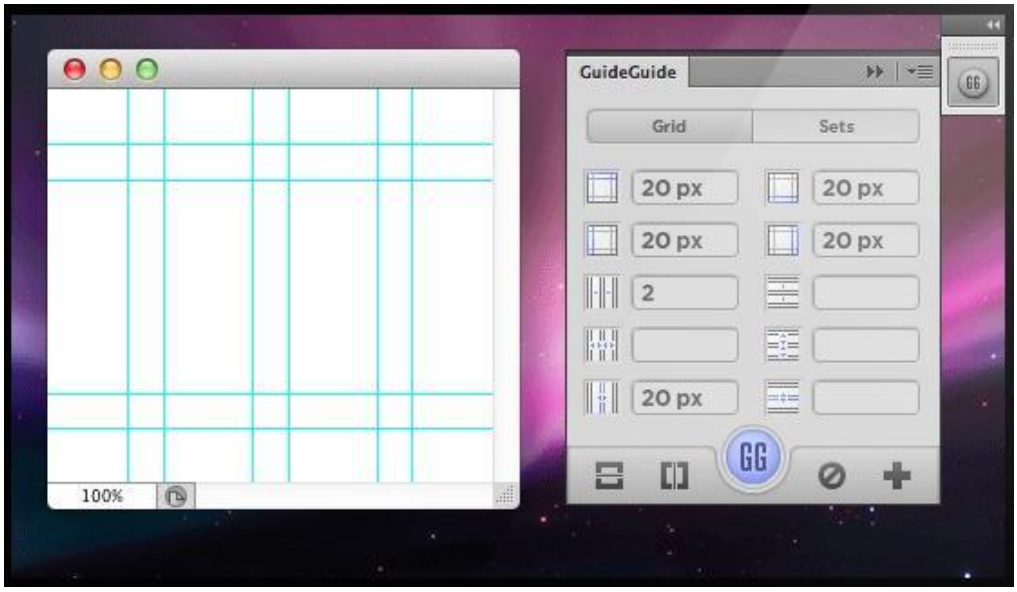

- [1200]

960 გრიდი არის შედარებით მოძველებული. მომხმარებლების უმრავლესობას აქვს მონიტორი, რომელსაც აქვს 1280 პიქსელი სიგანეში (ან მეტი) ჩვენებს საშუალება. ეს გრიდი ასევე დინამიურია და გაძლევთ საშუალებას მიუთითოთ სასურველი სვეტების რაოდენობა, მათი სიგანის და დაშორებების პარამეტრები.



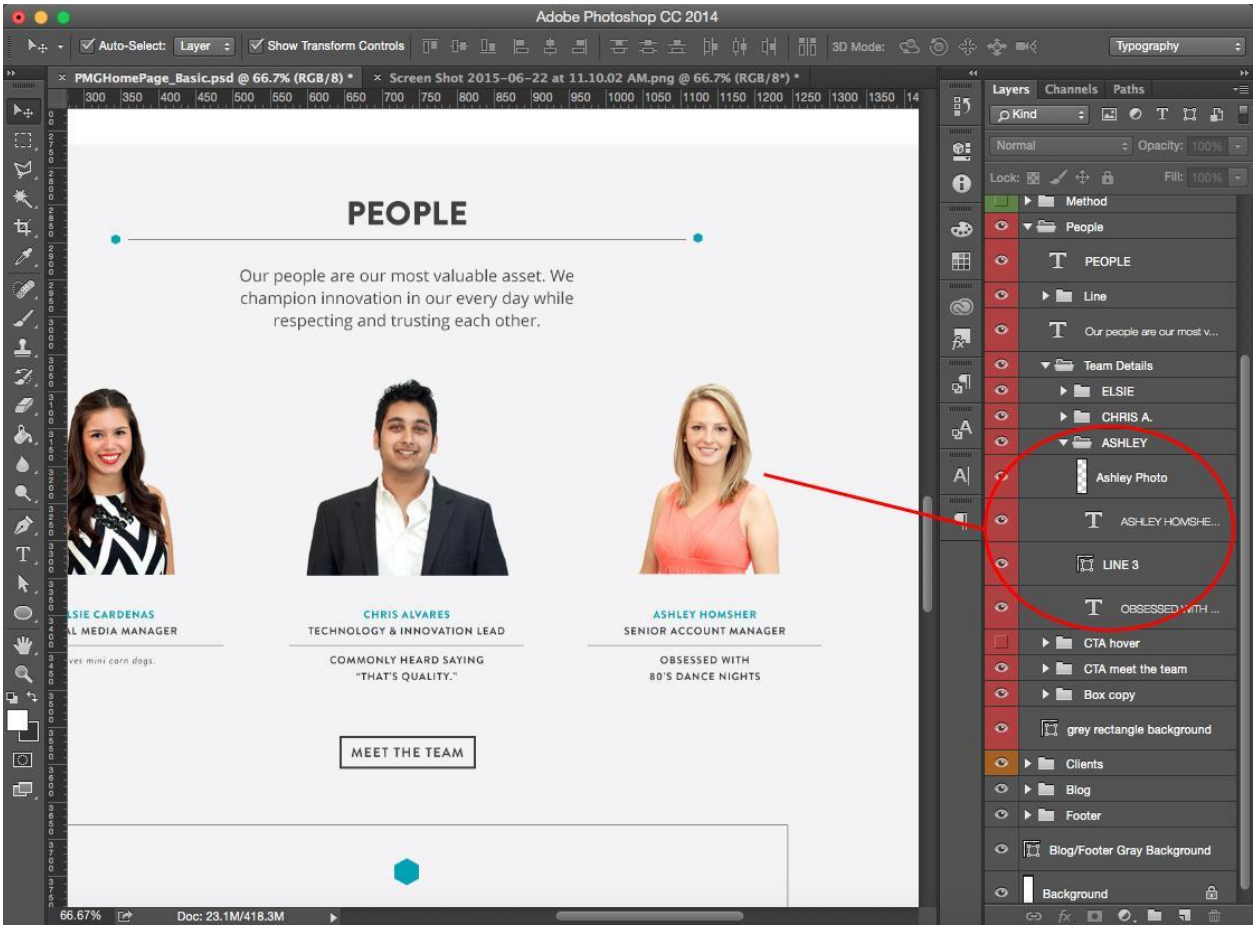

- [GuideGuide]

თქვენ ასევე შეგიძლიათ შექმნათ ნებისმიერი ტიპის გრიდი პირდაპირ Photoshop-ში. ხელით ამის კეთებას ძალიან დიდი დრო დასჭირდება, ამიტომ, არსებობს სპეციალური Photoshop-ის დანამატი, რომელიც გაძლევთ ნებისმიერი გრიდის ავტომატურად გენერაციის საშუალებას.




### შეიმუშავეთ ფენების ორგანიზების სისტემა

მაკეტის აწყობის პროცესში, სამუშაო ფაილში დაგროვდება სხვადასხვა ფენებზე განლაგებული ძალიან ბევრი ელემენტი. თვითონ მაკეტი შესაძლოა კარგად გამოიყურებოდეს, მაგრამ ძალიან მნიშვნელოვანია რომ ფენების ორგანიზების სისტემა გქონდეთ შემუშავებული.



გადაანაწილეთ ყველა თემატური ფენა საქაღალდეებში და შექმენით საქაღალდეები იერარქიული სტრუქტურა, რომელიც მოგცემთ საშუალებას მარტივად იპოვოთ ნებისმიერი სასურველი ფენა. ფენებს დაარქვით შესაბამისი სახელი, რათა მოგვიანებით მარტივად მოახდინოთ ფენის იდენტიფიცირება.

გაითვალისწინეთ რომ თქვენს მიერ აწყობილ ფაილში შესაძლოა რომელიმე თქვენ გუნდის წევრს მოუწიოს მუშაობა. დაალაგეთ ფენების და საქაღალდეების იერარქია ლოგიკურად, რათა გუნდის წევრისთვისაც იყოს გასაგები თუ რომელი ფენა სად არის შენახული.

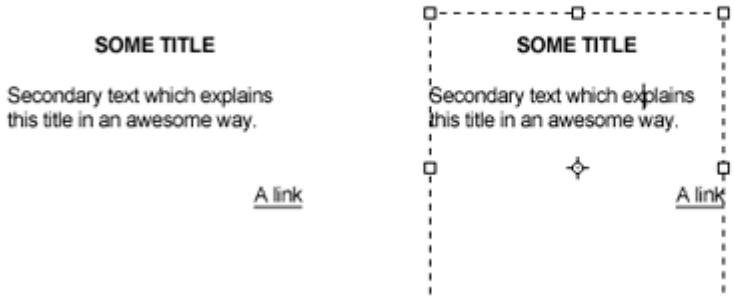
მაკეტის დამკვეთთან განხილვის შედეგად, დიდი ალბათობით თქვენ მოგიწევთ მაკეტში გარკვეული ტიპის ცვლილებების შეტანა. ორგანიზებული ფენების სტრუქტურა დაგიზოგავთ დროს და მოგცემთ საშუალებას ძალიან მარტივად მოახდინოთ ცვლილებები და წარადგინოთ მაკეტის განახლებული ვერსია.

### დამატებითი რჩევები

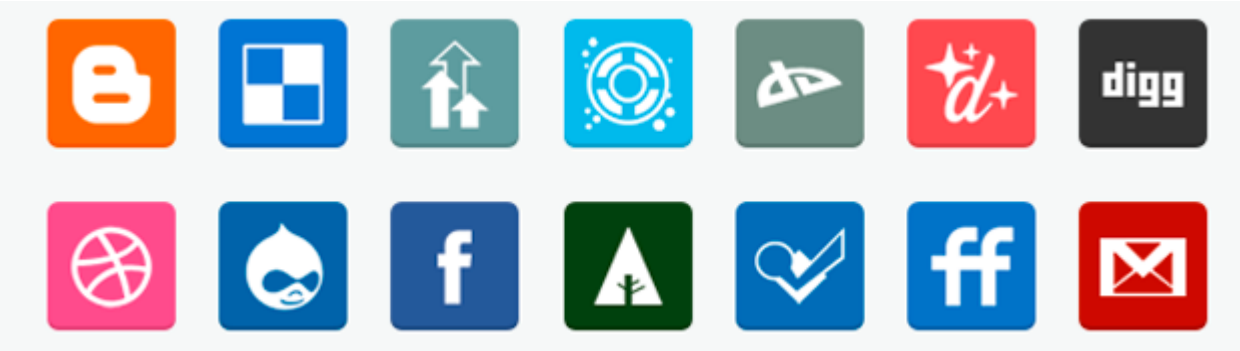
კარგი მაკეტის აწყობა ითხოვს გამოცდილებას. დროთა განმავლობაში თქვენ აუცილებლად დავხვეწავთ მაკეტის აწყობის ტექნიკას და ჩამოაყალიბებთ თქვენს პირად მეთოდიკას და სტილს ხარისხიანი მაკეტების აწყობაში. ამ ეტაპისთვის, გაითვალისწინეთ შემდეგი დამატებითი რჩევები, რომელიც გამოგადგებათ პირველივე მაკეტის აწყობაში:

- რასტრული რედაქტორის არჩევის შემთხვევაში, ყოველთვის შექმენით გეომეტრიული ფიგურები ვექტორში. ეს მოგცემთ საშუალებას ნებისმიერ დროს შეცვალოთ ფიგურის ზომა ხარისხის დანაკარგის გარეშე.

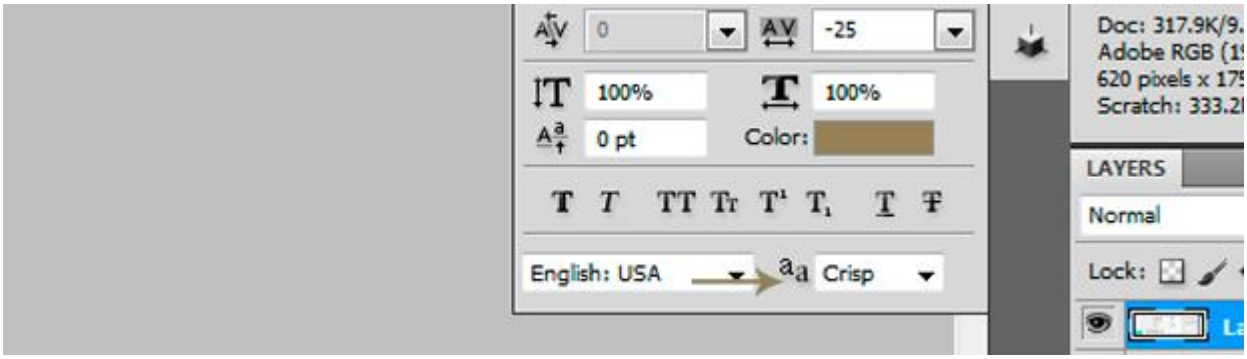
- ტექსტის შემთხვევაში, ყოველთვის გამოიყენეთ ტექსტური არეალი და არა უბრალოდ ტექსტი. იმ შემთხვევაში თუ ტექსტი შესაცვლელი გახდება, ტექსტური არეალი დინამიურად აღიქვამს ტექსტს და არ მოგიწევთ ხელით ტექსტის ხაზების გასწორება.



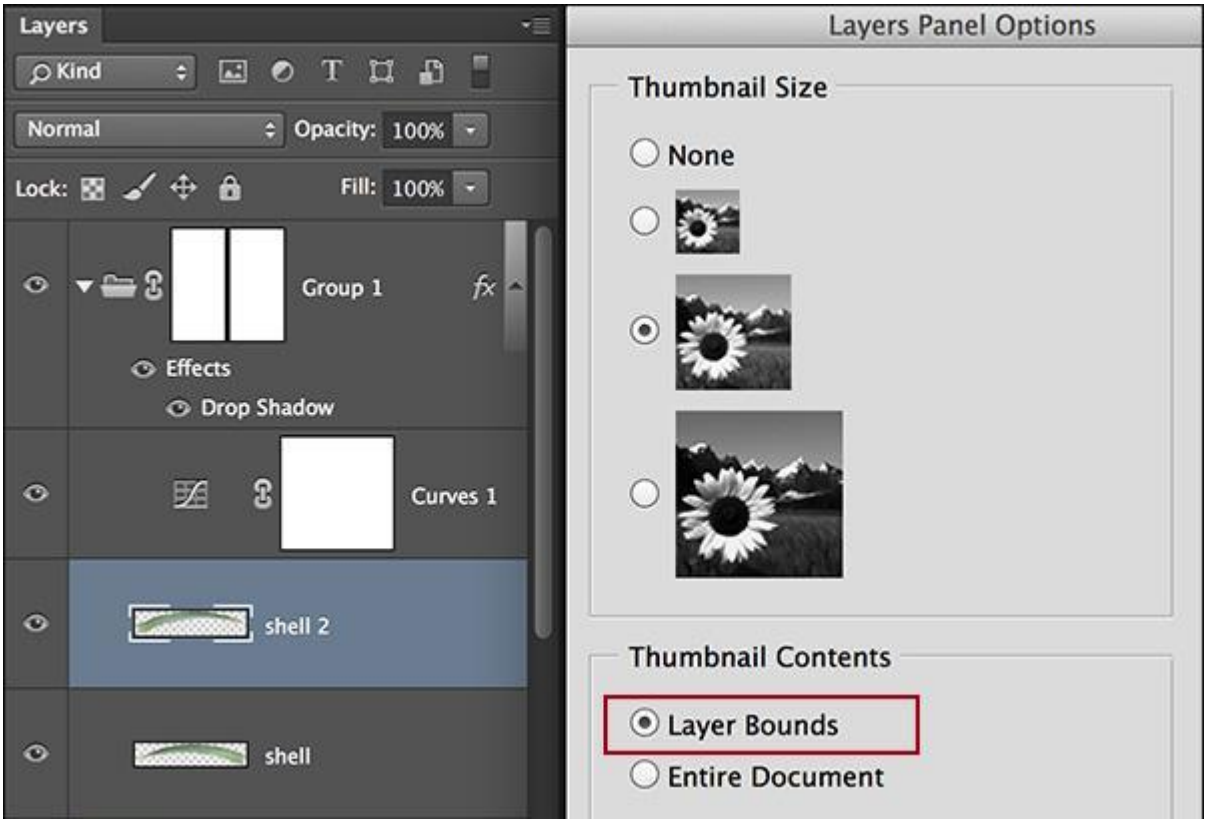

- UI Kit-ის ელემენტების გარდა, დამატებით, შეინახეთ ხშირად გამოყენებული ინტერფეისის ელემენტები Photoshop-ის რესურსებში. ამით, ერთი ღილაკის დაჭერით საჭირო ელემენტი პირდაპირ გაჩნდება თქვენს მაკეტში.




- სტანდარტულ ტექსტთან მუშაობისას, აირჩიეთ ტექსტის Crisp პარამეტრი.




Photoshop-ის ფენების პანელის პარამეტრებში, გააქტიურეთ Layer Bounds პარამეტრები, რომელიც ფენის დეტალებს გამოჩენს პანელის პატარა სურათებში


Layers
Kind
Normal
Opacity: 100%
Lock:
Fill: 100%
Group 1
Effects
Drop Shadow
Curves 1
shell 2
shell
Layers Panel Options
Thumbnail Size
None
Thumbnail Contents
Layer Bounds
Entire Document

### 3.5.3. ნაშრომის პრეზენტაცია

სწავლის შედეგის შესაბამისი თემატიკა

- პრეზენტაციის სხვადასხვა ხერხის გაცნობა

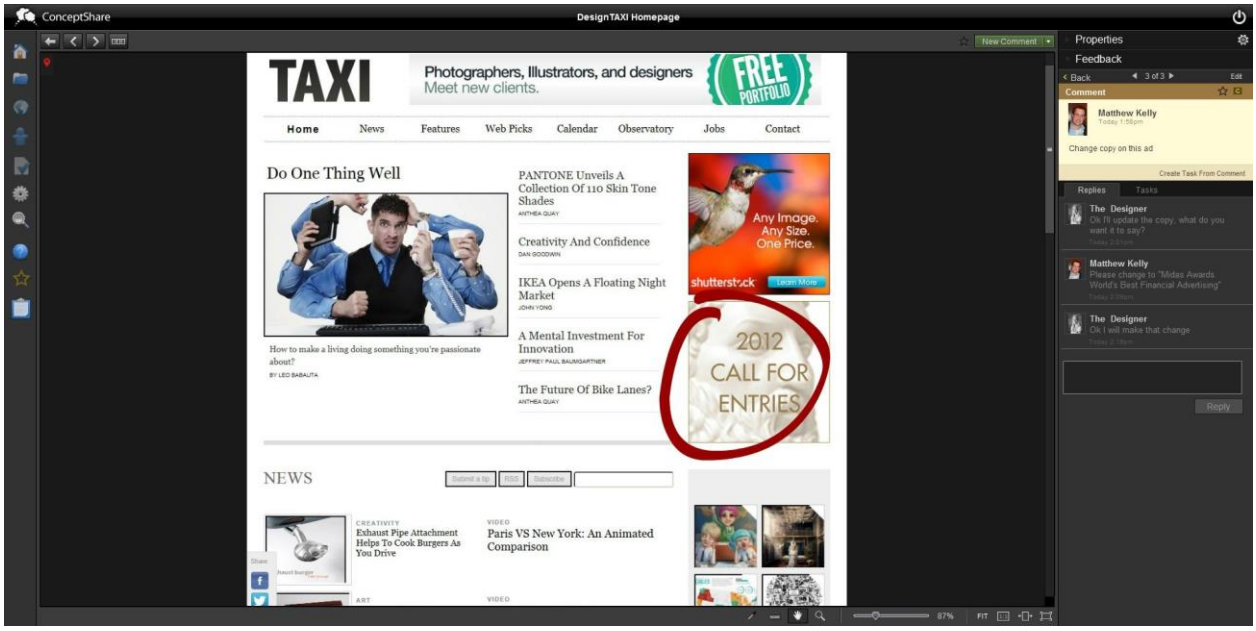
ნაშრომის პრეზენტაცია დამკვეთისთვის შესაძლოა იყოს ყველაზე რთული ამოცანა. თუ დამკვეთმა თქვენთან ერთად არ გაიარა დაგეგმარების ყველა საფეხური და არ იცის რატომ გაქვთ მიღებული სხვადასხვა გადაწყვეტილებები ვებ საიტის მაკეტის აწყობის პროცესში, მისი შეხედულება თქვენს ნაშრომზე იქნება საბოლოო ჯამში სუბიექტური.

ნაშრომის პრეზენტაციის, რამდენიმე ძირითადი მეთოდი არსებობს. ამ თავში ჩვენ ამ მეთოდებს განვიხილავთ. შეისწავლეთ თითოეული მეთოდი დეტალურად, ვინაიდან დიდი ალბათობით სხვადასხვა პირობებში მოგიწევთ სხვადასხვა მეთოდის გამოყენება.

1. **მაკეტი ჩასმულია HTML ფაილში და გაგზავნილია ბმულის სახით დამკვეთთან**
   დამკვეთი ხსნის ბმულს, რომელიც იხსნება ვებ გვერდის სახით თავის ბრაუზერში და ხედავს თქვენს მაკეტს როგორც ვებ საიტს. ვინაიდან HTML ფაილში არის მხოლოდ სურათი ჩასმული, ზოგი იკითხავს რატომ არ მუშაობს ბმულები და ტექსტური ველები. მომხმარებელი ხშირად ვერ ხვდება რომ ის სურათია და ცდილობს ინტერაქციას. ამიტომ, სასურველია რომ თქვენ გააფრთხილოთ დამკვეთი წინასწარ, რომ სურათს უგზავნით. შეგიძლიათ გამოიყენოთ შემდეგი HTML ფაილი თქვენი მაკეტის მოსამზადებლად:

```
<html>
  <head>
        <title>მაკეტის პრეზენტაცია</title>
        <style type="text/css">
        body {
        background: #1a0f1f url(img/bg.jpg) no-repeat center top;
        }

        #image {
        margin: auto;
        width: 1440px;
        padding: 0;
        }
        </style>
  </head>
  <body>
        <div id="image">
        <img src="img/image.png" title="მაკეტის პრეზენტაცია" alt="მაკეტის პრეზენტაცია"/>
        </div>
  </body>
</html>
```

2. **სპეციალიზებული ვებ სერვისის გამოყენებით**
   სპეციალურად დიზაინერებისთვის არსებობს რამდენიმე სასარგებლო რესურსი, როგორიც არის Notable ან ConceptShare. ამ სერვისების გამოყენებით თქვენ გაქვთ საშუალება ატვირთოთ მაკეტები თქვენს პირად ანგარიშზე და კონკრეტულ მაკეტზე გახსნათ წვდომა დამკვეთისთვის. სერვისის ნაწილი არის დიზაინის განხილვის დისტანციური მეთოდი, რომელიც დამკვეთს აძლევს კომენტარების დაფიქსირების საშუალებას მაკეტის კონკრეტულ ელემენტებთან დაკავშირებით. მსგავსი სერვისების გამოყენების კიდევ ერთი უპირატესობა არის მაკეტების ვერსიების ჩანაწერების შენახვა. თუ სამი განსხვავებული მაკეტი შექმენით დამკვეთის კომენტარებიდან გამომდინარე, ყოველთვის შენახული გექნებათ ყველა ვერსია და ყველა მიღებული კომენტარი. დისტანციური მომსახურების შემთხვევაში, ეს არის პრეზენტაციის ყველაზე პრაქტიკული მეთოდი.




3. **მაკეტი გაგზავნილი დამკვეთის ელ. ფოსტაზე JPG ან PNG ფაილის სახით**
   ეს არის ყველაზე მარტივი პრეზენტაციის მეთოდი. ასეთ შემთხვევაში დამკვეთი მიიღებს თქვენგან ელ. ფოსტის შეტყობინებას, რომელშიც იქნება მიმაგრებული თქვენი მაკეტი დანართის სახით. დანართის გადმოწერის და გახსნის შემთხვევაში თქვენი მაკეტი აღარ იხსნება ბრაუზერში. ოს დიდი ალბათობით გაიხსნება ფოტოების მართვის პროგრამაში, სადაც შეიცვლება სურათის სიგანე.

   ასევე, ხშირ შემთხვევაში დამკვეთი ამ შემთხვევაში ბეჭდავს მაკეტს პრინტერზე და მაკეტს ათვალიერებს ქაღალდზე. ამ შემთხვევაში დიდი ალბათობით დაკარგული იქნება მაკეტის ფერები ან/და სხვა წვრილმანი დეტალები.

   ამ მეთოდის გამოყენებისას, ყოველთვის სასურველია, დამკვეთს პარალელურად ესაუბროთ ტელეფონით. აუხსნათ როგორ უნდა გახსნას ის სურათი 100%-ის გაფართოებით და შემდგომ დეტალურად განიხილოთ მაკეტის მნიშვნელოვანი დეტალები. კიდევ ერთო რჩევა - სურათის ნაცვლად გამოიყენეთ PDF ფაილი.

4. **მაკეტის განხილვა დამკვეთთან**
   პრეზენტაციების ყველა შესაძლო ხერხებიდან, ეს ყველაზე ეფექტურია. დაგეგმეთ შეხვედრა დამკვეთთან და ერთად განიხილეთ მაკეტის დეტალები. ამ შემთხვევაში მაკეტის ტექნიკური პრეზენტაცია თვენი კონტროლის ქვეშ არის, შეგიძლია პროექტორზე ან პერსონალურ კომპიუტერზე წარმართოთ პრეზენტაცია. შეხვედრის დროს შეამჩნევთ დამკვეთის რეაქციას და იქვე გექნებათ საშუალება გასცეთ პასუხი ნებისმიერ კითხვაზე. ყოველთვის ეცადეთ აირჩიოთ ეს მეთოდი, თუ გაქვთ არჩევის საშუალება.

**სავარჯიშო**

შეადგინეთ თქვენი პირველი ვებ საიტის მაკეტის პრეზენტაცია და წარადგინეთ ის

# 4. ინტერაქტიული ელემენტების გრაფიკული დიზაინის მომზადება

## 4.1. ინტერაქცია

ინტერაქციის დიზაინი საკმაოდ დიდი დისციპლინა არის. ის აყალიბებს მომხმარებლის და ინტერფეისის ურთიერთობას. ინტერფეისი შესაძლებელია იყოს ავტომობილი, დანადგარი, და ბევრი სხვა ტიპის მოწყობილობა. კომპიუტერი არის ერთ-ერთი ინტერფეისი და ჩვენ შემთხვევაში მომხმარებლის ვიზუალური ინტერფეისთან ურთიერთობა, რომელიც ჩანს კომპიუტერის მონიტორზე, არის ზუსტად ის მიმართულება რომელსაც განვიხილავთ ამ თავში.

ვებ დიზაინში, მომხმარებელი არის ყველაზე მნიშვნელოვანი ნაწილი. მომხმარებლის გარეშე, ვებ საიტი თავის დანიშნულებას კარგავს. ვებ საიტი არის ვებ გვერდების ან სხვადასხვა ელემენტების ნაკრები. მომხმარებელი ღებულობს მისთვის სასურველ ინფორმაციას ან შედეგს ვებ საიტის გვერდებთან ან/და ელემენტებთან ინტერაქციის შედეგად.

ძალიან ხშირად, ვებ დიზაინერი იწყებს თავის სამუშაო პროცესს ინდივიდუალური გვერდების ვიზუალიზაციით. ამ დროს კი, დიზაინერმა პირველ ეტაპზე უნდა დაგეგმოს მომხმარებლის დინების პროცესი. დინების პროცესის ჩამოყალიბება გვეხმარება, მომხმარებლის მიერ გასავლელი საფეხურების წარმოდგენაში და ყველა საფეხურზე პროცესის გამარტივების შესაძლებლობების მოფიქრებაში.

“**ინფორმაციის არქიტექტურის**” თავში, ჩვენ უკვე განვიხილეთ დინების პროცესი და სცენარის ჩამოყალიბება, სადაც ვისწავლეთ მომხმარებლის სცენარის ჩამოყალიბება და იმის საფუძველზე დინების დაგეგმარება. ახლა კი ყურადღება მივაქციოთ ვებ საიტის კონკრეტული ელემენტის ინტერაქციას და მასთან დაკავშირებულ დინებას.

### 4.1.1. გამოყენებადობა (Usability)

სწავლის შედეგის შესაბამისი თემატიკა

- იუზაბილითის 10 ძირითადი პრინციპის განხილვა

გამოყენებადობა, ან როგორც ამას ინდუსტრიაში პირდაპირ ეძახიან “იუზაბილითი”, არის მომხმარებლის მიერ პროდუქტის გამოყენების ან კონკრეტული ამოცანის შესრულების სიმარტივე. გრაფიკული ინტერფეისების ინდუსტრიაში, ეს დისციპლინა მემკვიდრეობით მოხვდა პროდუქტის დიზაინიდან, სადაც ის სარეცხი მანქანების, ფოტოაპარატების და უამრავი სხვა პროდუქტების დიზაინში გამოიყენება.

გრაფიკული ინტერფეისი, რომელიც მომხმარებელს აძლევს საშუალებას, დამოუკიდებლად და ინსტრუქციის გარეშე მიაღწიოს სასურველ შედეგს და ამავდროულად თვითონ პროცესს ჰქმნის სწრაფს და მარტივს, ითვლება კარგი იუზაბილითის მქონე პროდუქტად.


მაგალითისთვის, განვიხილოთ მსოფლიოში პოპულარული ვებ საიტი amazon.com:

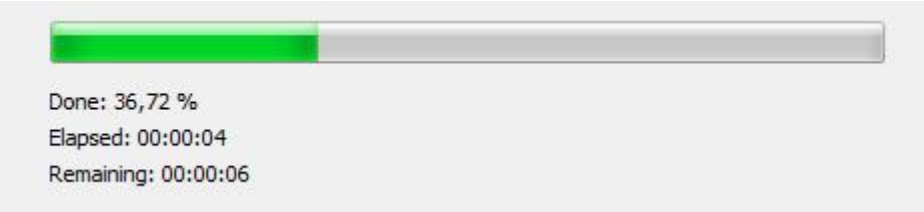
ძირითადად, ის არის ელექტრონული კომერციის ტიპის ვებ საიტი, რომელიც მომხმარებლებს აძლევს პროდუქტის შეძენის საშუალებას და შემდგომ ახდენს ამ პროდუქციის ადგილზე მიტანის მომსახურებას. ამ ვებ საიტით სარგებლობს მილიონობით მომხმარებელი, მსოფლიოს სხვადასხვა ქვეყნებში. ყველა ახერხებს მისთვის სასურველი შედეგის მიღწევას, დახმარების და ინსტრუქციის გარეშე. მომხმარებელი ბედნიერია და ამით კომპანიამ ძალიან დიდ წარმატებებს მიაღწია.

მომხმარებელი, რომელიც სარგებლობს amazon.com-ის ვებ საიტის, ახდენს ინტერაქციას ვებ საიტის სხვადასხვა ინტერფეისებთან და ამით სასურველ შედეგს ღებულობს. მიუხედავად იმისა რომ რეალურად, amazon.com არის ურთულესი, გრანდიოზულად დიდი და კომპლექსური ვებ საიტი, კომპანიის ინტერფეისების დიზაინერები ახერხებენ ათასობით სხვადასხვა ინტერფეისების და მათთან ინტერაქციის პროცესების გამარტივებას. მომხმარებლის ინტერაქცია ვებ საიტის ინტერფეისებთან არის ამ კომპანიისთვის ერთ-ერთი ყველაზე დიდი პრიორიტეტი.

ჯეიკობ ნიელსენმა (Jakob Nielsen), რომელიც არის სფეროს წამყვანი სპეციალისტი, ჯერ კიდევ 1990 წელს ჩამოაყალიბა ინტერაქციის იუზაბილითის 10 ძირითადი პრინციპი. წლების განმავლობა მან დახვეწა ეს სია, ათასობით სხვადასხვა სისტემის ტესტირებისას. მიუხედავად ტექნოლოგიების განვითარებისა, ეს 10 ძირითადი პრინციპი დღემდე აქტუალური რჩება. თქვენ შეგიძლიათ გამოიყენოთ ეს პრინციპები, თქვენი ინტერფეისის გადამოწმებისთვის, რომელიც უმაღლესი ხარისხის მისაღწევად, უნდა პასუხობდეს მათ ყველა მოთხოვნას.

1. **სისტემის სტატუსის ხილვადობა**
   დროული უკუკავშირის შედეგად, მომხმარებელი ყოველთვის უნდა იყოს ინფორმირებული სისტემის მიმდინარე სტატუსის შესახებ.

   *მაგალითად: ფაილის გადმოწერის ვიზუალური პროგრესი, დამატებითი პროცენტული და დარჩენილი დროის მონაცემებით:*

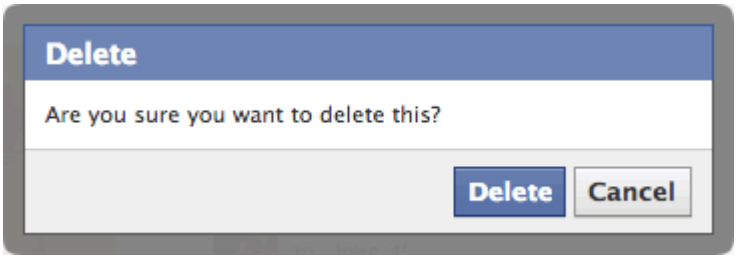



2. **სისტემის და რეალური სამყაროს შეხამება**
   სისტემა უნდა საუბრობდეს მომხმარებლის ენაზე, სიტყვებით, ფრაზებით და კონცეფციით, რომელიც მომხმარებლისთვის გასაგები იქნება. გამოიყენეთ რეალური სამყაროს კონვენციები და მოახდინეთ მომხმარებლისთვის ინფორმაციის გადმოცემა ნატურალურად და ლოგიკურად.

   *მაგალითად: ინტერფეისის ელექტრონული მაღაზიის კალათის და ნაგვის ყუთის სახელები და პიქტოგრამები აღებულია რეალური სამყაროს კონვენციებიდან.*

3. **მომხმარებლის თავისუფლება და კონტროლი**
   მომხმარებელი ხშირად უშვებს შეცდომებს ინტერფეისთან ინტერაქციაში, რის გამოც მას სჭირდება არასასურველი მდგომარეობიდან გამოსვლის მარტივი ხერხი.

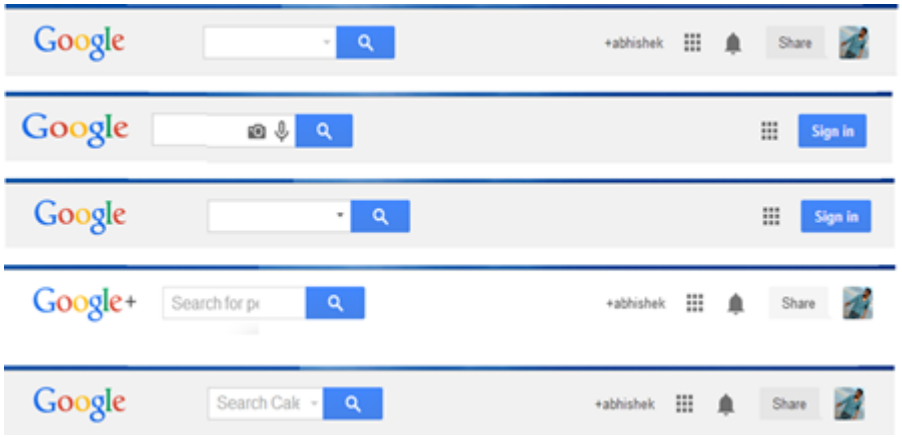
   *მაგალითად: სისტემაში undo და redo ბრძანებების მხარდაჭერა ან/და რომელიმე ბრძანების დადასტურების საშუალება*




4. **თანმიმდევრულობა და ნორმები**
   დანერგეთ თქვენი სისტემის სხვადასხვა ელემენტების ნორმები და ყველა ინტერფეისში

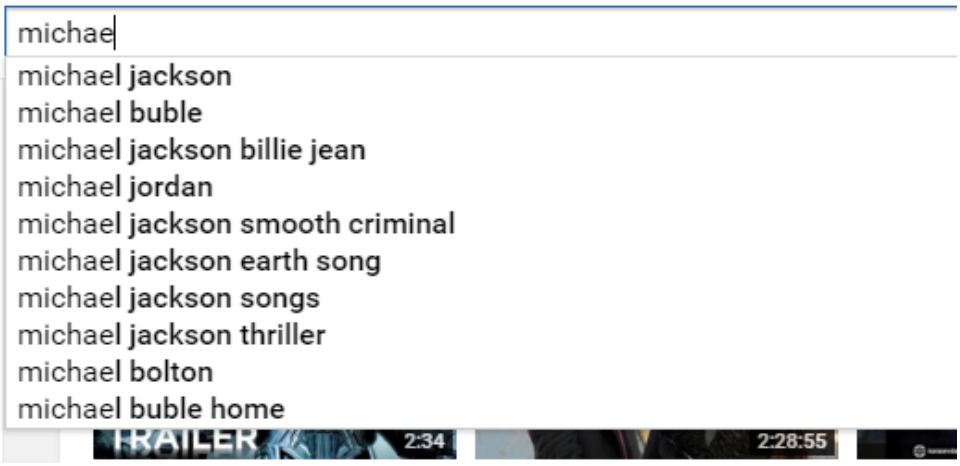
გამოიყენეთ იგივე ელემენტები თანმიმდევრულად.

*მაგალითად: Google-ის სხვადასხვა პროდუქტების ზედა ნაწილი იდენტურად გამოიყურება და არ აიძულებს მომხმარებელს სხვადასხვა პროდუქტებში სხვადასხვა ტიპის ინტერფეისთან ინტერაქციას.*




5. **შეცდომის პრევენცია**
   მომხმარებელი ხშირად უშვებს შეცდომებს ინტერფეისთან ინტერაქციაში, დაეხმარეთ მას პოტენციური შეცდომის დაშვების პრევენციაში.

*მაგალითად: YouTube-ის ძიების ველში, სისტემა ავტომატურად გთავაზობთ საძიებო ფრაზის დაბოლოებას, ვინაიდან სიტყვაში შეცდომის შემთხვევაში არასასურველი შედეგი იქნება ნაპოვნი.*

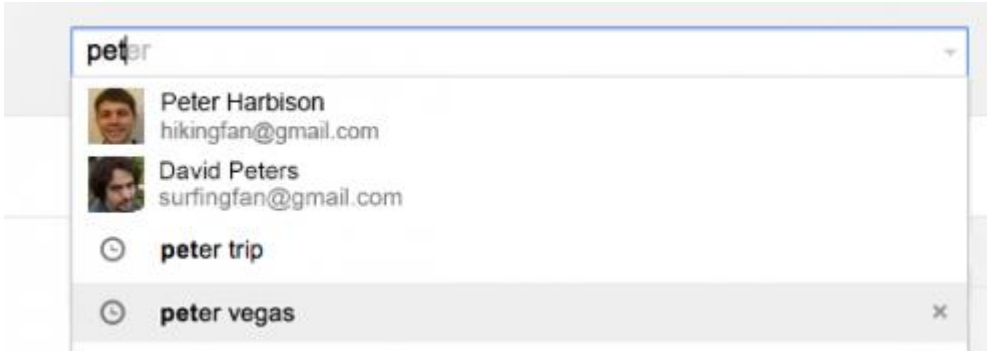



6. **გახსენებულს, ნაცნობი ჯობია**
   ნუ დატვირთავთ მომხმარებლის გონებას. მომხმარებელმა არ უნდა დაიმახსოვროს თქვენი სისტემის გამოყენების სპეციფიკა. მას ურჩევნია გამოიყენოს მისთვის უკვე ნაცნობი ინტერაქცია.

*მაგალითად: Window 8-ს პირველ ვერსიაში კომპიუტერის გათიშვის ალტერნატიული ხერხი იყო დანერგილი, რის გამოც მილიონობით უკმაყოფილო მომხმარებელი, რომელიც მიჩვეული იყო ტრადიციულ მეთოდს, Google-ში ეძებდა ინსტრუქციას.*

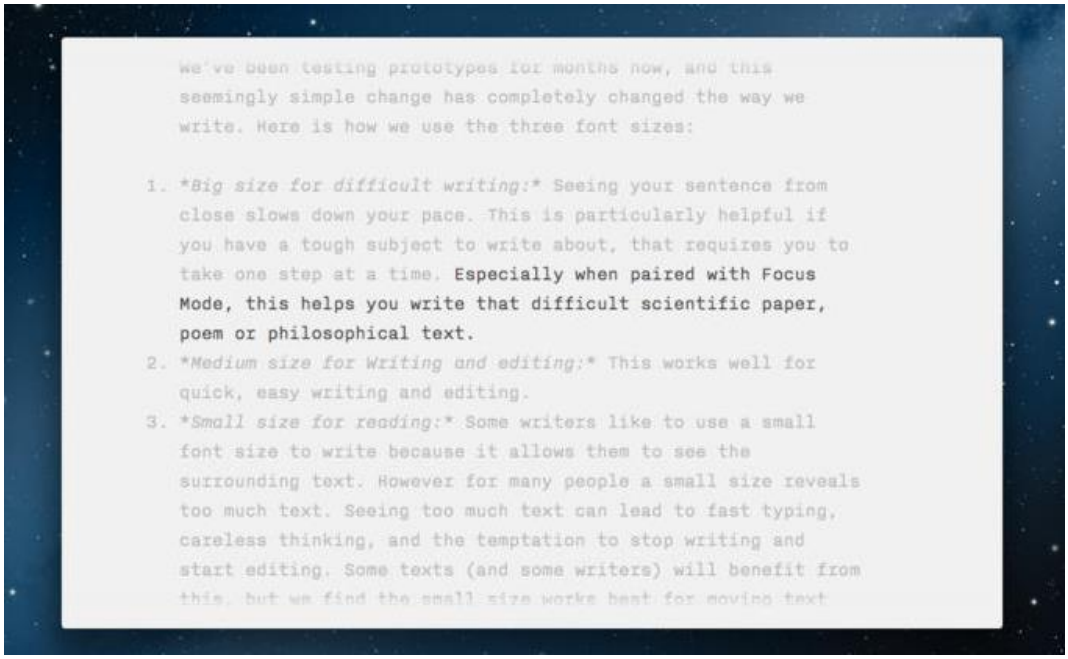

7. **გამოყენების მოქნილობა და ეფექტურობა**
   პროცესების ასწრაფების მარტივი საშუალებები, რომლებიც არ იქნება გამოყენებული გამოუცდელი მომხმარებლების მიერ, შესაძლოა საკმაოდ სასარგებლო იყოს გამოცდილი მომხმარებლებისთვის. შედეგად, ერთი ინტერფეისი ერგება ორივე ტიპის მომხმარებლის.

   *მაგალითად: GMail-ში, ძიების შედეგში ჩნდება საკონტაქტო პირების სახელები, მეტი ეფექტურობისთვის აქვე ჩანს მათი ფოტო და ელ. ფოსტის მისამართი. ასევე, ქვევით ჩანს საკვანძო ფრაზები რომელიც ადრე იყო გამოყენებული.*



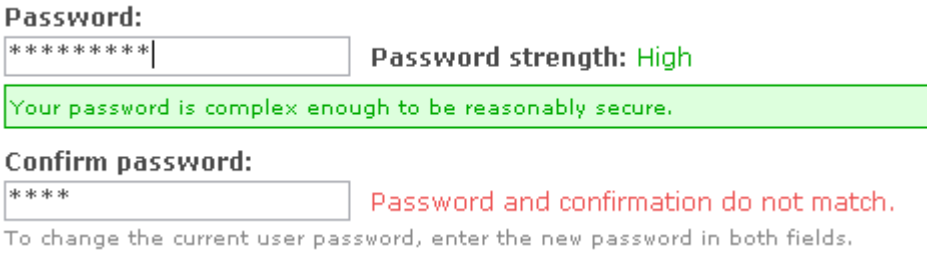

8. **ესთეტიკური და მინიმალისტური დიზაინი**
   გამოაჩინეთ ინტერფეისის მხოლოდ ის ელემენტები, რომელიც კონკრეტულ მოქმედებაში არის აქტუალური. ზედმეტი ელემენტი მომხმარებლის ყურადღებაზე მოქმედებს.

   *მაგალითად: iA Writer არის ტექსტური რედაქტორი, რომელიც არა მხოლოდ აქრობს რედაქტირების მენიუს ტექსტის აკრეფისას, ის ასევე მომხმარებლის ყურადღებას აფოკუსირებს იმ მონაკვეთზე სადაც მომხმარებელი წერს ტექსტს.*




9. **მომხმარებლის შეცდომის დაფიქსირება, ანალიზი და მოგვარება**
   სისტემური შეცდომები უნდა აისახოს მომხმარებლისთვის გასაგებ ენაზე. ახსენით რატომ მოხდა შეცდომა და შესთავაზეთ მომხმარებელს შეცდომის გასწორების გზა.

   *მაგალითად: სისტემა ატყობინებს მომხმარებელს, რომ მის მიერ შერჩეული პაროლი არის ძლიერი, ხოლო მეორე ველში პაროლი არასწორად არის შეყვანილი. ქვევით კი მოცემულია პაროლის განახლების დეტალური ინსტრუქცია.*




10. **დახმარება და ინსტრუქცია**
    სასურველია რომ სისტემას არ სჭირდებოდეს გამოყენების ინსტრუქცია, თუმცა არის შემთხვევები როცა გამოყენების ინსტრუქცია აუცილებელია. ასეთ შემთხვევებში ინსტრუქცია უნდა იყოს მოკლე, მარტივად ხელმისაწვდომი, ორიენტირებული მომხმარებლის კონკრეტულ ამოცანაზე და დეტალური.

    *მაგალითად: სარეგისტრაციო ფორმების ველებში ძალიან სასარგებლოა მოკლე ინსტრუქცია თითოეული ველისთვის, რომელიც მომხმარებელს დეტალურად აუხსნის თუ რა ინფორმაცია უნდა შეავსოს კონკრეტულ ველში. ინსტრუქცია ჩნდება კონკრეტულ ველზე გადასვლის*

*შემთხვევაში.*


**სავარჯიშო**

მოძებნეთ და წარადგინეთ განხილული 10 პრინციპის სხვა მაგალითები.

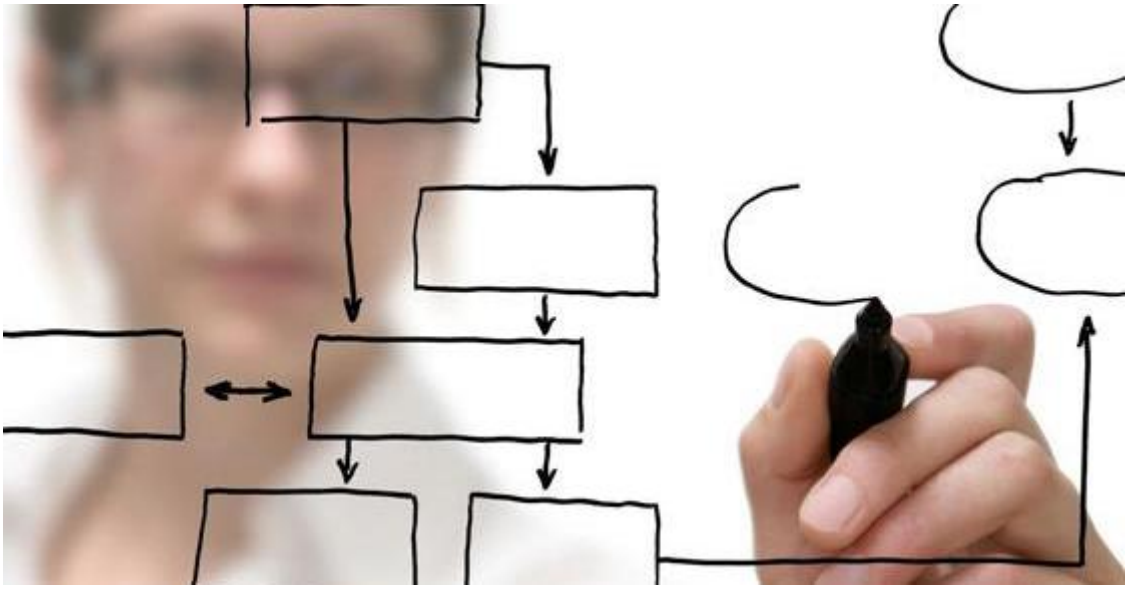
### 4.1.2. ინტერაქტივის დაგეგმარება

<u>სწავლის შედეგის შესაბამისი თემატიკა</u>

- ინტერაქტივის დაგეგმარების განხილვა
- ინტერაქტივის სცენარის შედგენა
- ინტერაქციის დიზაინის პრინციპების და სასარგებლო რჩევების განხილვა

პროქტის ფარგლებში, ჩვენ ვაყალიბებთ მომხმარებლების პროფილებს და სხვადასხვა ტიპის სცენარებს. ჩვენ ასევე შეგვიძლია ჩამოვაყალიბოთ და დავგეგმოთ კონკრეტული მომხმარებლის შეხება კონკრეტული ინტერფეისის ნაწილთან, კონკრეტული შედეგის მისაღწევად. ამისთვის ჩვენ უნდა გვქონდეს შემუშავებული პროექტის ბრიფი, მომხმარებლის პროფილები. ასევე, უნდა ვიცოდეთ კონკრეტული ინტერაქციის სპეციფიკა.

ამ თავში ჩვენ სამაგალითოდ განვიხილავთ პროექტს, რომელიც დაგვეხმარება ინტერაქციის დაგეგმარებაში.



### პროექტის ბრიფის მოკლე ვერსია

დამკვეთის ვებ საიტის ერთ-ერთი დანიშნულება უნდა იყოს ვაკანსიების გამოცხადების შესაძლებლობა. დამკვეთის მოთხოვნით, მომხმარებელს უნდა ჰქონდეს საშუალება, მისთვის სასურველი ვაკანსიის საფუძველზე, პირდაპირ ვებ საიტზე შეავსოს განაცხადის ფორმა, რომელიც ავტომატურად გაიგზავნება დამკვეთის კადრების მართვის განყოფილებაში.

### მომხმარებლის მოკლე პროფილი

მომხმარებელი არის ახლაგაზრდა პროფესიონალი, რომელიც ახალი სამსახურის ძიების პროცესში არის და ესტუმრება დამკვეთის ვებ საიტს. მანდ ის პოულობს მისთვის საინტერესო ვაკანსიას და წყვეტს თავის კანდიდატურის შეთავაზებას.

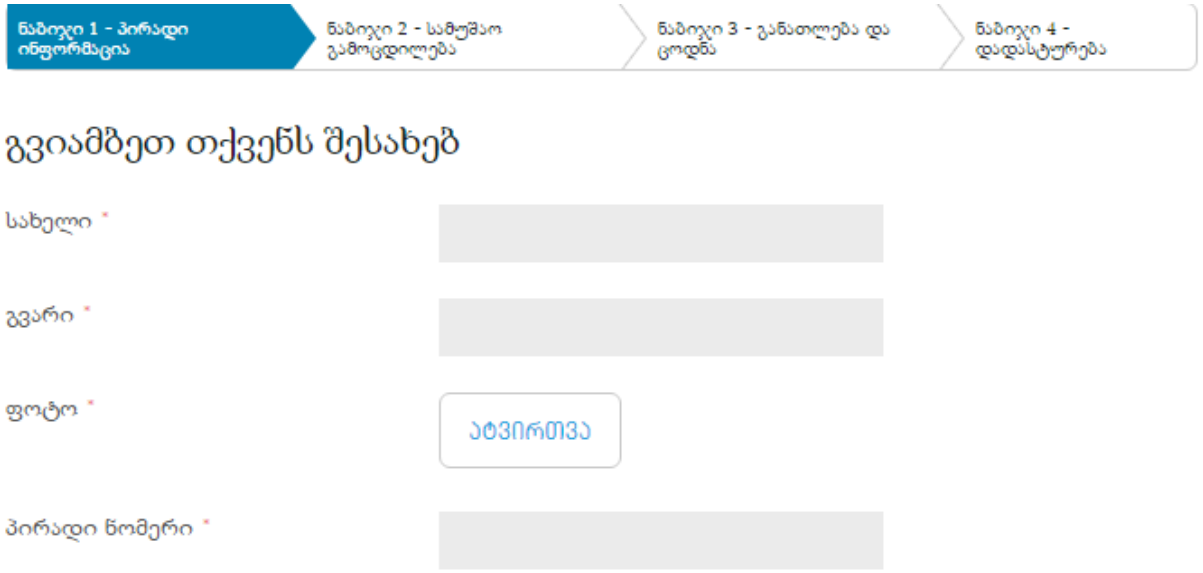
ტრადიციულად, ამ შემთხვევაში მომხმარებელი გააგზავნიდა თავის CV-ს, განცხადებაში მითითებულ ელ. ფოსტის მისამართზე და დაელოდებოდა შეხმიანებას. ჩვენი სამაგალითო პროექტის სცენარის მიხედვით კი, მომხმარებელი განცხადებაში ელ. ფოსტის მისამართის ნაცვლად ხედავს ღილაკს “განაცხადის შევსება”, რომელიც მომხმარებლისთვის არის მოულოდნელი, თუმცა ის მაინც აჭერს ღილაკს და გადადის ინტერაქციის პირველ საფეხურზე.

### ინტერაქციის სცენარი

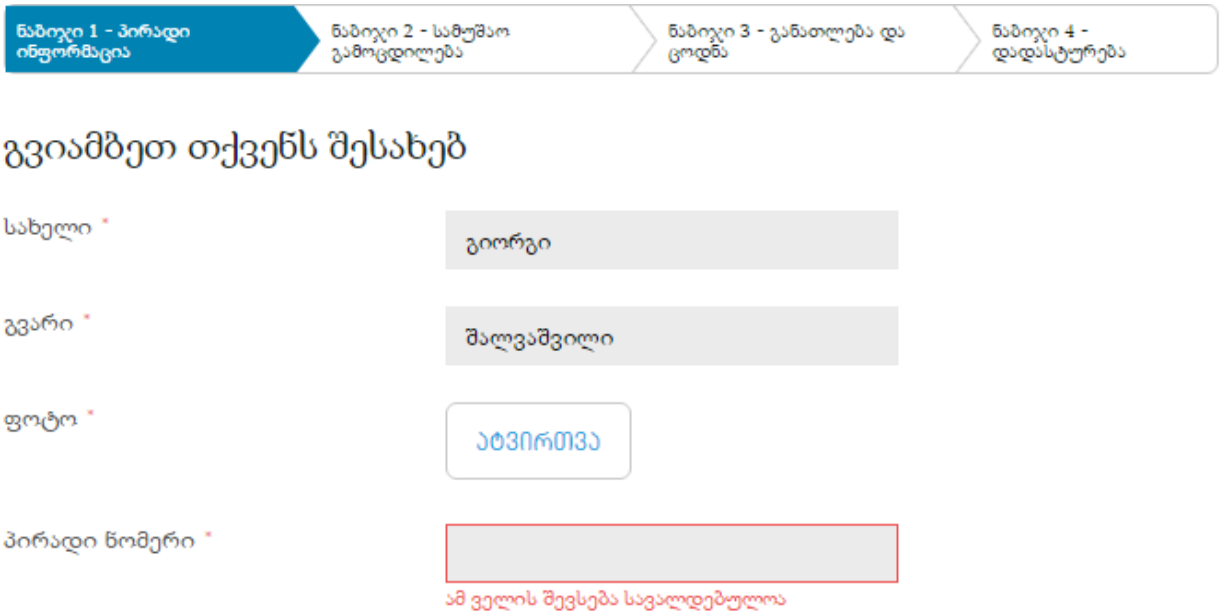
ის თუ რას დაინახავს მომხმარებელი ინტერაქციის შემდეგს საფეხურზე არის ჩვენი გადასაწყვეტი. ჩვენი მიზანი უნდა იყოს; მომხმარებელს დავახვედროთ განაცხადის შევსების ფორმა, რომელიც იქნება მისთვის მარტივი და გასაგები. თუ მომხმარებელი ამ ფორმას ვერ შეავსებს, დამკვეთი არ მიიღებს მის განაცხადს და გარდა რეპუტაციის საკითხისა, კომპანიამ შესაძლოა ვერ იშოვოს სასურველი, კვალიფიციური თანამშრომელი.

მოცემული სცენარიდან გამომდინარე, პირველ რიგში, ჩვენ გვაინტერესებს განაცხადის ფორმის ველების ჩამონათვალი. ამ ველების სია დამკვეთის კომპანიის კადრების მართვის განყოფილებას შესაძლოა ჰქონდეს, ვინაიდან სავარაუდოდ აქამდე იგივე ფორმას პოტენციური კანდიდატი ქაღალდზე ავსებდა.

ფორმის ერთიანობა ქაღალდის შემთხვევაში არ იყო პრობლემა და დიდი ალბათობით გვერდებად იქნებოდა დაყოფილი. ვებ ინტერფეისის შემთხვევაში, უნდა გავითვალისწინოთ ფორმის მთლიანობა და მისი შევსების პროცესის კომფორტი, ამიტომ, მეორე ეტაპზე უნდა გადავწყვიტოთ თუ როგორ დავყოთ ფორმის ველები თემატურ ჯგუფებად.




შედეგად მივიღეთ შემდეგი ინფორმაცია: სააპლიკაციო ფორმა გვაქვს დაყოფილი ოთხ თემატურ საფეხურად. როცა მომხმარებელი დაადასტურებს პირველი საფეხურის ველების შევსებას, მანდ უნდა დააჭიროს ღილაკს “შემდეგ” და გადავიდეს შემდეგ საფეხურზე. ამ ეტაპზე, ჩვენ ვამოწმებთ პირველი საფეხურის ველების ვალიდურობას და შეცდომის შემთხვევაში არ ვუშვებთ მომხმარებელს შემდეგ საფეხურზე, მანამ ის არ გამოასწორებს შეცდომებს.




ფორმასთან ინტერაქციის პროცესში, მომხმარებელმა არ შეავსო პირადი ნომერი საჭირო ველში და ამ შემთხვევაში სისტემა ატყობინებს კონკრეტულად რომელ ველში არის დაშვებული შეცდომა. როგორც წინა თავში ვახსენეთ, ყოველთვის სასურველია შეცდომის პრევენცია. ამ ფორმის შემთხვევაში, ჩვენ გვაქვს

მითითებული ველის სათაურთან წითელი ფიფქი, რაც იმის ნიშანია რომ ველი აუცილებლად უნდა შეივსოს.

ასევე, შეცდომის პრევენციის მიზნად ჩვენ შეგვეძლო დაგვემატებინა ორი დამატებითი დახმარების ნიშანი ა) ფოტოს ატვირთვის ღილაკთან მივუთითოთ თუ რა ტიპის და ფორმატის ფოტოს ველით მომხმარებლისგან და ბ) პირადი ნომრის ველი გვერდით დავსვათ კითხვის ნიშნის ღილაკი, რომელიც მომხმარებელს დამატებით დაუზუსტებს რომ პირადი ნომერი უნდა იყოს 11 ნიშნა ციფრი.

ინტერაქტივის სცენარის შედგენისას, ჩვენ შეგვიძლია მივუთითოთ ყველა ფორმის ველის სპეციფიური დეტალი და ასევე გავითვალისწინოთ მომხმარებლის მიერ დაშვებული შეცდომები. შედეგად, სცენარში გვექნება ყველა შესაძლო შემთხვევა გაწერილი და რომლის მიხედვითაც მოხდება ფორმის აგება ელექტრონულად.

**ფორმის შემუშავების ბრიფი**

ელექტრონული ფორმა ეხმარება მომხმარებელს ვაკანსიის განაცხადის შევსებაში. ფორმის ველები დაყოფილია საფეხურებად. შემდეგ საფეხურზე გადასვლა შესაძლებელია მხოლოდ წინა საფეხურის ყველა ველის კორექტულად შევსების შედეგად. ფორმის შევსებული მონაცემები ფინალური დასტურის შედეგად უნდა იგზავნებოდეს xx@xxx.com მისამართზე.

**ფორმის ველები**

ნაბიჯი პირველი - პირადი ინფორმაცია

- სახელი
  ტექსტური ველი. შესაძლებელია მხოლოდ ასოების შეყვანა.

- გვარი
  ტექსტური ველი. შესაძლებელია მხოლოდ ასოების შეყვანა.

- ფოტო
  ფაილის ატვირთვის საშუალება. დაშვებულია jpg და png ფაილები.

- პირადი ნომერი
  ტექსტური ველი. შესაძლებელია მხოლოდ ციფრების შეყვანა (11).

- სქესი
  ასარჩევი ველი. პარამეტრები: მამრობითი და მდედრობითი

- დაბადების თარიღი
  სამი, დღის, თვის და წლის ასარჩევი ველი. დღის პარამეტრები: 1-31, თვის პარამეტრები: კალენდარული 12 თვე, წლის პარამეტრები: 1960-1999 პერიოდი.

- ელ. ფოსტის მისამართი
  ტექსტური ველი. ველი უნდა ამოწმებდეს ელ. ფოსტის ფორმატირების სიზუსტეს

- ტელეფონის ნომერი
  ტექსტური ველი. შესაძლებელია მხოლოდ ციფრების შეყვანა. ავტომატურად შეყვანილია +995 და ცხრა ნიშნა ნომრის შეყვანის საშუალება აქვს.

ბრიფში უნდა იყოს მითითებული ფორმის ყველა ველი და მათი ყველა დეტალი. ბრიფის ჩამოყალიბებაში შეგიძლიათ ასევე დახაზოთ ფორმის სტრუქტურა (wireframe) და პირდაპირ ქაღალდზე მიუთითოთ ველების მოთხოვნები და შეზღუდვები.

ფორმის შევსება არის მხოლოდ ერთი ინტერაქციის მაგალითი. მომხმარებელსა და ინტერფეისს შორის არსებობს სხვა ინტერაქციაც, რომელიც ეხება ინტერფეისის სხვადასხვა ელემენტებს: სურათებს, სანავიგაციო მენიუს, ტექსტს და ა.შ. ინტერაქციის დაგეგმარება არის პროცესი, რომელიც დაგეხმარებათ არასტანდარტული ინტერაქციის შემუშავებაში და ინტერაქციის ყველა მნიშვნელოვანი საკითხის მკაფიოდ ჩამოყალიბებაში. ყოველთვის ჯობია წინასწარ დაგეგმოთ, ვიდრე სამუშაო პროცესში აღმოაჩინოთ რომ ინტერფეისის რომელიმე ელემენტის ინტერაქციის სცენარი არ გაქვთ გათვალისწინებული.

გაითვალისწინეთ შემდეგი სასარგებლო რჩევები, ინტერაქციის დაგეგმარების პროცესში:

- განსაზღვრეთ მომხმარებლის და ინტერფეისს შორის ინტერაქციის ტიპი და დასვით შემდეგი კითხვები:
  - რის გაკეთება შეუძლია მომხმარებელს მოცემული ინტერფეისის ფარგლებში?
  - ინტერფეისის რომელი ნაწილი იქნება გამოყენებული მაუსით ან თითით?
  - არის თუ არა ინტერფეისში ელემენტები, რომელიც საჭიროებს ინსტრუქციას?

- მიაწოდეთ მომხმარებელს ინფორმაცია ინტერფეისის კონკრეტული დეტალის შესახებ, მანამ მომხმარებელი თვითონ მიხვდება მას.
  - ფორმის ველების სახელები უნდა იყოს გასაგები.

- თუ ფორმის რომელიმე ველი სავალდებულოა, ანიშნეთ მომხმარებელს წინასწარ.
- ტექსტის მონაკვეთში სიტყვა, რომელიც ბმულია, უნდა გამოიყურებოდეს როგორც ბმული.

- გაითვალისწინეთ და მართეთ შეცდომები
  - მაქსიმალურად აირიდეთ სისტემური ან/და მომხმარებლის მიერ დაშვებულ შეცდომები. პირადად გადაამოწმეთ ყველა ინტერაქცია და დარწმუნდით რომ ის გამართულად მუშაობს.
  - განსაზღვრეთ, რა ინფორმაციის მიწოდება შეგიძლიათ მომხმარებლისთვის იმისათვის რომ მან არ დაუშვას ინტერაქციაში შეცდომა
  - შეცდომის შემთხვევაში მიაწოდეთ ინფორმაცია მომხმარებელს, იმის შესახებ თუ რა მოხდა და რის გაკეთება შეუძლია მას შეცდომის გამოსასწორებლად.

- გაითვალისწინეთ სისტემური უკუკავშირი და შედეგის დრო
  - ფორმის შევსების და დასტურის შედეგად გამოაჩინეთ შესაბამისი შეტყობინება.
  - ვებ საიტის რომელიმე შიდა გვერდზე გადასვლის შემდეგ, ანიშნეთ მომხმარებელს სანავიგაციო მენიუში თუ რომელი გვერდი არის გააქტიურებული.
  - ნებისმიერი მოქმედება, რომლის შედეგად მომხმარებელი არ არის ინფორმირებული 10 წამის ფარგლებში, მომხმარებლის მიერ აღიქმება შეცდომად

- განსაზღვრეთ ინტერფეისის ყველა ცალკეული ელემენტები და მათი ჯგუფები და დასვით შემდეგი კითხვები:
  - არის თუ არა ინტერფეისის ელემენტების ზომები მისაღები მომხმარებლების სამიზნე ჯგუფისთვის?
  - არის თუ არა ელემენტების დაჯგუფება ლოგიკურად გადაწყვეტილი?
  - გათვალისწინებულია თუ არა ინტერფეისში დამკვიდრებული ნორმები?

- გაუმარტივეთ მომხმარებელს ინტერაქციის პროცესი
  - სანავიგაციო მენიუში პუნქტების რაოდენობა არ უნდა აღემატებოდეს 5-7 პუნქტს, ვინაიდან მომხმარებელი ვერ იმახსოვრებს მეტს ინტერფეისთან ინტერაქციისას.
  - დანერგეთ ისეთი გადაწყვეტილებები, რომელიც გავრცელებულია ინდუსტრიაში და გამოყენებულია პოპულარულ რესურსებზე. მომხმარებელს აქვს გამოცდილება იმ რესურსების გამოყენების და დამახსოვრებული აქვს იმ გადაწყვეტილების კონვენციები.
  - ყოველთვის ეცადეთ დანერგოთ ყველაზე მარტივი გადაწყვეტილება. მომხმარებელს არ უნდა სჭირდებოდეს ინსტრუქტორი ინტერფეისთან ინტერაქციის პროცესში.

**სავარჯიშო**

დაასრულეთ განაცხადის ფორმა და შეადგინეთ ფორმის თქვენი ვერსია ქაღალდზე. მიუთითეთ ყველა ველის სპეციფიკა და განიხილეთ ჯგუფის წევრებთან.

## 4.2. ინტერაქტივი და ეფექტები

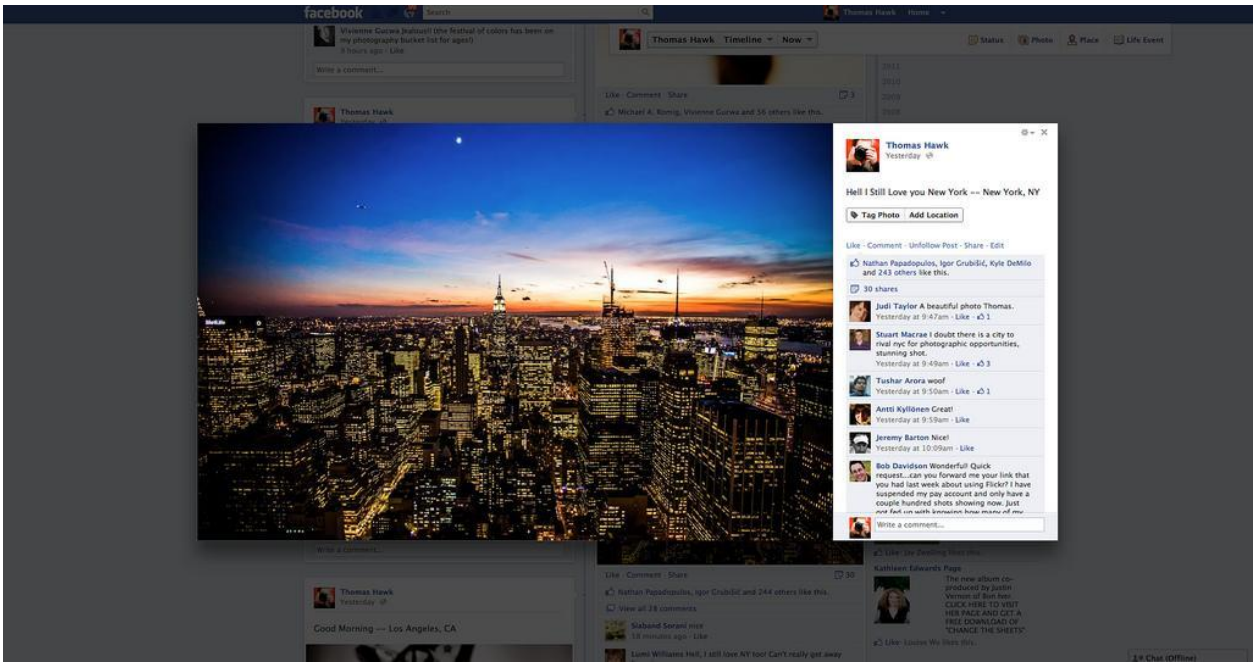
### 4.2.1. მიკრო ინტერაქცია

სწავლის შედეგის შესაბამისი თემატიკა

- მიკრო ინტერაქციის გაცნობა
- მიკრო ინტერაქციის ნაწილების განხილვა

გარდა ინტერფეისის სტანდარტული ნაწილებისა, როგორიც არის ფორმები და სანავიგაციო მენიუ, მომხმარებელს ზოგჯერ შეხება აქვს ინტერფეისის ისეთ ელემენტებთან, რომელიც ერთის მხრივ შესაძლოა მოგვეჩვენოს წვრილმანი ან/და უმნიშვნელო.

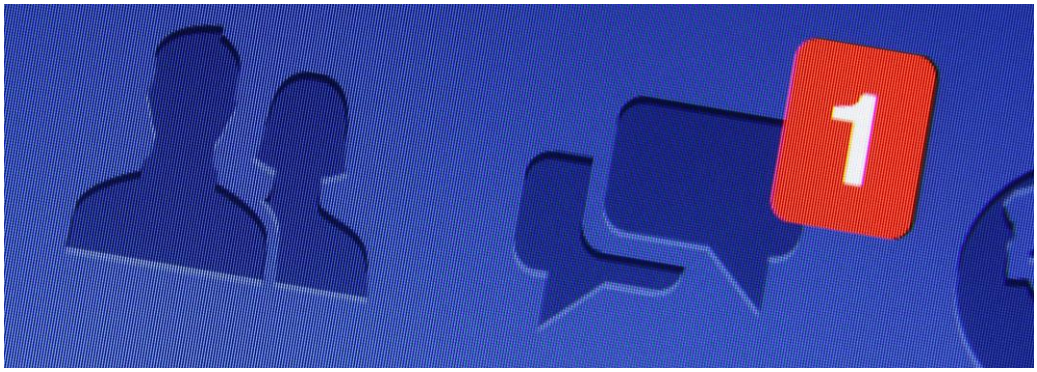
მიკრო ინტერაქცია არის მომხმარებლის შეხება ინტერფეისის პატარა ელემენტებთან, როგორიც არის მაგალითად Facebook-ის Like ღილაკი. დღის განმავლობაში მომხმარებელს ძალიან ბევრ მიკრო ინტერაქციასთან უწევს შეხება, რომელსაც ის ხშირად არ აღიქვამს როგორც ინტერაქციას. სინამდვილეში, მიკრო ინტერაქციების ნაკრები, ერთი ინტერფეისის ფარგლებში ჰქმნის მომხმარებლის გონებაში წარმოდგენას კონკრეტული ვებ საიტის ინტერფეისის შესახებ.




მაგალითად, Facebook-ის გამოყენებისას, მომხმარებელი არ ფიქრდება იმაზე თუ როგორ ათვალიერებს თავისი მეგობრის ფოტოებს და აჭერს Like ღილაკს. დღის განმავლობაში, მხოლოდ Facebook-ის ინტერფეისის ფარგლებში მომხმარებელს აქვს შეხება ასობით სხვადასხვა მიკრო ინტერაქციასთან. Facebook-ის დიზაინერებმა დაგეგმეს ეს მიკრო ინტერაქციების, რომ საერთო ინტერფეისი იყოს მომხმარებლისთვის მაქსიმალურად ინტუიტიური და კომფორტული. ზუსტად ეს არის ერთ-ერთი მიზეზი თუ რატომ არის Facebook-ის პლატფორმა ასეთი პოპულარული.

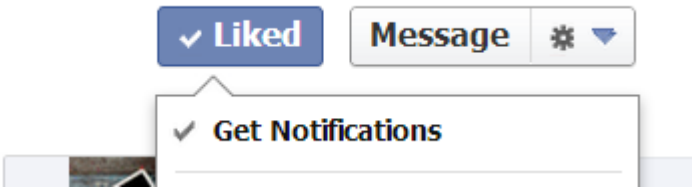
ყველა მიკრო ინტერაქცია ძირითადად ოთხი ნაწილისგან შედგება. მხოლოდ იმ შემთხვევაში როცა ოთხივე ნაწილი სრულდება მიკრო ინტერაქცია აღიქმება წარმატებულად:

1. **ინიცირება**
   მიკრო ინტერაქციის შესრულების პირველი ნაწილი. ინიცირება შესაძლოა მოხდეს მომხმარებლის ხშირად, მაგალითად ღილაკის დაჭერისას. ასევე, სისტემას შეუძლია მიკრო ინტერაქციის ინიცირება, მაგალითად ხმოვანი და ვიზუალური შეტყობინებით, რომელიც გატყობინებთ Facebook-ში ახალი შეტყობინების მიღების შესახებ.




2. **წესები**
   იმისათვის რომ მიკრო ინტერაქცია  შესრულდეს, უნდა არსებობდეს კონკრეტული წესები. ინტერაქციის წესი არის იგივე სცენარი, რომელიც დაგეგმილია წინასწარ და აყალიბებს თუ რა უნდა მოხდეს მომხმარებლის დინების პროცესში, კონკრეტული მიკრო ინტერაქციის შემთხვევაში. მაგალითად, ფონური პროცესი, თუ ხდება სისტემაში მაშინ როცა თქვენ მეგობრის ფოტოზე Facebook-ში Like-ის ღილაკს აჭერთ.

3. **უკუკავშირი**
   ვინაიდან თქვენ მიერ დაგეგმილი წესები არის ფარული, აუცილებელია უკუკავშირი, რათა მომხმარებელი ჩართული იყოს ინტერაქციის პროცესში და დარწმუნდეს რომ მისი მიკრო ინტერაქცია შესრულდა. მაგალითად, სურათზე Like-ის დაჭერის შემთხვევაში, თქვენს ინტერფეისში, Like-ის ღილაკი უქმდება, თვენი სახელი ავტომატურად ჩნდება იმ სურათის Like-ის ღილაკის ტერიტორიაში, თქვენს მეგობარს მისდის ამის შესახებ შეტყობინება, და ა.შ.




4. **რეჟიმი და განმეორება**
   ეს არის მიკრო ინტერაქციის ბოლო ნაწილი. რეჟიმი არის მიკრო ინტერაქციის რეჟიმი, რომელიც მომხმარებლისგან ითხოვს დამატებით მოქმედებას. მაგალითად შეტყობინების წაშლის შემთხვევაში დასტურს. განმეორება კი გამოიყენება მიკრო ინტერაქციის ციკლის შემთხვევაში, როცა საჭიროა კონკრეტული ინტერაქციის განმეორება რამდენიმე ჯერ სასურველი შედეგის მისაღწევად. მაგალითად, Like-ის ღილაკის დაჭერა არის ერთჯერადი მოქმედება და

შეტყობინებების გათიშვა დილის 9:00 საათამდე არის განმეორებითი მოქმედება, რომელიც სრულდება დილის 9:00 საათამდე.

**თვითშეფასება**

- ითვლება თუ არა სისტემაში შესვლის / ავტორიზაციის ინტერაქცია მიკრო ინტერაქციად?
- რა უკუკავშირს იძლევა GMail-ის სისტემა წერილი გაგზავნის ღილაკის დაჭერის შემდეგ?

### 4.2.2. ანიმაცია და სხვა ეფექტები

სწავლის შედეგის შესაბამისი თემატიკა

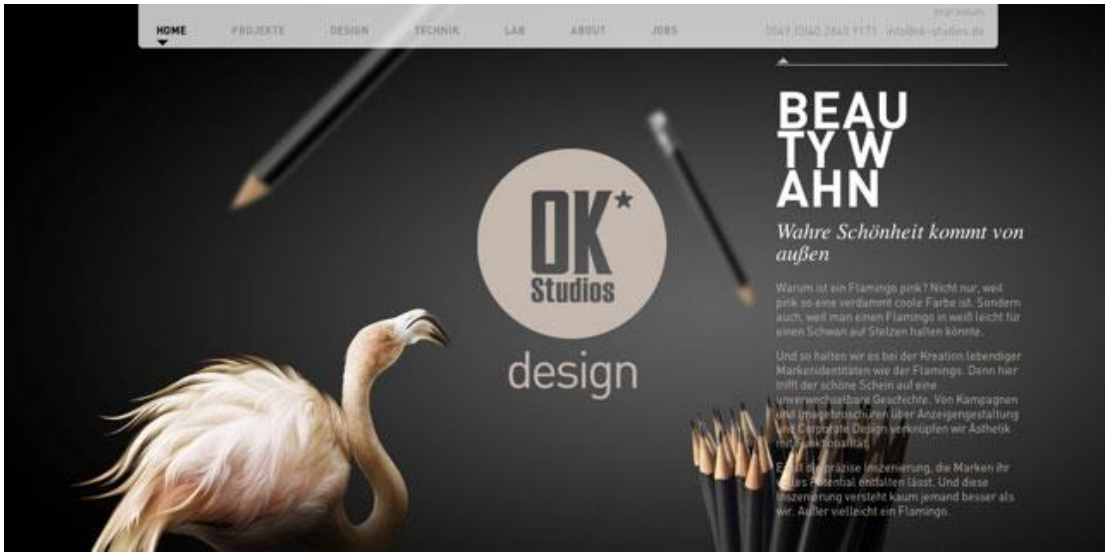
- ანიმაციის და სხვა ეფექტების გაცნობა
- ანიმაციის და სხვა ეფექტების ნაწილების განხილვა

ინტერფეისთან ინტერაქციაში ანიმაცია და სხვა ტიპის ეფექტები აუმჯობესებს ინტერფეისის უკუკავშირს მომხმარებელთან. ღილაკთან მაუსის კურსორის მიტანისას შეგვიძლია შევცვალოთ ღილაკის ფერი. ბმულის შემთხვევაში შეგვიძლია გავაქროთ ქვედა ხაზი.

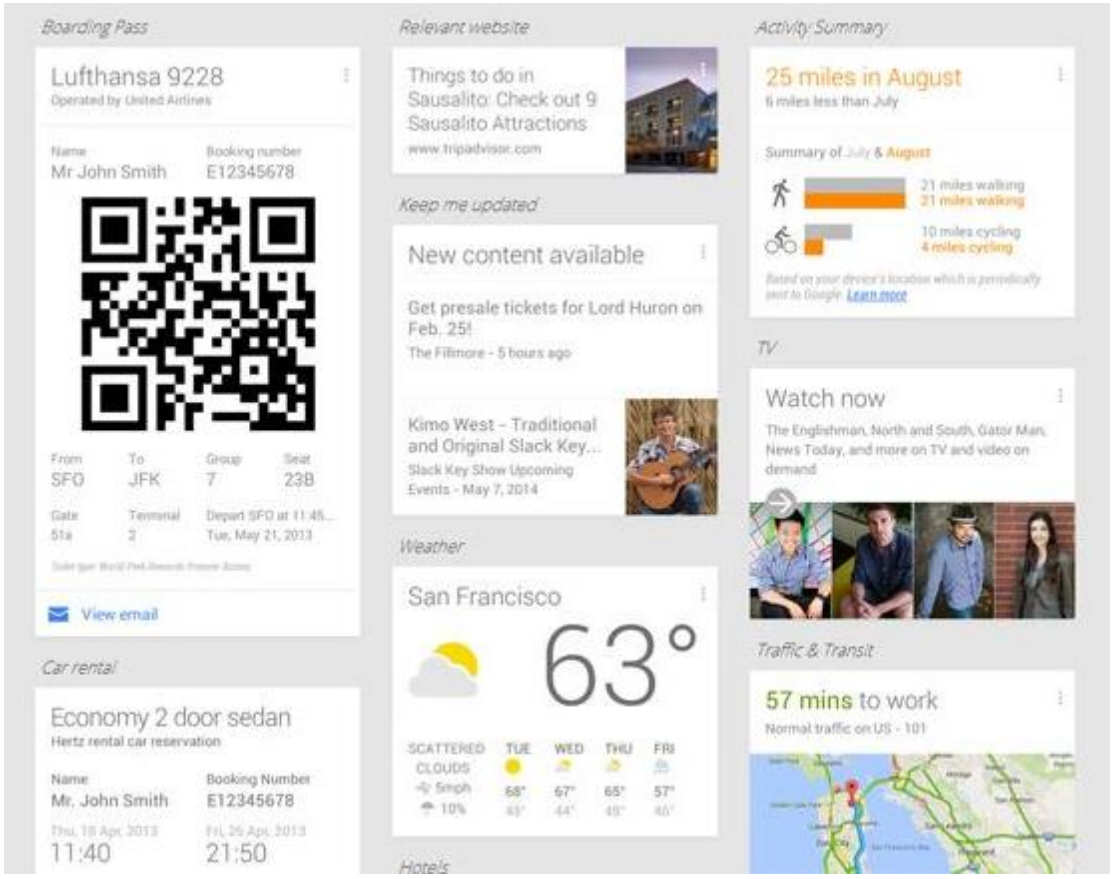
თანამედროვე HTML-ი და CSS-ი გვაძლევს დამატებით შესაძლებლობებს ინტერაქტივში დავამატოთ ანიმაცია და სპეც. ეფექტები, რომელიც ინტერფეისის გამოყენებას უფრო სასიამოვნოს ხდის.

მაგალითისთვის, განვიხილოთ რამდენიმე ეფექტი, რომელიც აუმჯობესებს ინტერფეისთან ინტერაქციას:

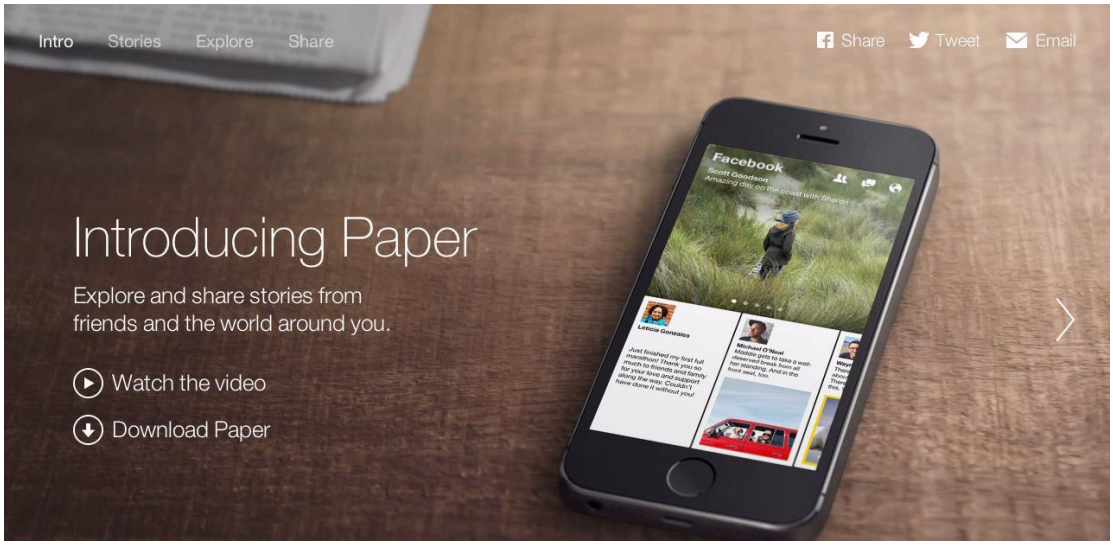
- პარალაქსი
  ვებ საიტის ინტერფეისის ელემენტები დალაგებულია სხვადასხვა ფენებზე, მაუსის სკროლით მომხმარებელი ახდენს ნავიგაციას ვებ საიტის გვერდის ფარგლებში და ხედავს ელემენტების ცალკეულ მოძრაობას. ეს ჰქმნის სივრცულ ეფექტს, რომელიც თუ კორექტულად არის შემუშავებული, შესაძლოა იყოს ძალიან ეფექტური.




- [ბარათები]
  ამ სტილის და მისი ეფექტების პოპულარიზაცია Google-ს ეკუთვნის. Google +-ის მომხმარებლებისთვის, ბარათების სტილის ინტერფეისი ნაცნობი იქნება. ბარათების გამოყენება ინტერფეისში ასევე გამოიყენება ვებ ინტერფეისებში. ბარათები იძლება ინფორმაციის ეფექტური განლაგების შესაძლებლობას, ძალიან ეფექტურია ინტერაქციაში და საჭიროების შემთხვევაში იძლევა 3D ანიმაციის შესაძლებლობას, რომელიც მომხმარებელს ბარათის უკანა მხარეს აჩვენებს.




- [ფონური ვიდეო]
  ვებ საიტის ფონად განთავსებული ვიდეო, ბევრად უფრო ეფექტურად აწვდის მომხმარებელს საჭირო ინფორმაციას, ვიდრე ტექსტი ან/და სურათები.




ვებ საიტის ინტერფეისის აგებისას, ანიმაციის და სპეციალური ეფექტების გამოყენება მოითხოვს ვიზუალურ ესთეტიკას. ზედმეტი ეფექტების გამოყენების შემთხვევაში, ეფექტურობის ნაცვლად შესაძლებელია შეიძინოთ უკმაყოფილო მომხმარებელი.

lingscars.com არის ვებ საიტი, რომელიც გახდა ამ ფენომენის მსხვერპლი და პარადოქსულად გახდა პოპულარული თავისი გადატვირთული ინტერფეისის გამო. ვებ საიტის გამოყენება პრაქტიკულად შეუძლებელია, ვინაიდან ის სავსეა სპეც. ეფექტებით, რომელიც პრაქტიკულად აიძულებს მომხმარებელს, რაც შეიძლება სწრაფად დახუროს ვებ საიტი.


## სავარჯიშო

განიხილეთ ვებ საიტის ინტერაქტივის სხვა შესაძლებლობები ჯგუფურად და გაუზიარეთ ჯგუფის წევრებს სხვა მაგალითები.

### 4.2.3. ხელსაწყოები

სწავლის შედეგის შესაბამისი თემატიკა

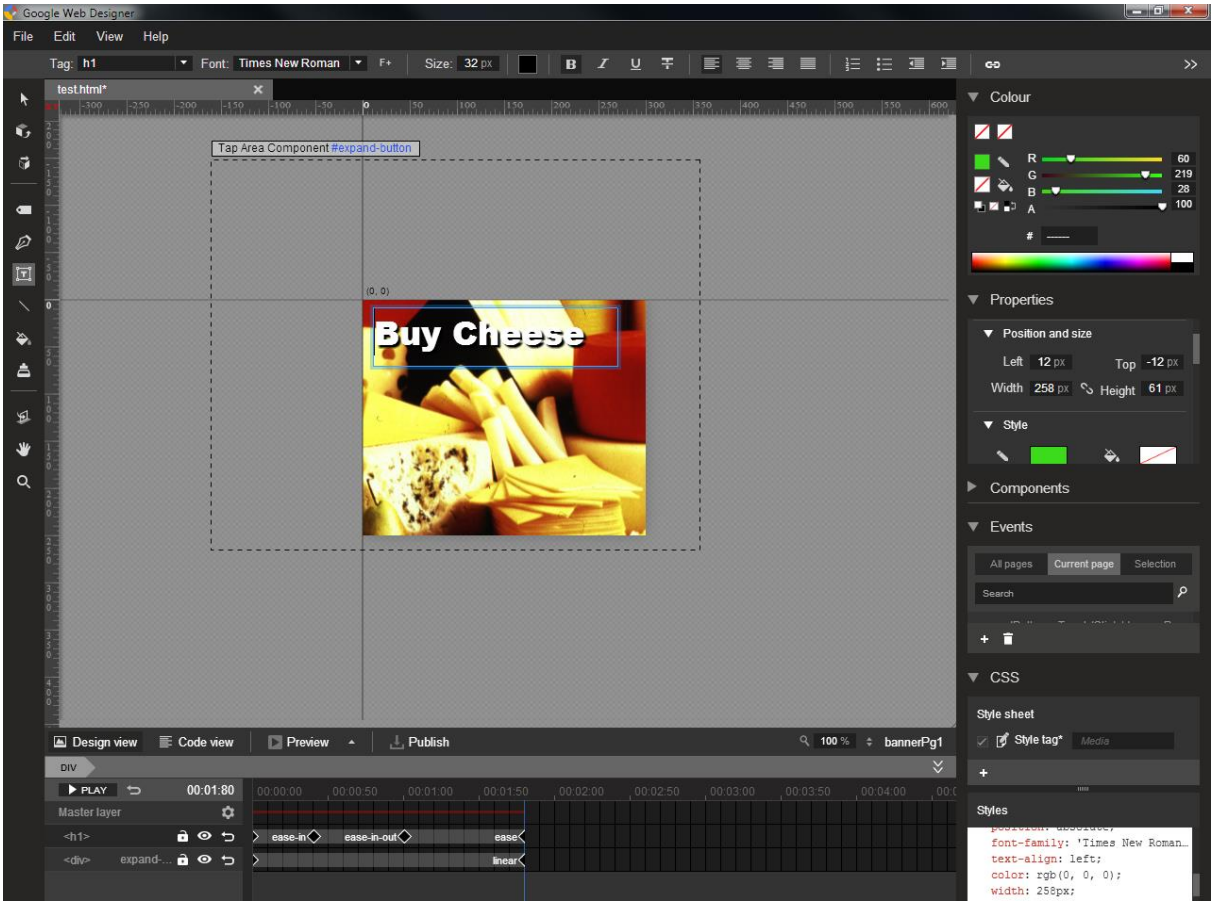
- ვებ ანიმაციის ახალი ხელსაწყოების განხილვა
- რესპონსიული ვებ ინტერფეისების ხელსაწყოების განხილვა

ვებ საიტის ინტერფეისის აგებაში დაგჭირდებათ რამდენიმე სხვადასხვა ტიპის პროგრამული უზრუნველყოფის გამოყენება. ყველა პროგრამას თავისი დანიშნულება აქვს, რომელიც დაგეხმარებათ კონკრეტული ამოცანის შესრულებაში.

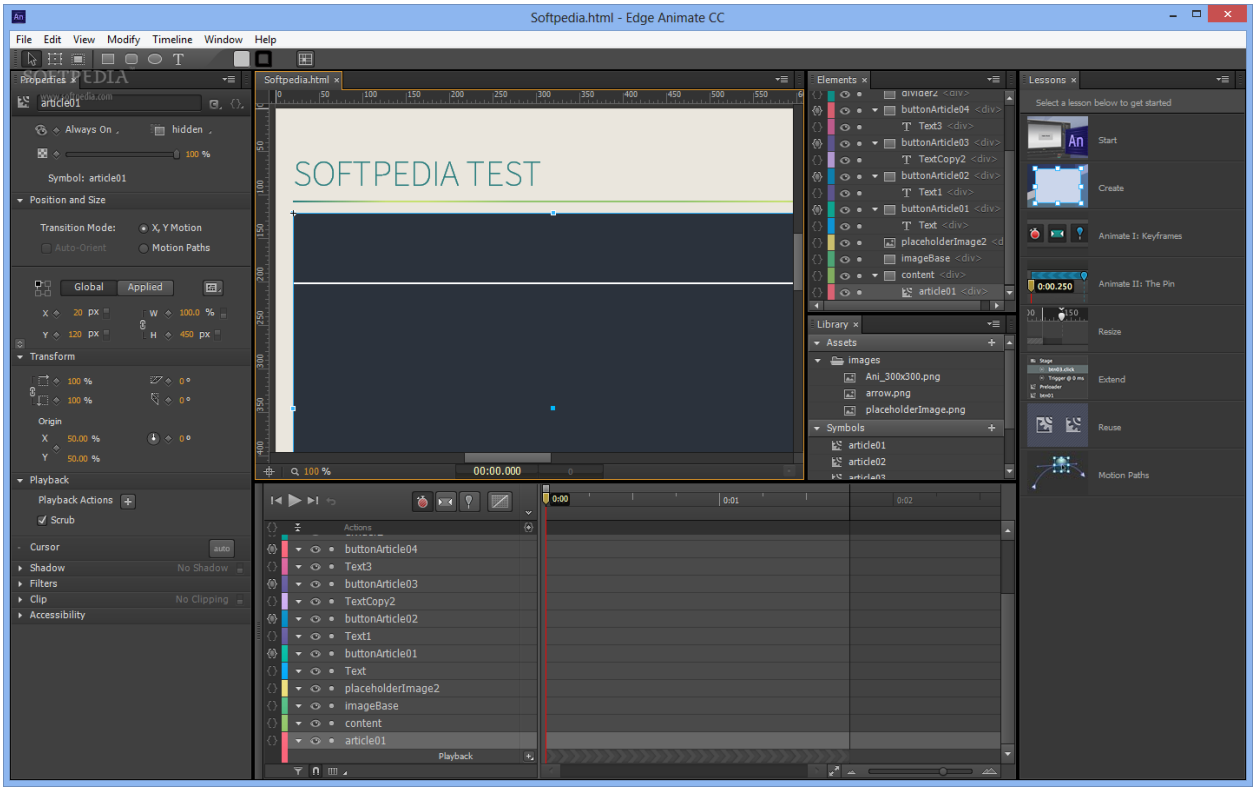
ჩვენ ცალკე განვიხილეთ სტრუქტურების (wireframes) და პროტოტიპების აგების პროგრამები. ასევე, სრული მოდულები გვაქვს დათმობილი რასტრული და ვექტორული ობიექტების შექმნისთვის. ამ თავში კი განვიხილავთ დამატებით პროგრამებს, რომლებიც დაგვეხმარება ინტერაქტიული ვებ ინტერფეისის აგებაში.

პირველ რიგში, გვაქვს ვებ ანიმაციის პროგრამები. ამ პროგრამების მეშვეობით თქვენ შეძლებთ, ვიზუალურად ააგოთ სასურველი ანიმაცია, რომელიც დამატებითი პროგრამირების გარეშე იმუშავებს სხვადასხვა ბრაუზერებში.

Google Web Designer ეს პროგრამა არის უფასო. წინასწარ მომზადებული ინტერფეისის ელემენტების შემუშავების შემდეგ, მისი დახმარებით თქვენ შეძლებთ ძალიან მარტივად ააგოთ სრულფასოვანი HTML გვერდი. პროგრამის ყველაზე დიდი უპირატესობა არის HTML/CSS ანიმაციის შემუშავება, რომელიც შესაძლებელია მარტივი ფუნქციების გამოყენებით.



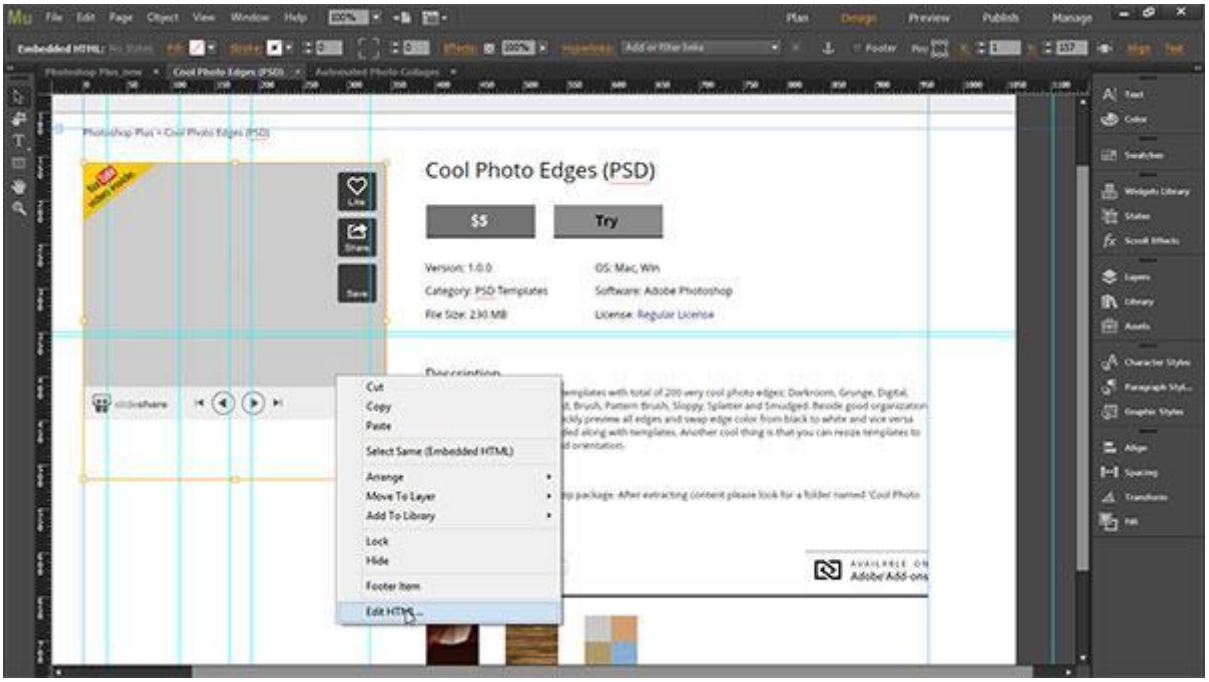

Adobe Edge Animate წინა პროგრამასთან შედარებით, ეს არის ფასიანი პროდუქტი. ამ პროგრამას ზუსტად იგივე შესაძლებლობები აქვს რაც წინას, თუმცა ვინაიდან ეს არის Adobe-ის პროდუქტი, მას სხვა Adobe-ის პროდუქტებთან ინტეგრაციის შესაძლებლობა აქვს, რაც ძალიან კომფორტულია.



ანიმაციის კატეგორიაში კიდევ არსებობს Tumult Hype და Sencha Animator. ძირითადად ყველა პროგრამას ერთიდაიგივე შესაძლებლობები აქვს, თავისი გარკვეული უპირატესობებით. ყველა პროგრამას აქვს საცდელი პერიოდი, რომელიც შეგიძლიათ გამოიყენოთ სასურველი სამუშაო გარემოს შერჩევისთვის.

იმისათვის რომ მოვახდინოთ ჩვენს მიერ დაგეგმილი ვებ ინტერფეისის ინტერაქცია, ჩვენ გვჭირდება ხელსაწყო, რომელიც მოგვცემს ინტერფეისის აგების მარტივ შესაძლებლობას. ინტერფეისის დეველოპერის დახმარებით ჩვენ შეგვიძლია სრულფასოვანი ინტერფეისის აგება, მაგრამ ბევრად უფრო ეფექტური იქნება თუ ჩვენ შევძლებთ ინტერაქტიული პროტოტიპის აგებას და მისი მეშვეობით დავიწყებთ ჩვენი ინტერაქციის გადამოწმებას.

აღნიშნულისთვის დაგვეხმარება Adobe Muse ამ პროგრამის გამოყენებით ჩვენ შეგვიძლია ავაგოთ რესპონსიული ვებ საიტის ინტერფეისი, გამოვიყენოთ ჩვენს მიერ შემუშავებული ინტერფეისის ყველა ელემენტი და ძალიან მარტივად ვნახოთ შედეგი პირდაპირ ბრაუზერში. ამისთვის არ არის საჭირო ინტერფეისის დეველოპერის ან/და ვებ პროგრამისტის პროცესში ჩარევა.




**სავარჯიშო**

- გამოიყენეთ Adobe Muse თქვენს მიერ ქაღალდზე შემუშავებული ინტერაქციის აგებისთვის და გადაამოწმეთ მიღებული შედეგი პოპულარულ ბრაუზერებში.
- გამოიყენეთ ანიმაციის რომელიმე პროგრამა და ააგეთ თქვენი პირველი ვებ ანიმაცია.

## 4.3. ხარისხის უზრუნველყოფა

### 4.3.1. ბრაუზერები

სწავლის შედეგის შესაბამისი თემატიკა

- ბრაუზერების დანიშნულების განხილვა
- პოპულარულის ბრაუზერების განხილვა

ბრაუზერი არის კომპიუტერული პროგრამა. ის მომხმარებლებს ვებ საიტებთან ინტერაქციის შესაძლებლობას აძლევს. ბრაუზერების დიდი არჩევანი არსებობს. ყველას თავისი უპირატესობები და დამატებითი ხელსაწყოები გააჩნია. მომხმარებლებს აქვს საშუალება აირჩიოს მისთვის სასარგებლო ბრაუზერი და ისარგებლოს მისი შესაძლებლობებით.


გარდა სხვა შესაძლებლობებისა, ბრაუზერის ძირითადი დანიშნულება არის გამოქვეყნებული ვებ საიტის HTML, CSS და JavaScript-ის კოდის კონვერტაცია, რიგითი მომხმარებლისთვის მისაღებ, ვიზუალურ ინტერფეისად. ბრაუზერი ამას ახდენს თავისი ბირთვის საშუალებით და ბირთვების უნიკალურობის გამო, ყველა ბრაუზერი თავისებურად აღიქვამს ვებ საიტის კოდს.

ყველა ბრაუზერში ვებ საიტის გადამოწმება საკმაოდ შრომატევადი პროცესია. Internet Explorer მოყვება Windows ოპერაციულ სისტემას, ამიტომ ძალიან ბევრი მომხმარებელი მსოფლიოში სარგებლობს ამ ბრაუზერით. Chrome-ი არის ასევე ძალიან პოპულარული ალტერნატიული ბრაუზერი, განსაკუთრებით საქართველოში.

პირველ ეტაპზე, საკმარისია ვებ საიტის გადამოწმება მხოლოდ პოპულარული ბრაუზერების გამოყენებით. ამის გარდა, გაითვალისწინეთ პროექტის პერსონები და სცენარები. თუ პროექტის ფარგლებში, რომელიმე პერსონას სცენარში უწერია რომ სარგებლოს მოძველებული კომპიუტერით ან/და მობილური მოწყობილობებით, რომელსაც აქვს სპეციფიური ბრაუზერი, შესაბამისად თვენი ნამუშევარში აუცილებლად უნდა გადამოწმდეს იმ მოწყობილობაზე ან/და ბრაუზერზე.

**სავარჯიშო**

გადმოწერეთ და დააყენეთ 4 პოპულარული ბრაუზერი თქვენს კომპიუტერზე. შეადარეთ რამდენიმე ვებ საიტის სხვადასხვა ბრაუზერში და იპოვეთ განსხვავებები.

### 4.3.2. მოწყობილობები

<u>სწავლის შედეგის შესაბამისი თემატიკა</u>

- მოწყობილობების სპეციფიკის განხილვა
- პოპულარული მოწყობილობების განხილვა

მომხმარებელი სარგებლობს სხვადასხვა ტიპის მოწყობილობებით. დღესდღეობით ჩვენ უკვე გვაქვს საათი და მაცივარი, რომელიც ინფორმაციას ინტერნეტიდან ღებულობს. მომავალში იქნება კიდევ სხვა, განსხვავებული ტიპის მოწყობილობებიც.

ცხადია, ყველა მოწყობილობა შესაძლოა არ იყოს თქვენთვის ხელმისაწვდომი და ეს ართულებდეს გადამოწმების პროცეფურას, ამიტომ ვებ საიტის შემოწმებისას პრიორიტეტული უნდა იყოს ის მოწყობილობები, რომელიც ყველაზე პოპულარულია მომხმარებლებში.


- **კომპიუტერი (დესკტოპი)**
  მომხმარებელი სარგებლოს ამ ტიპის მოწყობილობით სახლის პირობებში ან/და სამსახურში. ზოგი ძალიან სწრაფია და დიდი ოპერატიული მეხსიერება აქვს, ზოგი მოძველებული მოდელის არის და შედარებით ნელა მუშაობს. ზედმეტი ელემენტებით გადატვირთულმა ვებ საიტმა შესაძლოა მომხმარებელს "გაუჭედოს" კომპიუტერი.

  ასევე, გასათავლისწინებელია პერსონალური კომპიუტერს მონიტორებს და მათი გარჩევადობის ნაირსახეობა. ზოგ მომხმარებელს 17 დუიმიანი მონიტორების ზომა აქვს, რომელსაც 1366 x 768 გარჩევადობის მხარდაჭერა ექნება და სხვას შესაძლოა 24 დუიმიანი მონიტორი ჰქონდეს 1920 x 1080 გარჩევადობით.

- **პორტატული კომპიუტერი (ლეპტოპი)**
  პერსონალური კომპიუტერის შემთხვევაში, იგივე მახასიათებლები არის მნიშვნელოვანი რაც ზევით იყო ნახსენები, დამატებითი შესაძლოა გავითვალისწინოთ რომ პერსონალურ კომპიუტერზე მომხმარებელს ხშირად არ აქვს მაუსი მიერთებული და სარგებლობს touchpad ტიპის მოწყობილობით.

- **პორტაბელური კომპიუტერი (ტაბლეტი და სმარტფონი)**
  ტაბლეტების შემთხვევაში, გაითვალისწინეთ მისი ეკრანის ზომა და გარჩევადობა. ამ მოწყობილობის გამოყენებისას, მომხმარებელი თითებით ახერხებს ნავიგაციას. ამიტომ, სანავიგაციო ელემენტები უნდა იყოს შესაბამისი ზომის, რათა კომფორტული იყოს მათი გამოყენება.

  პორტაბელური მოწყობილობა დამოკიდებული ელემენტის სიმძლავრეზე. გადატვირთული ვებ საიტის ამ/და აპლიკაცია საჭიროებს დამატებით რესურს, რომელიც შესაბამისად იყენებს უფრო მეტ ენერგიას. აქედან გამომდინარე, გადატვირთული ვებ საიტის სარგებლობისას მომხმარებლის მოწყობილობის ელემენტი შედარებით სწრაფად იცლება.

  ასევე, ხშირ შემთხვევაში პორტაბელური მოწყობილობის გამოყენებისას, მომხმარებელი

სარგებლობს 4G, 3G ან LTE ინტერნეტის შეერთებით. ამ მომსახურების ღირებულება კი დამოკიდებულია მომხმარებლის მიერ მიღებული ინფორმაციის მოცულობაზე. ვებ საიტის ვერსია, რომელიც პორტაბელური კომპიუტერებისთვის არის გათვალისწინებული, უნდა იყოს ოპტიმიზირებული და შედარებით "მსუბუქი", რათა მომხმარებელს არ მოუწიოს ზედმეტი (არასაჭირო) ინფორმაციის გადმოწერა.


ყველა პროექტი განსხვავებულია. თქვენ შესაძლოა მოგიწიოთ მუშაობა პროექტზე, რომელიც ითვალისწინებს არასტანდარტულ მოწყობილობაზე ან/და გარემოზე მორგებას.

მაგალითად, თქვენი ამოცანაა ვებ საიტის დამზადება სადისტრიბუციო კომპანიისთვის, რომლის ხშირი მომხმარებელი არის თავად კომპანიის თანამშრომელი, დისტრიბუტორი, რომელსაც ხშირად სჭირდება ვებ საიტზე გამოქვეყნებული პროდუქციის სურათები ან/და სხვა დეტალები. კომპანიის ყველა დისტრიბუტორს სპეციფიური, ინტერნეტთან დაკავშირებული პორტაბელური კომპიუტერი აქვს, რომელსაც ისინი იყენებენ ქალაქში გადაადგილებისას.

ამ შემთხვევაში, უნდა გაითვალისწინოთ ინტერნეტის შეერთების ლიმიტირებული სიჩქარე, კომპიუტერის ეკრანის გარჩევადობა და ჩაშენებული ბრაუზერის შესაძლებლობები. ტესტირების პროცესში, სასურველია ვებ საიტის შემოწმება ზუსტად იგივე მოწყობილობებით და ამავდროულად ვებ საიტით სარგებლობის სიმულაცია მობილური ინტერნეტის შეერთებით, სამუშაო ადგილის გარეთ.

**სავარჯიშო**

შეადარეთ რამდენიმე ვებ საიტი დესკტოპ-კომპიუტერზე და სმარტფონში და შეადგინეთ განსხვავებების სია.

### 4.3.3. სუბიექტურობა

სწავლის შედეგის შესაბამისი თემატიკა

- ხარისხის უზრუნველყოფის სუბიექტურობის განხილვა

თქვენ პირადად მიიღეთ მონაწილეობა ვებ საიტის დამზადებაში და დიდი ხნის განმავლობაში იყავით ჩართული სამუშაო პროცესში. თქვენთვის ძალიან რთული იქნება ობიექტური შეფასებით ხარისხის შემოწმება. ის თქვენი პირადი ნაშრომია, ყველა ელემენტთან დაკავშირებით ავტომატურად გიჩნდებათ ემოციური კავშირი, რის გამოც შესაძლოა ვერ შეამჩნიოთ გაპარული ხარვეზი.

ამის გამო, რეკომენდირებულია ვებ საიტის შემოწმება მოხდეს დამოუკიდებელ სუბიექტთან. ადამიანთან, რომელსაც არ მიუღია ვებ საიტი შემუშავებაში მონაწილეობა და არ არის ინფორმირებული ამოცანის სპეციფიკაში. ეს შესაძლოა იყოს თქვენი მეგობარი ან ოჯახის წევრი.

შერჩეული ტესტირების სუბიექტი პროექტის შერჩეული პერსონის შესაბამისი უნდა იყოს. მაგალითად, იმ შემთხვევაში, თუ ვებ საიტის არის უნივერსიტეტისთვის და მისი მომხმარებლების ერთ-ერთი პერსონა იქნება უნივერსიტეტის სტუდენტი, მაშინ ტესტირების სუბიექტად შეარჩიეთ ადამიანი, რომელიც რეალურად არის სტუდენტი, თუნდაც სხვა უნივერსიტეტიდან.

ხარვეზების დიდი ნაწილის აღმოჩენაში, საკმარისია ხუთ სუბიექტთან გაიაროთ ტესტირების სხვადასხვა სცენარი.

**სავარჯიშო**

შეადგინეთ დავალების სცენარი და განიხილეთ რომელიმე ვებ საიტის მეგობართან ერთად.

### 4.3.4. შემოწმების სცენარი

სწავლის შედეგის შესაბამისი თემატიკა

- ხარისხის უზრუნველყოფის სცენარის განხილვა
- ხარისხის უზრუნველყოფის სცენარის შედგენა

პროექტის დაგეგმარებაში, ჩვენ უნდა განვიხილეთ სცენარის ჩამოყალიბება ვებ საიტის გამოყენების თვალსაზრისით. არსებული სცენარების მიხედვით შეგვიძლია შევიმუშაოთ ვებ საიტის შემოწმების სცენარიც და შემდგომ ტესტირების პროცესში, ვთხოვოთ შემოწმების სუბიექტებს სცენარის მიხედვით შეასრულონ კონკრეტული დავალება ჩვენს მიერ შემუშავებულ ვებ საიტზე.

სცენარის ჩამოყალიბებისას, გაითვალისწინეთ ყველა საფეხური, რომელიც მომხმარებელს ექნება გასავლელი. იქნება თუ არა მომხმარებლისთვის შესაძლებელი იმ ამოცანის შესრულება, რაც სცენარში ექნება, დამატებითი, დამხმარე ინფორმაციის გარეშე? ტესტირების პროცესში, თქვენ ითვისებთ ფასილიტატორის როლს და სუბიექტს არ ეხმარებით ამოცანის შესრულებაში. გაითვალისწინეთ, რომ მომხმარებლების უმრავლესობას დამხმარე არ ეყოლება და კონკრეტული ამოცანის შესრულების ხერხი თავად უნდა იპოვოს.

შემოწმების პროცესში, სცენარი შეგვიძლია ამობეჭდილი სახით მივაწოდოთ შემმოწმებელ სუბიექტს და შემდგომ თვალყური ვადევნოთ მის მოქმედებებს. მაგალითისთვის, განვიხილოთ რამდენიმე სცენარი, რომელიც ელექტრონული კომერციის ტიპის ვებ საიტისთვის იყო შედგენილი.

### სცენარი

1. გაარკვიეთ, არის თუ არა ელექტრონულ მაღაზიაში საბავშვო აკუსტიკური გიტარა
   *აქ ვამოწმებთ მომხმარებლის კომფორტს საიტის ნავიგაციის სტრუქტურასთან ან/და ძიების გამოყენებასთან მიმართებაში. რამდენად მარტივად პოულობს მომხმარებელი სასურველ ინფორმაციას/პროდუქტს და თუ აქვს რაიმე კომენტარი პროცესში.*

2. გაარკვიეთ, რა ასაკის ბავშვებისთვის არის მისაღები ეს პროდუქტი
   *აქ ვამოწმებთ იპოვის და წაიკითხავს თუ არა მომხმარებელი პროდუქტის აღწერას.*

3. შეიძინეთ პროდუქტი და გადაიხადეთ პლასტიკური ბარათით
   *აქ ვამოწმებთ, მუშაობს თუ არა პლასტიკური ბარათით გადახდის სერვისი. საჭიროების შემთხვევაში შეგიძლიათ მომხმარებელს სატესტო ბარათის მონაცემები მიაწოდოთ, რომელიც ხშირად შესაბამისი ბანკის მიერ არის მოწოდებული.*

4. ადგილზე მიტანის მომსახურებისთვის, მიუთითეთ მეგობრის მისამართი
   *აქ ვამოწმებთ, შეძლებს თუ არა მომხმარებელი რეგისტრაციის პროცესში პირადი მისამართის მითითებას და შემდგომ ადგილზე მიტანის მისამართის მითითებას პროდუქტის შეძენის პროცესში?*

5. მოითხოვეთ დასტურის წერილი, როცა პროდუქტი ადგილზე იქნება მიტანილი
   *აქ ვამოწმებთ, იპოვის თუ არა მომხმარებელი ამ პარამეტრს პროდუქტის შეძენისას*

როგორც ხედავთ, ამ სცენარით ჩვენ ერთდროულად ვამოწმებთ რამდენიმე პროცესს, რომელსაც მომხმარებელს ექნება გასავლელი, სასურველი შედეგის მისაღწევად.

ტესტირების პროცესს სასურველია ესწრებოდეს მხოლოდ ერთი ფასილიტატორი, რათა ტესტირების სუბიექტს არ ჰქონდეს დისკომფორტი. ტესტირების პროცესში, ჩაინიშნეთ ყველა საინტერესო მოვლენა ან/და კომენტარი (შეძლებისდაგვარად ჩაწერეთ პროცესის ვიდეო). მოგვიანებით შეგიძლია განიხილოთ მიღებული კომენტარები სამუშაო გუნდის წევრებთან.

**სავარჯიშო**

შეადგინეთ დავალების სცენარი და განიხილეთ რომელიმე ვებ საიტის მეგობართან ერთად.

# 5. ვებსაიტის მარკირება - html

## 5.1. ვებგვერდის სტრუქტურის აგება

**მიმდინარე პარაგრაფის თემატიკა**

- თანამედროვე ვებ ტექნოლოგიები, ვებ მარკირების დანიშნულება და გამოყენების სფერო
- კოდის წერის სინტაქსი და ეთიკა
- ვებგვერდის შენახვის ფორმატი
- ვებგვერდის მაკეტის შექმნის ტეგები
- სათაურების ტეგები
- აბზაცის ფორმატირების ტეგები
- სიმბოლოების ფორმატირების ტეგები
- სიების შექმნის ტეგები
- ხაზოვანი და ბლოკური ტეგები

### თანამედროვე ვებ ტექნოლოგიები, ვებ მარკირების დანიშნულება და გამოყენების სფერო

ინტერნეტში არსებული ნებისმიერი ინფორმაცია, რომლთა დათვალიერება შესაძლებელია ინტერნეტ-ბრაუზერების საშუალებით, ინახება რომელიმე ვებ სერვერზე საიტის სახით. ვებ საიტი შედგება ვებგვერდებისაგან, რომელთა მარკირება და ერთმანეთთან დაკავშირება ხდება HTML (Hyper Text Markup Language) ჰიპერტექსტების მარკირების ენის საშუალებით.

**HTML** ინტერნეტის ფუნდამენტურ საბაზო ტექნოლოგიას წარმოადგენს. HTML ენა ბრიტანელმა ფიზიკოსმა ტიმ ბერნს ლიმ (Tim Berners-Lee) შეიმუშავა 1989 წელს ჟენევაში. თავდაპირველად ის მეცნიერულ-ტექნიკური ინფორმაციის გასაცვლელად შეიქმნა, მას შემდეგ მუდმივად ვითარდება.

| ვერსიები | წელი |
|---|---|
| HTML | 1991 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.01 | 1999 |
| XHTML | 2000 |
| HTML5 | 2014 |

ვებგვერდის ბრაუზერში ასახვისათვის გამოიყენება HTML ენა, რომლის საშუალებით ხორციელდება ვებ-გვერდის სტრუქტურის (Layout) შექმნა, ხოლო გაფორმება ანუ ვიზუალური სახის მიცემა ხდება CSS (Cascading Style Sheets - კასკადური სტილების ცხრილები) სტილების საშუალებით.

html კოდის დასაწერად შესაძლებელია გამოვიყენოთ ნებისმიერ ტექსტური რედაქტორი, თუმცა რეკომენდირებულია ვებ ედიტორების გამოყენება, ვინაიდან ვებ ედიტორში განსხვავებული ფერებით აისახება html კოდი და ვებგვერდის ტექსტური ნაწილი.

ვებ ედიტორებია: notepad++, edit+, sublime Text ჩვენ სახელმძღვანელოში მაგალითების განსახილველად გამოვიყენებთ ღია კოდის პროგრამას - **notepad++,** რომლის გადმოწერა შესაძლებელია შემდეგი მისამართიდან: https://notepad-plus-plus.org/download/v6.8.6.html (სურ. 1.1)

Download-ით გადმოწერეთ საინსტალაციო პაკეტი და დააყენეთ თქვენს კომპიუტერზე.

ვებგვერდების დასათვალიერებლად გამოიყენება სპეციალური პროგრამა, რომელსაც ბრაუზერს უწოდებენ (browse - დათვალიერება). გავრცელებული ბრაუზერებია: Google Chrome, Mozilla Firefox, Internet Explorer, Apple Safari, Opera.


სურ. 1.1 . საინსტალაციო პაკეტის გადმოწერა

## კოდის წერის სინტაქსი და ეთიკა

ნებისმიერი html დოკუმენტი შედგება ბრაუზერში გამოსატანი ინფორმაციისაგან (ტექსტი, გრაფიკული ობიექტი) და მმართველი ელემენტებისაგან, რომელის სახელი თავსდება კუთხურ (< და >) ფრჩხილებს შორის და ეწოდება **ტეგი/თეგი** (Teg).

როგორც წესი გამოიყენება:

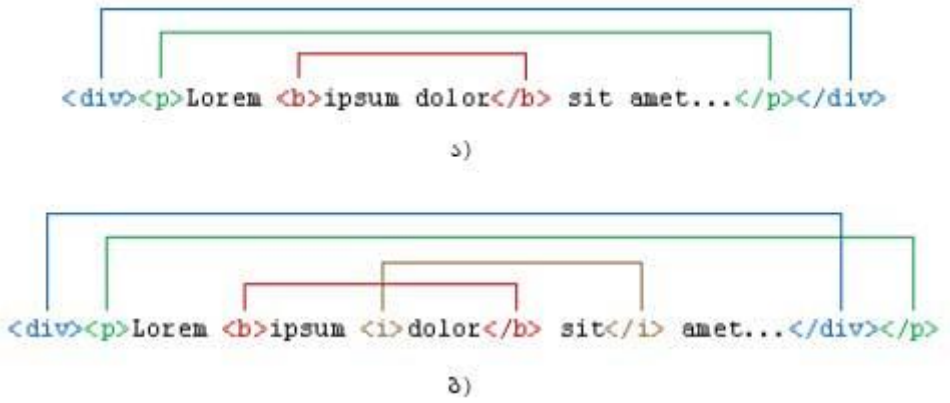
გამხსნელი ტეგი - **<ტეგის_სახელი>** მაგ. <html> , <head> , <title> , <body>

დამხურავი ტეგი - **</ტეგის_სახელი>** მაგ. </html> , </head> , </title> , </body>

ტეგების გახსნისა და დახურვის მიმდევრობა შემდეგნაირად გამოიყურება:

**<ტეგი 1><ტეგი 2><ტეგი 3> ..... </ტეგი 3></ტეგი 2></ტეგი 1>**

პირველად იხურება ბოლოს გახსნილი ტეგი და ა.შ. გახსნის უკუ თანმიმდევრობით, სხვა მიმდევრობით ტეგების განლაგებამ შეიძლება შეცდომა მოგცეთ (სურ 1.2).




სურ. 1.2 ტეგების დახურვის სტრუქტურა ა - მართებული, ბ - არასწორი

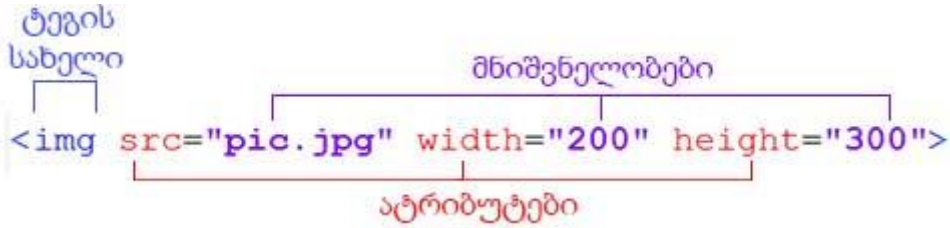
არსებობს გამონაკლისი ტეგები, რომელთათვისაც არ გამოიყენება დამხურავი ტეგი და იხურება თავის თავშივე მაგ: <img/>, <br/>.

ცალკეული ტეგის დანიშნულება დაწვრილებით აღწერილია შემდგომ თავებში.

თითოეულ ტეგს შეიძლება ჰქონდეს ატრიბუტი. შეიძლება ითქვას, რომ ატრიბუტები ტეგის თვისებების სახელებია, რომლებსაც გარკვეული მნიშვნელობის მიღება შეუძლია. ტეგის სახელი განსაზღვრავს ობიექტს, ატრიბუტის საშუალებით კი ხდება ამ ობიექტის სხვადასხვა თვისებაზე მნიშვნელობის მინიჭება. ატრიბუტის კონსტრუქცია - **თვისება="მნიშვნელობა"**.

ტეგები და ატრიბუტები აუცილებლად იწერება პატარა სიმბოლოებით, ატრიბუტის მნიშვნელობა იწერება ბრჭყალებში.

ერთ ტეგში შესაძლებელია რამოდენიმე ატრიბუტის გამოყენება, რომელიც ტეგისაგან და ერთმანეთისაგან ინტერვალითაა დაშორებული. (სურ. 1.3).




სურ. 1.3 ტეგის კონსტრუქცია

ატრიბუტები მხოლოდ გამხსნელ ტეგებს აქვთ, ხოლო დამხურავ ტეგებს არა. საბოლოოდ ტეგის კონსტრუქციას აქვს შემდეგი სახე (სურ. 1.4)


სურ. 1.4 ტეგის სრული კონსტრუქცია

html კოდის წერის ეთიკის ნორმების უმნიშვნელოვანესი თემაა კოდის კომენტირება.

კომენტარები შესაძლებელია განვათავსოთ HTML დოკუმენტში ნებისმიერ ადგილას, რადგანაც იგი არ აისახება ბრაუზერის მიერ ვებგვერდზე.

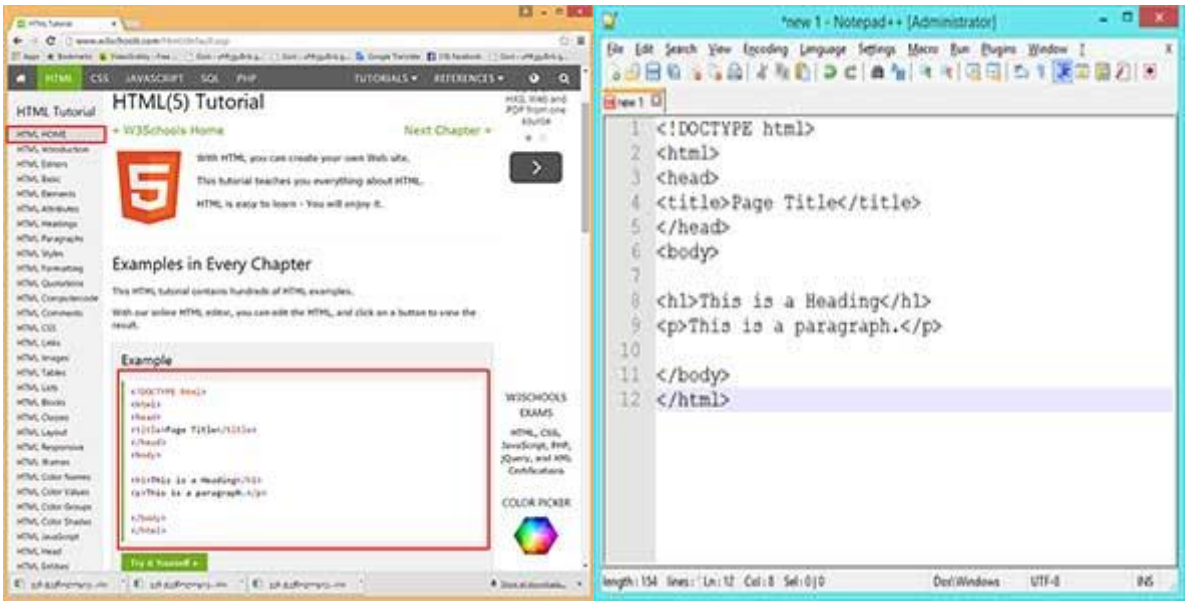
კომენტარში იწერება შენიშვნის/კოდის განმარტებითი ტექსტები, რომელიც განკუთვნილია ვებ დეველოპერისათვის.

<!-- ერთსტრიქონიანი კომენტარი -->

### ვებ გვერდის შენახვის ფორმატი

პირველი ვებგვერდის შესაქმნელად გავხსნათ ვებ ედიტორი (notepad++) http://www.w3schools.com– დანგადმოაკოპირეთშემდეგიკოდი (სურ. 1.5)

**გადაამოწმეთ html დოკუმენტის კოდირება.** იმისათვის რომ მას ჰქონდეს უნიკოდის მხარდაჭერა **Encoding** მენიუში აქტიური უნდა იყოს - **Encoding in UTF-8**



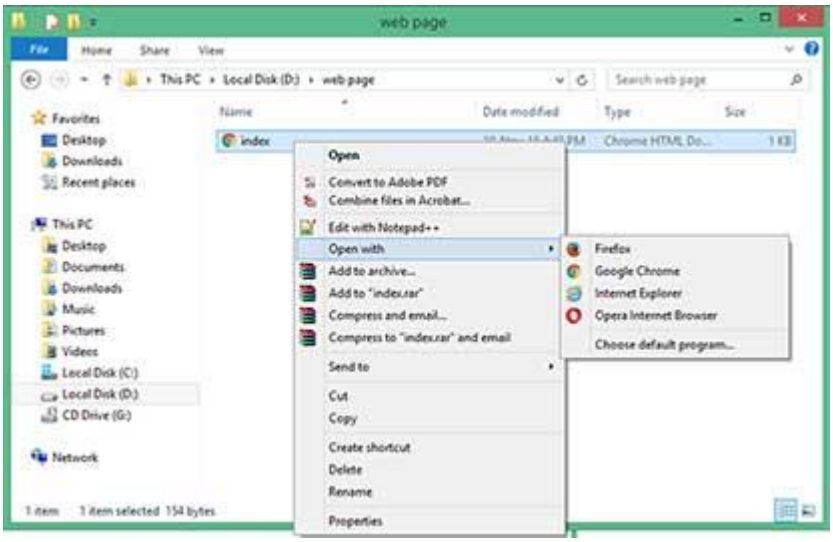

სურ. 1.5 პირველი html კოდი

html კოდი ტექსტური ფაილის სახით რომ არ შეინახოს, დოკუმენტის დამახსოვრებისას (File >Save) Save As დიალოგური ფანჯრის **file name** - ველში ვუთითებთ **ფაილის სახელი.html** (სურ.1.6)

თუ კოდის წერისათვის იყენებთ უმარტივეს ტექსტურ რედაქტორს notepad ფაილის შენახვისას Save As დიალოგურ ფანჯარაში **Encoding** - ველში აირჩიეთ - **UTF-8.**


სურ. 1.6 ვებგვერდის შენახვა

ვებგვერდის ნახვა შესაძლებელია ნებისმიერ ბრაუზერში. გახსენით საქაღალდე სადაც შეინახეთ თქვენი ფაილი და index.html ფაილის კონტექსტურ მენიუში (მაუსის მარჯვენა კლავიშზე დაჭერით) აირჩიეთ ბრძანება Open With და თქვენთვის სასურველი ბრაუზერი (სურ. 1.7).



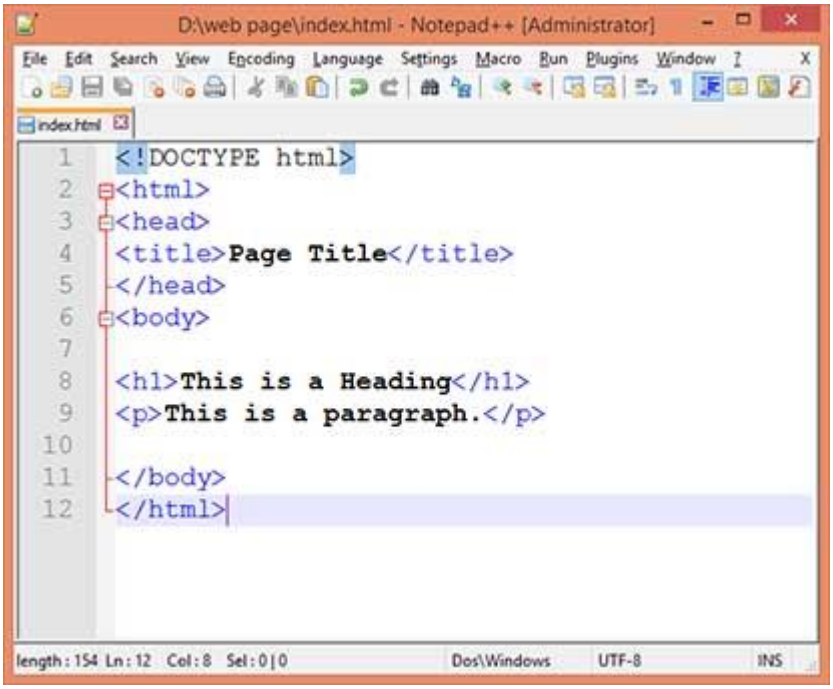

სურ. 1.7 ვებგვერდის გახსნა ბრაუზერში

ჩვენს მიერ შექმნილი პირველი ვებგვერდი შემდეგნაირად გამოიყურება (სურ. 1.8.)


სურ. 1.8 პირველი ვებგვერდი

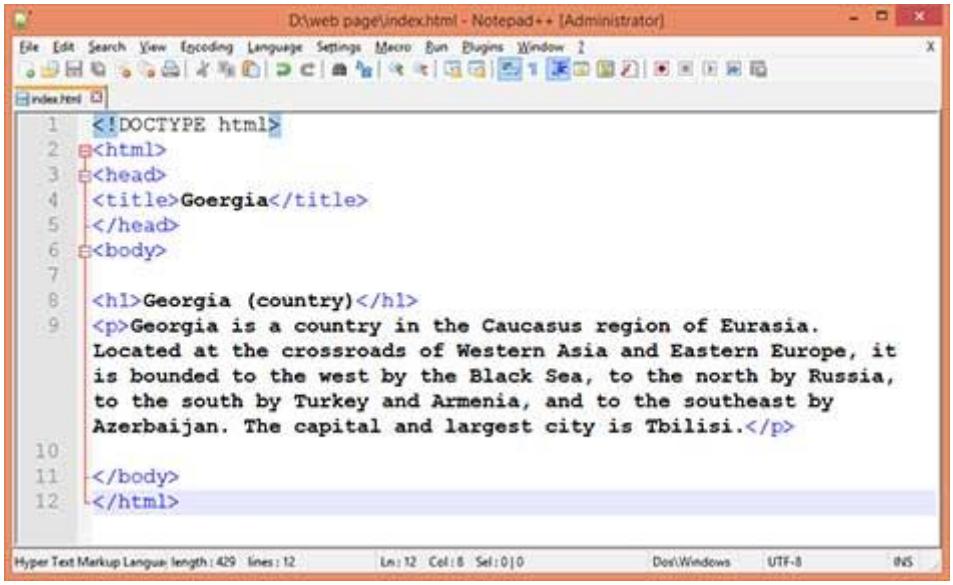
ვებგვერდის რედაქტირება შესაძლებელია ვებ ედიტორში. გავხსნათ ჩვენს მიერ შექმნილი index.html ფაილი ვებ ედიტორში - notepad++.

გახსენით საქაღალდე, სადაც შეინახეთ index.html ფაილი და კონტექსტურ მენიუდან (მაუსის მარჯვენა კლავიშზე დაჭერით) აირჩიეთ ბრძანება Edit with Notepad++ (სურ. 1.7). ფაილის გახსნა ასევე შესაძლებელია ვებ ედიტორიდან file > Open ბრძანებით. ვებ ედიტორში html კოდის ტეგები აისახება ლურჯი ფერით, ტექსტი - შავით (სურ 1.8).




სურ 1.8 პირველი index.html ფაილი

Html დოკუმენტში შეტანილი ცვლილებები (სურ.1.9) ბრაუზერში ვებგვერდზე რომ აისახოს, უნდა შევინახოთ რედაქტირებული ფაილი (Ctrl+S) და ბრაუზერში განვაახლოთ ვებგვერდი ინსტრუმენტების ველზე არსებული Reload ღილაკით ან კლავიატურიდან F5 ფუნქციონალური კლავიშით (სურ. 1.10).




სურ 1.9 index.html ფაილის რედაქტირებული ვერსია


სურ. 1.10 ვებგვერდის რედაქტირების შემდეგ

## ვებ გვერდის მაკეტის შექმნის ტეგები

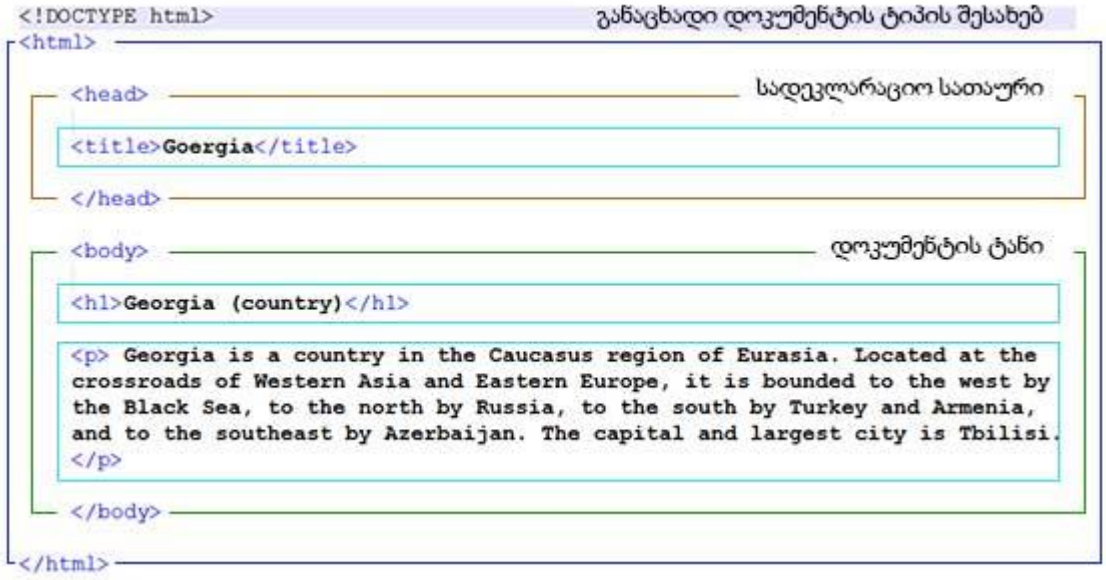
დოკუმენტის სტრუქტურის შექმნისა და მასში საჭირო ინფორმაციის შენახვისათვის გამოიყენება სხვადასხვა ელემენტები, რომლებიც მოიცავენ დოკუმენტის შექმნის ყველა საჭირო პუნქტს. ვებგვერდის შექმნისას გამოიყენება საბაზო კომპონენტები:

- განაცხადი დოკუმენტის ტიპის შესახებ;
- სადეკლარაციო სათაური;
- დოკუმენტის ტანი

**<!DOCTYPE>** - განაცხადი დოკუმენტის ტიპის შესახებ, ვერსიების მიხედვით სხვადასხვა სახის ჩანაწერი არსებობა. მაგალითისათვის იხ. ცხრილი 1.1

| ჩანაწერები | ვერსია |
|---|---|
| <!DOCTYPE html> | HTML5 |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> | HTML 4.01 |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> | XHTML 1.0 |

ცხრილი 1.1 განაცხადი დოკუმენტის ტიპის შესახებ




სურ. 1.11 html დოკუმენტის ზოგადი სტრუქტურა

**<html> ... </html>**- არის ვებ დოკუმენტის პირველი და ბოლო ტეგი, რომელიც მიუთითებს რომ აღნიშნული ფაილი არის ვებგვერდი და არა უბრალო ტექსტი

**<head> ... </head>** - სათაურის ბლოკი, სადაც იწერება ზოგადი ინფორმაცია ვებგვერდის შესახებ.

**<title> ... </title>** - დოკუმენტის სახელი, რომელიც გამოისახება ბრაუზერის სათაურის ველში (სურ. 1.12).

**<body>** ... </**body>** – დოკუმენტის ტანი, სადაც აღიწერება ვებგვერდის შემცველობა. (სურ. 1.12).


სურ. 1.12 ვებგვერდის სტრუქტურის ასახვა ბრაუზერში

ჩვენსმიერშექმნილივებგვერდისსათაურიდაშემცველობაწარმოდგენილიიყოლათინურიშრიფტით. ახლა ჩვენს მიერ შექმნილ index.html შევიტანოთ ცვლილებები და ავკრიფოთ სურ. 1.13 ა-ზე მოცემული ქართული ტექსტი (აუცილებელია გამოვიყენოთ ქართული უნიკოდ სიმბოლოები - KA). ბრაუზერი ავტომატურად ვერ აღიქვამს ქართულ დამწერლობას (სურ. 1.13 ა ).

იმისათვის რომ ბრაუზერმა სწორად აღიქვას ქართული ტექსტი აუცილებელია სათაურის ბლოკში (<head> - </head>) დავამატოთ <meta charset="UTF-8"> ტეგი.

html დოკუმენტი

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>საქართველოს შესახებ</title>
5  </head>
6  <body>
7  <h1>საქართველო</h1>
8  <p> საქართველო – სახელმწიფო ევრაზიაში, კავკასიაში, შავი ზღვის
   აღმოსავლეთ სანაპიროზე. ესაზღვრება ჩრდილოეთიდან რუსეთი, სამხრეთიდან
   თურქეთი და სომხეთი, და სამხრეთ-აღმოსავლეთიდან აზერბაიჯანი.
   ტრანსკონტინენტური ქვეყანა სამხრეთ-აღმოსავლეთ ევროპისა და დასავლეთ
   აზიის გასაყარზე მდებარეობს, თუმცა სოციოპოლიტიკურად და კულტურულად
   ევროპის ნაწილია.</p>
9  </body>
10 </html>
```

ვებ გვერდი

áƒ¡áƒáƒ¥áƒáƒ áƒ—áƒ•áƒ”áƒšáƒáƒ¡ áƒ¨áƒ”áƒ®áƒ”áƒ‘áƒ

áƒ¡áƒáƒ¥áƒáƒ áƒ—áƒ•áƒ”áƒšáƒ – áƒ¡áƒáƒ®áƒ”áƒšáƒ›áƒ¬áƒ˜áƒ¤áƒ áƒ”áƒ•áƒ áƒáƒ–áƒ˜áƒáƒ¨áƒ˜, áƒ™áƒáƒ•áƒ™áƒáƒ¡áƒ˜áƒáƒ¨áƒ˜, áƒ¨áƒáƒ•áƒ˜ áƒ–áƒ¦áƒ•áƒ˜áƒ¡ áƒáƒ¦áƒ›áƒáƒ¡áƒáƒ•áƒšáƒ”áƒ—áƒ˜ áƒ¡áƒáƒœáƒáƒáƒ˜áƒ áƒáƒ–áƒ”. áƒ”áƒ¡áƒáƒ–áƒ¦áƒ•áƒ áƒáƒ•áƒ¡ áƒ©áƒ áƒ“áƒ˜áƒšáƒ”áƒ—áƒ˜áƒ“áƒáƒœ áƒ áƒ£áƒ¡áƒ”áƒ—áƒ˜, áƒ¡áƒáƒ›áƒ®áƒ áƒ”áƒ—áƒ˜áƒ“áƒáƒœ áƒ—áƒ£áƒ áƒ¥áƒ”áƒ—áƒ˜ áƒ“áƒ áƒ¡áƒáƒ›áƒ®áƒ”áƒ—áƒ˜, áƒ“áƒ áƒ¡áƒáƒ›áƒ®áƒ áƒ”áƒ—-áƒáƒ¦áƒ›áƒáƒ¡áƒáƒ•áƒšáƒ”áƒ—áƒ˜áƒ“áƒáƒœ áƒáƒ–áƒ”áƒ áƒ‘áƒáƒ˜áƒ¯áƒáƒœáƒ˜. áƒ¢áƒ áƒáƒœáƒ¡áƒ™áƒáƒœáƒ¢áƒ˜áƒœáƒ”áƒœáƒ¢áƒ£áƒ áƒ˜ áƒ¥áƒ•áƒ”áƒ§áƒáƒœáƒ áƒ¡áƒáƒ›áƒ®áƒ áƒ”áƒ—-áƒáƒ¦áƒ›áƒáƒ¡áƒáƒ•áƒšáƒ”áƒ—áƒ˜áƒ¡ áƒ“áƒ áƒ“áƒáƒ¡áƒáƒ•áƒšáƒ”áƒ—áƒ˜ áƒ”áƒ•áƒ áƒáƒáƒ˜áƒ¡ áƒ“áƒ áƒáƒ–áƒ˜áƒ˜áƒ¡ áƒ’áƒáƒ¡áƒáƒ§áƒáƒ áƒ–áƒ”, áƒ—áƒ£áƒ›áƒªáƒ áƒ¡áƒáƒªáƒ˜áƒáƒáƒšáƒ˜áƒ¢áƒ˜áƒ™áƒ£áƒ áƒáƒ“ áƒ“áƒ áƒ™áƒ£áƒšáƒ¢áƒ£áƒ áƒ£áƒšáƒáƒ“ áƒ”áƒ•áƒ áƒáƒáƒ˜áƒ¡ áƒœáƒáƒ¬áƒ˜áƒšáƒ˜áƒ.

ა) არასწორი html კოდი - meta ტეგის გარეშე


ბ) მართებული html კოდი

სურ. 1.13 ვებგვერდზე ქართული ტექსტის ასახვა

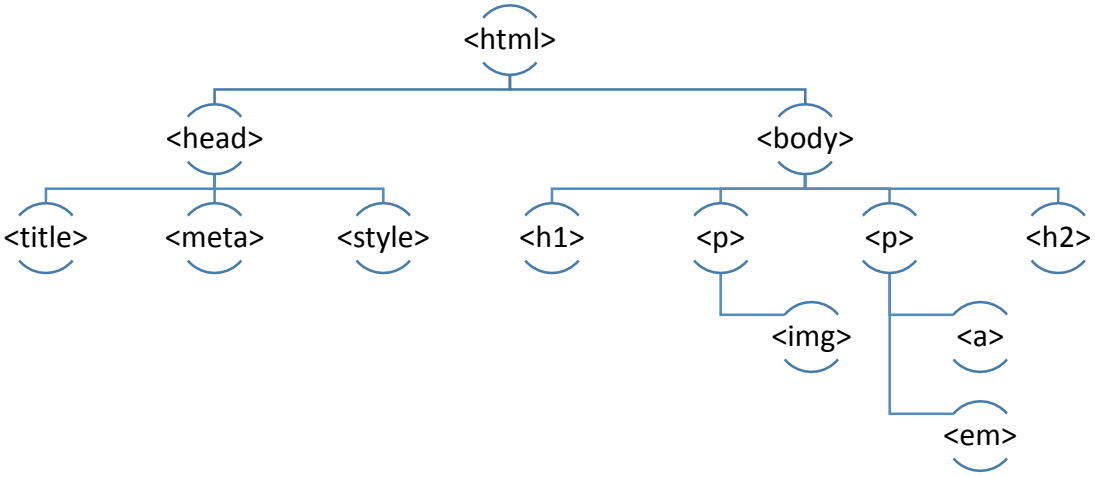
meta ტეგებში ინახება დამატებითი ინფორმაცია ვებგვერდის შესახებ. ამ ელემენტში არსებულ ინფორმაციას ბრაუზერები იყენებენ ვებგვერდის დამუშავებისას, ხოლო საძიებო სისტემები ინდექსირებისას.

<meta name="author"content="ავტორი"> ტეგი გამოიყენება ვებსაიტის ავტორის მისათითებლად, content ატრიბუტში უთითებთ ავტორს.

<meta name="description" content="ვებგვერდის ანოტაცია"> ტეგი გამოიყენება ვებ საიტის ანოტაციისათვის, content ატრიბუტში უთითებთ საიტის მოკლე ანოტაციას.

<meta name="keywords" content="საკვანძო სიტყვა1, საკვანძო სიტყვა2"> ტეგი გამოიყენება ვებ საიტის საკვანძო სიტყვების მისათითებლად.

ვებგვერდზე ტეგების განლაგებას აქვს ხისებრი სტრუქტურა და ტეგები ლოგიკურადაა ერთმანეთში ჩალაგებული სურ. 1.14




სურ. 1.14 html დოკუმენტში ტეგების განლაგების სტრუქტურა

### *პრაქტიკული დავალება*

შექმენით ვებგვერდის მაკეტი ვებსტანდარტის მოთხოვნათა დაცვით. რომელზეც ასახავთ ტექსტს „ეს ჩემი შექმნილი პირველი ვებგვერდია“.

### სათაურების ტეგები

სათაურები ვებ-გვერდის მნიშვნელოვან ელემენტებს წარმოადგენენ. ვებგვერდზე შესაძლებელია მნიშვნელობის მიხედვით სხვადასხვა დონის სათაურების შექმნა. ვებ-გვერდის სათაურებში არსებული ტექსტი მნიშვნელოვანწილად მოქმედებს საძიებო სისტემების მიერ საიტის ინდექსირებაზე, რადგან რობოტების უმეტესობა ძებნის პროცესში მნიშვნელოვან ყურადღებას აქცევენ სათაურების შემცველობას.

html დოკუმენტში შესაძლებელია ექვსი დონის სათაურის შექმნა შემდეგი ტეგებით: <**h1**>, <**h2**>, <**h3**>, <**h4**>, <**h5**> და <**h6**>. ყველაზე მნიშვნელოვანია პირველი დონის სათაური <h1> და მნიშვნელობის მიხედვით კლებულობს <h6>-მდე. დოკუმენტის სტრუქტურის შექმნისას საჭიროა დავიცვათ სათაურების სუბორდინაციული იერარქია.

**HTML დოკუმენტში პირველი დონის სათაური <h1> გამოიყენება მხოლოდ ერთხელ.**

ბრაუზერში პირველი დონის სათაური <h1> აისახება უფრო დიდი ზომის შრიფტით, ვიდრე მომდევნო დონეები (სურ. 1.15).


სურ. 1.15 html დოკუმენტში სათაურის დონეები

ბრაუზერში სათაურების (h1-h6) თვისებებისათვის (შრიფტის ზომა, სტილი) ავტომატურად მინიჭებული მნიშვნელობების ცვლილება შესაძლებელია სტილების კასკადური ენის (css) გამოყენებით. დაწვრილებით ეს საკითხი განხილულია შემდგომ თავში (იხ. თავი 2).

### აბზაცის ფორმატირების ტეგები

აბზაცები html დოკუმენტს ყოფს ლოგიკურ ნაწილებად და ერთმანეთისგან გამოიყოფა ერთი სტრიქონის დაშორებით. აბზაცის ორგანიზებისათვის html დოკუმენტში გამოიყენება <**p**> ტეგი (დამხურავი ტეგი - </**p**>).

აბზაცის და მასში მოთავსებული ტექსტის ბრაუზერში ვიზუალური თვისებების განსაზღვრა შესაძლებელია სტილების კასკადური ენის (css) გამოყენებით. დაწვრილებით ეს საკითხი განხილულია შემდგომ თავში (იხ. თავი 2).

**ნუ გამოიყენებთ:**

- აბზაცის ტეგების ცარიელ კონსტრუქციას - 

– აბზაცში სხვა აბზაცის ტეგს -  ააა ააა  ბბ ბბ  აააა აააა

html დოკუმენტი

```
<p>საქართველოს ტერიტორიაზე უძველესი დროიდან ადამიანთა ცხოვრების
ფაქტს ადასტურებს დმანისში ჩატარებული არქეოლოგიური გათხრები.
დმანისში აღმოჩენილი ადამიანის ჩონჩხის ფრაგმენტები უძველესია მთელს
ევრაზიაში და მისი ასაკი 1 800 000 წლის არის. </p>
<p>ქართველების პირველი პოლიტიკური გაერთიანება დიაოხი და კოლხა
მდინარე ჭოროხის აუზში ძვ.წ II ათასაწლეულის ბოლოს შეიქმნა, მათ
მხოლოდ რამდენიმე საუკუნე იარსებეს. ისინი დაამხეს ჩრდილოეთიდან
შემოჭრილმა მომთაბარე ტომებმა. ძვ.წ VI საუკუნეში ჩამოყალიბდა
ეგრისის ანუ კოლხეთის სამეფო. </p>
<p>ძვ.წ IV საუკუნეში აღმოსავლეთ საქართველოში შეიქმნა იბერიის
სამეფო. სწორედ იბერიის მეფეს ფარნავაზს უკავშირდება ქართული
დამწერლობის შექმნა. </p>
```

ვებ გვერდი

საქართველოს ტერიტორიაზე უძველესი დროიდან ადამიანთა ცხოვრების ფაქტს ადასტურებს დმანისში ჩატარებული არქეოლოგიური გათხრები. დმანისში აღმოჩენილი ადამიანის ჩონჩხის ფრაგმენტები უძველესია მთელს ევრაზიაში და მისი ასაკი 1 800 000 წლის არის.

ქართველების პირველი პოლიტიკური გაერთიანება დიაოხი და კოლხა მდინარე ჭოროხის აუზში ძვ.წ II ათასაწლეულის ბოლოს შეიქმნა, მათ მხოლოდ რამდენიმე საუკუნე იარსებეს. ისინი დაამხეს ჩრდილოეთიდან შემოჭრილმა მომთაბარე ტომებმა. ძვ.წ VI საუკუნეში ჩამოყალიბდა ეგრისის ანუ კოლხეთის სამეფო.

ძვ.წ IV საუკუნეში აღმოსავლეთ საქართველოში შეიქმნა იბერიის სამეფო. სწორედ იბერიის მეფეს ფარნავაზს უკავშირდება ქართული დამწერლობის შექმნა.

სურ. 1.16 აბზაცის ფორმატირება

ასეთმა კონსტრუქციებმა შეიძლება ბრაუზერების სხვადასხვა ვერსიებში არ იმუშაოს და კოდის წერის სტანდარტიც დარღვეულია.

ახალ სტრიქონზე გადასვლისათვის ისე, რომ არ დახუროს მიმდინარე აბზაცი გამოიყენება **<br>** ტეგი. ეს ელემენტი არ მოითხოვს დამხურავ ტეგს, მაგრამ რეკომენდირებულია ის დაიხუროს გამხსნელ ტეგშივე **<br/>,** რათა ყველა ბრაუზერმა ასახოს კორექტულად.

ვებგვერდზე ტექსტის განლაგება განისაზღვრება ტეგების საშუალებით და არ აქვს მნიშვნელობა ვებ ედიტორში მათ განლაგებას (სურ. 1.17). ედიტორში ახალ აბზაცზე enter კლავიშით გადატანილი ტექსტი ბრაუზერში ერთ სტრიქონზე აისახება. ვინაიდან ბრაუზერი ინფორმაციის განლაგებისათვის იყენებს მხოლოდ ტეგებს.

html დოკუმენტი

```
<p>
ჩემი ხატია სამშობლო,
სახატე მთელი ქვეყანა,
განათებული მთა-ბარი,
წილნაყარია ღმერთთანა.
თავისუფლება დღეს ჩვენი
მომავალს უმღერს დიდებას,
ცისკრის ვარსკვლავი ამოდის
ამოდის და ორ ზღვას შუა ბრწყინდება,
დიდება თავისუფლებას,
თავისუფლებას დიდება!
</p>
```

ვებგვერდი

ჩემი ხატია სამშობლო, სახატე მთელი ქვეყანა, განათებული მთა-ბარი, წილნაყარია ღმერთთანა. თავისუფლება დღეს ჩვენი მომავალს უმღერს დიდებას, ცისკრის ვარსკვლავი ამოდის ამოდის და ორ ზღვას შუა ბრწყინდება, დიდება თავისუფლებას, თავისუფლებას დიდება!

სურ. 1.17 არასწორად განაწილებული აბზაცი

იმისათვის რომ, ვებ ედიტორში გამოყენებული განლაგება ბრაუზერში უცვლელად რომ აისახოს გამოიყენება **<pre>** ტეგი (სურ. 1.18).

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<pre>
ჩემი ხატია სამშობლო,
სახატე მთელი ქვეყანა,
განათებული მთა-ბარი,
წილნაყარია ღმერთთანა.
თავისუფლება დღეს ჩვენი
მომავალს უმღერს დიდებას,
ცისკრის ვარსკვლავი ამოდის
ამოდის და ორ ზღვას შუა ბრწყინდება,
დიდება თავისუფლებას,
თავისუფლებას დიდება!
</pre>
```

ჩემი ხატია სამშობლო,
სახატე მთელი ქვეყანა,
განათებული მთა-ბარი,
წილნაყარია ღმერთთანა.
თავისუფლება დღეს ჩვენი
მომავალს უმღერს დიდებას,
ცისკრის ვარსკვლავი ამოდის
ამოდის და ორ ზღვას შუა ბრწყინდება,
დიდება თავისუფლებას,
თავისუფლებას დიდება!

სურ. 1.18 <pre> ტეგის გამოყენება

ციტატების, ვრცელი ფრაზებისა და გამონათქვამებისათვის გამოიყენება **<blockquote>**ტეგი. მასში არსებულ ტექსტი ბრაუზერში მარცხნიდან ტაბულაციით შეწეული აისახება. <blockquote> ... </blockquote> ელემენტში დასაშვებია ფორმატირების სხვა ელემენტების გამოყენება (სურ. 1.19).

ვებგვერდის ლოგიკურ ნაწილებად დასაყოფად გამოიყენება <hr> ტეგი და ბრაუზერში ჰორიზონტალური ხაზის სახით აისახება. აღნიშნულ ტეგს არ აქვს დამხურავი ტეგი და იხურება გამხსნელ ტეგშივე <hr/>

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<blockquote>
<p>ჩემი ხატია სამშობლო,<br/>
სახატე მთელი ქვეყანა,<br/>
განათებული მთა-ბარი,<br/>
წილნაყარია ღმერთთანა.
</p>
<p>თავისუფლება დღეს ჩვენი<br/>
მომავალს უმღერს დიდებას,<br/>
ცისკრის ვარსკვლავი ამოდის<br/>
ამოდის და ორ ზღვას შუა ბრწყინდება,
</p>
<p>
დიდება თავისუფლებას,<br/>
თავისუფლებას დიდება!
</p>
</blockquote>
```

ჩემი ხატია სამშობლო,
სახატე მთელი ქვეყანა,
განათებული მთა-ბარი,
წილნაყარია ღმერთთანა.

თავისუფლება დღეს ჩვენი
მომავალს უმღერს დიდებას,
ცისკრის ვარსკვლავი ამოდის
ამოდის და ორ ზღვას შუა ბრწყინდება,

დიდება თავისუფლებას,
თავისუფლებას დიდება!

სურ. 1.19 <blockquote> ტეგის გამოყენება

### სიმბოლოების ფორმატირების ტეგები

**** ტეგი განკუთვნილია ტექსტზე აქცენტებისათვის და ბრაუზერში აისახება მუქად.
**** გამოყოფს ტექსტის ფრაგმენტს და ბრაუზერში აისახება - *დახრილად*
**<abbr>** განსაზღვრავს ტექსტს, როგორც აბრევიატურას, რომლის **title** ატრიბუტში იწერება აბრევიატურის გაშიფვრა. ბრაუზერში აბრევიატურისა სრული ტექსტი აისახება, შესაბამის აბრევიატურასთან მაუსის მიტანისას. საძიებო სისტემები ყურადღებას აქცევენ აბრევიატურის სრულ ვარიანტს.

**<q>** ტეგი გამოიყენება ვებ-გვერდზე მოკლე ფრაზის გამოსატანად და მასში არსებულ ტექსტს ბარაუზერი გამოყოფს როგორც ციტატას. ზოგადად ეს ტექსტი ბრაუზერში აისახება დახრილი შრიფტით ან ბრჭყალებით.

**<sup>** ტეგი გამოიყენება ტექსტის ზედა ინდექსად გამოსახვისათვის.

**<sub>** ტეგი გამოიყენება ტექსტის ქვედა ინდექსად გამოსახვისათვის.

html დოკუმენტი

```
<p>
<strong>პროფესიული განათლების კვირეული </strong>
</p>
<p>საქართველოს <strong>WorldSkills Georgia</strong>-ს ფარგლებში, გერმანიის საერთაშორისო თანამშრომლობის
საზოგადოების  <abbr title="Gesellschaft für Internationale Zusammenarbeit">(GIZ) </abbr>
მხარდაჭერით, ქვეყნის მასშტაბით პროფესიული განათლების კვირეული ჩატარდა. </p>
<p>კვირეულის დასკვნითი ღონისძიება <em> მოსწავლე-ახალგაზრდობის ეროვნული სასახლეში </em> გაიმართა, რომელსაც
საქართველოს განათლებისა და მეცნიერების მინისტრი თამარ სანიკიძე დაესწრო.</p>
<p>პროფესიული განათლების დევიზია<q>გახდი პროფესიონალი და დასაქმდი</q> </p>

<p>აღნიშნული კვირეულის ფარგლებში სტუდენტთა ნამუშევრების გამოფენა გაიხსნება 1 ნოემბერს 15<sup>00</sup>სთ–ზე.
</p>
```

ვებგვერდი

**პროფესიული განათლების კვირეული**

საქართველოს **WorldSkills Georgia**-ს ფარგლებში, გერმანიის საერთაშორისო თანამშრომლობის საზოგადოების (GIZ) მხარდაჭერით, ქვეყნის მასშტაბით პროფესიული განათლების კვირეული ჩატარდა.

Gesellschaft für Internationale Zusammenarbeit

კვირეულის დასკვნითი ღონისძიება *მოსწავლე-ახალგაზრდობის ეროვნული სასახლეში* გაიმართა, რომელსაც საქართველოს განათლებისა და მეცნიერების მინისტრი თამარ სანიკიძე დაესწრო.

პროფესიული განათლების დევიზია"გახდი პროფესიონალი და დასაქმდი"

აღნიშნული კვირეულის ფარგლებში სტუდენტთა ნამუშევრების გამოფენა გაიხსნება 1 ნოემბერს $15^{00}$სთ–ზე.

სურ. 1.20 ტექსტის ფორმატირების ტეგები

### *პრაქტიკული დავალება*

შექმენით (განათლებისა და მეცნიერების სამინისტროს ვებ საიტზე ბოლოს გამოქვეყნებული სიახლის იდენტური) ვებგვერდის სტრუქტურა აბზაცებისა და ტექსტის ფორმატირების ტეგების გამოყენებით.

### სიების შექმნის ტეგები

სიები ვებგვერდზე გამოიყენება მონაცემების ორგანიზებისა და დაჯგუფებისათვის. იგი შეიძლება გამოყენებულ იქნეს მენიუს, სტრუქტურების და სხვა მსგავსი ობიექტების შესაქმნელად.

Html დოკუმენტში გამოყოფენ სამი ტიპის სიებს

- მარკირებული/დაუნომრავი – Unordered List
- დანომრილი – Ordered List
- განმარტებითი - Description list

**მარკირებული სიის** შემთხვევაში მისი შემცველი ყოველი პუნქტის წინ გამოისახება სხვადასხვა სახის სიმბოლო. ასეთ სიებს არათანმიმდევრულ/დაუნომრავ სიებსაც უწოდებენ. რადგან ამ ჩამონათვალის ელემენტებში რიგითობას მნიშვნელობა არა აქვს.

მარკირებული სიის (Unordered List) ორგანიზებისათვის გამოიყენება **<ul>** ტეგი, როლის დახურვა **</ul>** ხდება სიის ბოლოს.

მარკირებული სიის პუნქტების შესაქმნელად გამოიყენება **<li>** ტეგი, რომლის დახურვა **</li>** აუცილებელია შესაბამისი პუნქტის ბოლოს (სურ.1.21).

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<!DOCTYPE html>
<html>
<head>
<title>საქართველოს კუთხეები</title>
<meta charset="UTF-8">
</head>
<body>

<p>საქართველოს კუთხეები:</p>
<ul>
    <li>ქართლი</li>
    <li>კახეთი</li>
    <li>იმერეთი</li>
    <li>გურია</li>
    <li>სამეგრელო</li>
</ul>

</body>
</html>
```

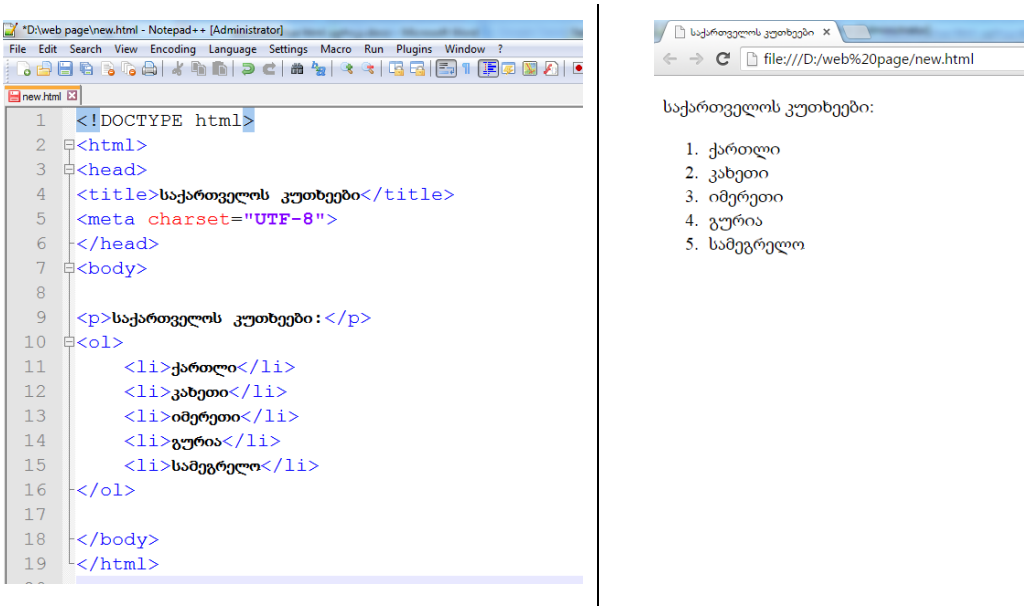
საქართველოს კუთხეები:

- ქართლი
- კახეთი
- იმერეთი
- გურია
- სამეგრელო

სურ. 1.21 მარკირებული სია

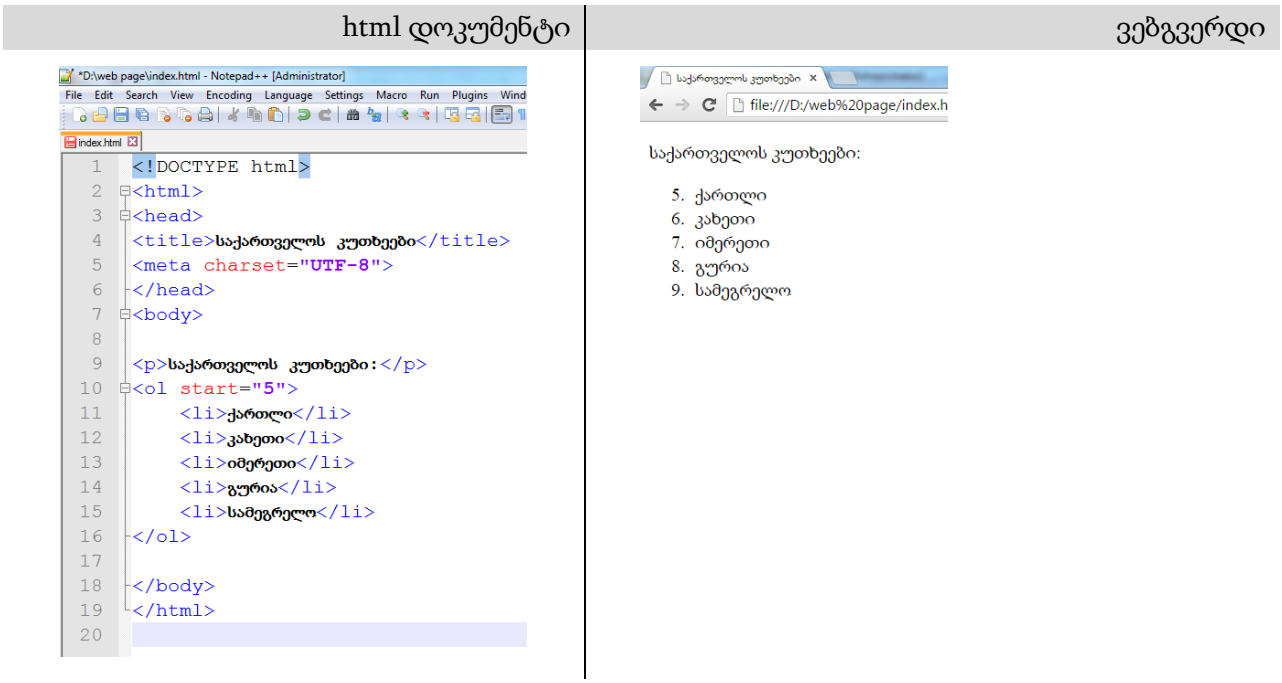
**დანომრილი სია** გამოიყენება როცა საჭიროა რიგითობის დაცვა, მას თანმიმდევრულ სიასაც უწოდებენ. დანომრილი სიის (Ordered List) შესაქმნელად გამოიყენება **<ol>** ტეგი, რომელიც იხურება **</ol>** სიის ბოლოს. ისევე როგორც მარკირებულ სიაში ჩამონათვალის ელემენტების შესაქმნელად გამოიყენება **<li>** ტეგი, რომლის დახურვა **</li>** აუცილებელია შესაბამისი ელემენტის ბოლოს (სურ.1.22).

| html დოკუმენტი | ვებგვერდი |
|---|---|




სურ. 1.21 დანომრილი სია

დანომრილი სიის ორგანიზებისას შესაძლებელია start ატრიბუტის გამოყენება, რომლიც მნიშვნელობაც განსაზღვრავს სიის პირველი ელემენტის ნომერს (სურ.1.23).




სურ. 1.23 დანომრილი სიის საწყისი რიცხვის განსაზღვრა

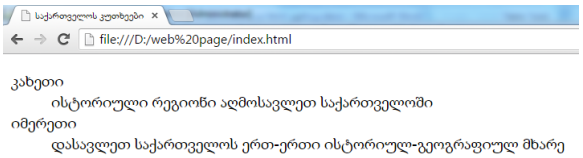
**განმარტებითი სია** გამოიყენება სიტყვების, ფრაზების ან ტერმინების განმარტებისათვის. განმარტებითი სია შედგება ორი ნაწილისაგან, პირველში აღიწერება განსამარტებელი სიტყვას, ხოლო მეორში ამ სიტყვის განმარტება.

განმარტებითი სიის Description list ორგანიზებისათვის გამოიყენება **<dl>** ტეგი, რომელიც იხურება **</dl>** სიის ბოლოს. განსამარტი სიტყვების აღწერა ხდება **<dt>** ტეგით, განმარტების კი – **<dd>** ტეგით. ორივე ტეგი იხურება ცალკეული სიტყვისა **</dt>** და განმარტების **</dd>** ბოლოს (სურ.1.24).

html დოკუმენტი

```
<!DOCTYPE html>
<html>
<head>
<title>საქართველოს კუთხეები</title>
<meta charset="UTF-8">
</head>
<body>

<dl>
    <dt>კახეთი</dt>
        <dd> ისტორიული რეგიონი აღმოსავლეთ საქართველოში</dd>
    <dt>იმერეთი</dt>
        <dd>დასავლეთ საქართველოს ერთ-ერთი ისტორიულ-გეოგრაფიულ მხარე</dd>
</dl>

</body>
</html>
```

ვებგვერდი




სურ. 1.24 განმარტებითი სია

ვებგვერდზე ხშირად საჭირო ხდება **მრავალ დონიანი სიის** ორგანიზება მაგ. ჩამოშლადი მენიუს ან/და საიტის რუქის შესაქმნელად (სურ.1.25).

მრავალ დონიანი სიის კოდის წერისას, ყურადღება მიაქციეთ კოდის წერის ეთიკას და ყველა მომდევნო დონის სიის ამსახველი ტეგი გახსენით ერთი ტაბულაციით მარჯვნივ შეწეული. ქვედონის სიის დახურვის შემდეგ კი ერთი ტაბულაციით მარცხნივ გამოწევით გააგრძელეთ ზედა დონის სიის პუნქტების ასახვა (სურ.1.25 – html დოკუმენტი).

ბრაუზერში სხვდასხვა დონის სიის ელემენტები განსხვავებული სიმბოლოებით აისახება (სურ.1.25 – ვებგვერდი). პირველი დონე – შავი წრე (disc), მეორე დონე – თეთრი წრე (circle), მესამე დონე – შავი კვადრატი (square).

მრავალ დონიანი სიის ორგანიზებისათვის შესაძლებელია გამოვიყენოთ დანომრილი სია ან/და მარკირებისა და ნუმერაციის კომბინაცია.

html დოკუმენტი

D:\web page\index.html - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

index.html

```
<!DOCTYPE html>
<html>
<head>
<title>საიტის რუკა</title>
<meta charset="UTF-8">
</head>
<body>
<p>საიტის რუკა</p>
<ul>
<li>საქართველოს შესახებ</li>
    <ul>
    <li>ისტორია</li>
    <li>სახელმწიფო სიმბოლიკა</li>
        <ul>
        <li>დროშა</li>
        <li>გერბი</li>
        <li>ჰიმნი</li>
        </ul>
    <li>საქართველოს რეგიონები</li>
        <ul>
        <li>მლიერი რეგიონი მლიერი საქართველოსთვის</li>
        <li>საქართველოს რეგიონები</li>
        </ul>
    <li>კონსტიტუცია</li>
    <li>საქართველოს ოკუპირებული ტერიტორიები</li>
        <ul>
        <li>აფხაზეთი</li>
        <li>ცხინვალის რეგიონი</li>
        <li>სახელმწიფო სტრატეგია ოკუპირებული ტერიტორიების მიმართ</li>
        </ul>
    <li>ფოტოგალერეა</li>
    </ul>
<li>პრემიერ-მინისტრი</li>
    <ul>
    <li>სიახლეები</li>
        <ul>
        <li>2015 წლის სიახლეები</li>
        <li>2014 წლის სიახლეები</li>
        </ul>
    </ul>
</ul>
</body>
</html>
```

ვებგვერდი

საიტის რუკა

file:///D:/web%20page/index.html

საიტის რუკა

- საქართველოს შესახებ
  - ისტორია
  - სახელმწიფო სიმბოლიკა
    - დროშა
    - გერბი
    - ჰიმნი
  - საქართველოს რეგიონები
    - მლიერი რეგიონი მლიერი საქართველოსთვის
    - საქართველოს რეგიონები
  - კონსტიტუცია
  - საქართველოს ოკუპირებული ტერიტორიები
    - აფხაზეთი
    - ცხინვალის რეგიონი
    - სახელმწიფო სტრატეგია ოკუპირებული ტერიტორიების მიმართ
  - ფოტოგალერეა
- პრემიერ-მინისტრი
  - სიახლეები
    - 2015 წლის სიახლეები
    - 2014 წლის სიახლეები

სურ. 1.25 მრავალ დონიანი სია

დანომრილი სიის ელემენტები ბრაუზერში შეიძლება აისახოს არაბული ციფრებით (1,2,3 და ა.შ.); ლათინური ასომთავრული ან ხელნაწერი ალფავიტით (A,B,C… ან a,b,c.. ); რომაული ციფრებით (I, II, III …).

ბრაუზერში მარკირებისა და ნუმერაციის ვიზუალური თვისებების განსაზღვრა შესაძლებელია სტილების კასკადური ენის (css) გამოყენებით. დაწვრილებით ეს საკითხი განხილულია შემდგომ თავში (იხ. თავი 2).

### *პრაქტიკული დავალება*

შექმენით მრავალ დონიანი სიის სტრუქტურა. მაგ. ეროვნული სასწავლო გეგმების პორტალის საიტის რუქის იდენტური http://ncp.ge/ge/sitemap

### ხაზოვანი და ბლოკური ტეგები

ტეგები იყოფა **ბლოკური** (block-level) და **ხაზოვანი** (inline-level) ტიპის ტეგებად.

ბლოკური ტიპის ტეგებში განთავსებული ინფორმაცია ბრაუზერში აისახება ახალი სტრიქონიდან და შესაბამისი ტეგი მთელ ჰორიზონტალურ სტრიქონს იკავებს იმ შემთხვევაშიც კი თუ ამ ტეგებში მხოლოდ ერთი სიტყვაა მოთავსებული. დიდი ზომის ტექსტის შემთხვევაში ერთი სტრიქონის შევსების შემდეგ ტექსტი ავტომატურად გადადის მეორე სტრიქონზე.

ჩვენს მიერ განხილული ტეგებიდან ბლოკური ტიპის ტეგებია: <p>, <blockquote>, <pre>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <ol>, <ul>, <dl>, <li>, <dd>, <dt>

ვებგვერდზე დიდი ზომის ინფორმაციის გამოსაყოფად ხშირად გამოიყენება ** ** ტეგი.

ხაზოვანი ტეგები ბრაუზერში იკავებენ მხოლოდ მათში მოთავსებული ტექსტის ზომის ადგილს. შეიძლება ერთ სტრიქონზე რამდენიმე ხაზოვანი ტეგი იყოს ასახული.

ჩვენს მიერ განხილული ტეგებიდან ხაზოვანი ტეგებია: <abbr>, <br>, , <q>, , <sub>, <sup>.

ტექსტის მცირე ფრაგმენტის გამოსაყოფად ასევე ხშირად გამოიყენება **** ხაზოვანი ტეგი.

ბლოკურ ტეგებში შეიძლება გამოვიყენოთ სხვა ბლოკური ან/და ხაზოვანი ტეგები. ხაზოვან ტეგებში კი დაუშვებელია ბლოკური ტეგის გამოყენება.

ცალკეული ტეგების ვიზუალური თვისებების განსაზღვრა შესაძლებელია სტილების კასკადური ენის (css) გამოყენებით. დაწვრილებით ეს საკითხი განხილულია შემდგომ თავში (იხ. თავი 2).

*კითხვები თვითშეფასებისათვის*

1. **როგორ განიმარტება HTML?**
   a. Hyper Text Markup Language
   b. Hyperlinks and Text Markup Language
   c. Home Tool Markup Language
2. **რომელი ფორმატით უნდა შევინახოთ ვებგვერდი?**
   a. exe
   b. doc
   c. html
   d. ნებისმიერი ზემოჩამოთვლილი
3. **რომელი ტეგებია სავალდებულო ვებგვერდის სტრუქტურის შესაქმნელად**
   a. doctype, html, head, title და p
   b. html და body
   c. html, title, body და p
   d. doctype, html, head, title და body
4. **სტანდარტულად რა ჰქვია ვებ საიტის საწყის ფაილს**
   a. first.html
   b. page.html
   c. index.html
   d. index.exe
5. **Html დოკუმენტში აუცილებელია ყველა ტეგი იყოს დახურული**
   a. მცდარია
   b. ჭეშმარიტია
6. **რომელი ტეგი განსაზღვრავს ვებგვერდის პირველი დონის სათაურს**
   a. <heading>
   b. <h6>
   c. <head>
   d. <h1>
7. **რომელი ტეგი უზრუნველყოფს ვებ ბრაუზერის სათაურის ველში ვებგვერდის სათაურის ასახვას?**
   a. <body>
   b. <head>
   c. <title>
8. **რომელი პროგრამა უზრუნველყოფს ვებგვერდის დათვალიერებას?**
   a. ნებისმიერი ტექსტური რედაქტორი
   b. ნებისმიერი გრაფიკული რეადქტორი
   c. ნებისმიერ ოპერაციულ სისტემაში
   d. ნებისმიერი ბრაუზერი
9. **რომელი პროგრამაშია შესაძლებელი html კოდის აკრეფა**
   a. გრაფიკულ რედაქტორებში
   b. ინტერნეტ ბრაუზერებში
   c. ვებ ედიტორებში ან/და ტექსტურ რედაქტორში
10. **html მარკირების ენა უზრუნველყოფს ვებგვერდის ვიზუალურ გაფორმებას**
    a. მცდარია
    b. ჭეშმარიტია

11. html დოკუმენტში ტეგები აუცილებელია აიკრიფოს პატარა ასოებით
    a. მცდარია
    b. ჭეშმარიტია
12. რომელი ტეგი გამოიყენება ვებგვერდზე მნიშვნელოვანი ტექსტის გამოსაყოფად?
    a. <p>
    b. <br>
    c. 
    d. 
13. რომელი ტეგი გამოიყენება მარკირებული სიის ორგანიზებისათვის
    a. <li>
    b. <ol>
    c. <ul>
    d. <dl>
14. რომელი ტეგი გამოიყენება დანომრილი სიის ორგანიზებისათვის
    a. <li>
    b. <ol>
    c. <ul>
    d. <dl>

15. რომელი ტეგი უზრუნველყოფს სიის ცალკეული ელემენტის აღწერას
    a. <br>
    b. <ol>
    c. <li>
    d. <ul>

16. რომელი ატრიბუტი განსაზღვრავს ნუმერირებული სიის საწყის რიცხვს?
    a. type
    b. number
    c. start number
    d. start
17. რომელი ტეგი გამოიყენება აბზაცის შესაქმნელად?
    a. <p>
    b. <br>
    c. <ul>
    d. <bloqoute>
18. რომელი ტეგი უზრუნველყოფს ახალ სტრიქონზე გადასვლას?
    a. <line>
    b. <enter>
    c. <br>
19. რომელი ტეგი უზრუნველყოფს ზედა ინდექსში გადაყვანას
    a. <sub>
    b. <down>

c. <sup>
d. <up>

**20. რომელი ტეგი უზრუნველყოფს ქვედა ინდექსში გადაყვანას**

a. <sub>
b. <down>
c. <sup>
d. <up>

## 5.2. ვებგვერდზე ობიექტებისა და ბმულების ასახვა

**მიმდინარე პარაგრაფის თემატიკა**

- გრაფიკული და მულტიმედია ობიექტის ფორმატები და მათი ვებგვერდზე ასახვის საშუალებები
- ბმულის ტიპები და პარამეტრები
- ცხრილის ტეგები და მისი პარამეტრები

### გრაფიკული და მულტიმედია ობიექტის ფორმატები და მათი ვებგვერდზე ასახვის საშუალებები

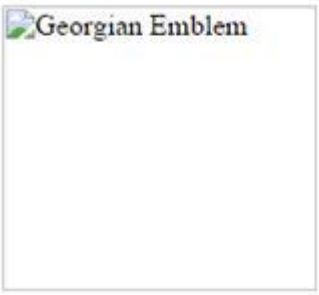
ვებგვერდზე ინფორმაციის წარმოდგენის მრავალფეროვნების მიზნით შესაძლებელია გრაფიკული და მულტიმედია ობიექტის გამოყენება. წარმოუდგენელია ვებსაიტი გრაფიკული გამოსახულებების გარეშე. ვინაიდან გრაფიკული ობიექტების ვებგვერდზე ასახვა ზრდის საიტის მეხსიერების ზომას/მოცულობას, აუცილებელია მათი გამოყენებისას მიღებული ნორმების დაცვა.

ვებგვერდზე გამოსაყენებელი გამოსახულების პარამეტრები ოპტიმალურად უნდა შეირჩეს, ისე რომ არც ხარისხი გაუარესდეს და არც ვებგვერდის ბრაუზერში წარმოდგენა გართულდეს.

html დოკუმენტში გრაფიკული ობიექტის/სურათის ასახვა ხდება შემდეგი ტეგით (სურ. 1.25):

html დოკუმენტი

```
<img src="geo_emblem.png" alt="Georgian Emblem" title="საქართველოს
                    გერბი"  width="174" height="157"border="4"/>
```


ა) ბ)

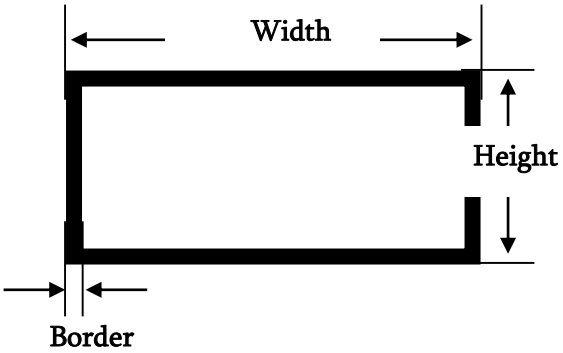
სურ. 1.25 ა) ვებგვერდზე გრაფიკული ობიექტის ასახვა

ბ) ვებგვერდზე ალტერნატიული ტექსტის ასახვა

**src** – ატრიბუტის მნიშვნელობაა ვებგვერდზე ასასახი სურათის მისამართი.

**alt** – ალტერნატიული ტექსტი, მისი მნიშვნელობა ვებგვერდზე აისახება როცა სხვადასხვა მიზეზის (ტექნიკური გაუმართაობა, მომხმარებელს გათიშული აქვს ბრაუზერში სურათის ასახვის რეჟიმი და ა.შ.) გამო არ ხდება სურათის ბრაუზერში ჩატვირთვა.

**title** – სათაური, რომლის მნიშვნელობა გამოჩნდება ბრაუზერში სურათზე მაუსის მაჩვენებლის მიტანისას ამოტივტივებდი შეტყობინების სახით.




სურ. 1.26 სურათის ზომისა და ჩარჩოს ატრიბუტები

**width**– სურათის სიგრძე. **height**– სურათის სიმაღლე. **border**– სურათის ჩარჩოს სისქე.

თუ არ მიუთითებთ სურათის ზომებს (სიგრძე და სიმაღლე) ბრაუზერში აისახება რეალური ზომით ანუ იმ სიგრძით და სიგანით, რაც სურათს გააჩნია. სურათის ზომის მცირე ცვლილებები დასაშვებია **width** და **height** ატრიბუტების საშუალებით, მაგრამ თუ საჭიროა სურათის ზომების მნიშვნელოვნად შეცვლა უმჯობესია გამოვიყენოთ გრაფიკული რედაქტორი. ვინაიდან html დოკუმენტში სურათის ზომის გაზრდით მისი ხარისხი იკლებს, ხოლო შემცირებით სურათის მეხსიერება რჩება უცვლელი და ვიზუალურად ბრაუზერში აისახება მცირე ზომის სურათი და საიტის ასახვა გართულდება.

**border** ატრიბუტის მნიშვნელობა ავტომატურად ნულის ტოლია, თუმცა შეიძლება მიეთითოს სასურველი ზომა ან მოეხსნას ჩარჩო (border=0).

img ტეგი არ მოითხოვს დამხურავ ტეგს (</img> - არასწორია) და ეს ტეგი იხურება გამხსნელ ტეგშივე ანუ თავის თავში (სურ. 1.25).

html დოკუმენტის მიმართ სურათის მდებარეობიდან გამომდინარე src ატრიბუტში სხვადასხვანაირად ხდება მისი მისამართის მითითება.

სურათის მისამართების მითითების ვარიანტები (სურ. 1.27):

1. html დოკუმენტი მიმართავს ინტერნეტში, რომელიმე სერვერზე განთავსებულ ფაილს, მაშინ src ატრიბუტის მნიშვნელობად მიეთითება ამ სურათის უნივერსალური მისამართი (URL).

```
<img src="http://mes.gov.ge/geo_emblem.png" alt="Georgian Emblem"/>
```

2. html დოკუმენტი და სურათი ერთ საქაღალდეშია (სურ. 1.27), მაშინ სურათის მისამართი ნაცვლად მიეთითება მხოლოდ ფაილის სახელი.

```
<img src="picture1.gif" alt="picture1"/>
```

3. html დოკუმენტი მიმართავს ქვესაქაღალდე Image-ში არსებულ picture2.gif ფაილს (სურ.1.27), მაშინ მიეთითება **ქვესაქაღალდის სახელი/ფაილის სახელი.**

```
<img src="Image/picture2.gif" alt="picture2"/>
```

4. html დოკუმენტი მიმართავს ქვესაქაღალდე Image-სი ქვესაქაღალდე New-ში არსებულ picture3.gif ფაილს (სურ.1.27), მაშინ მიეთითება **ქვესაქაღალდის სახელი/ქვესაქაღალდის სახელი/ფაილის სახელი.**

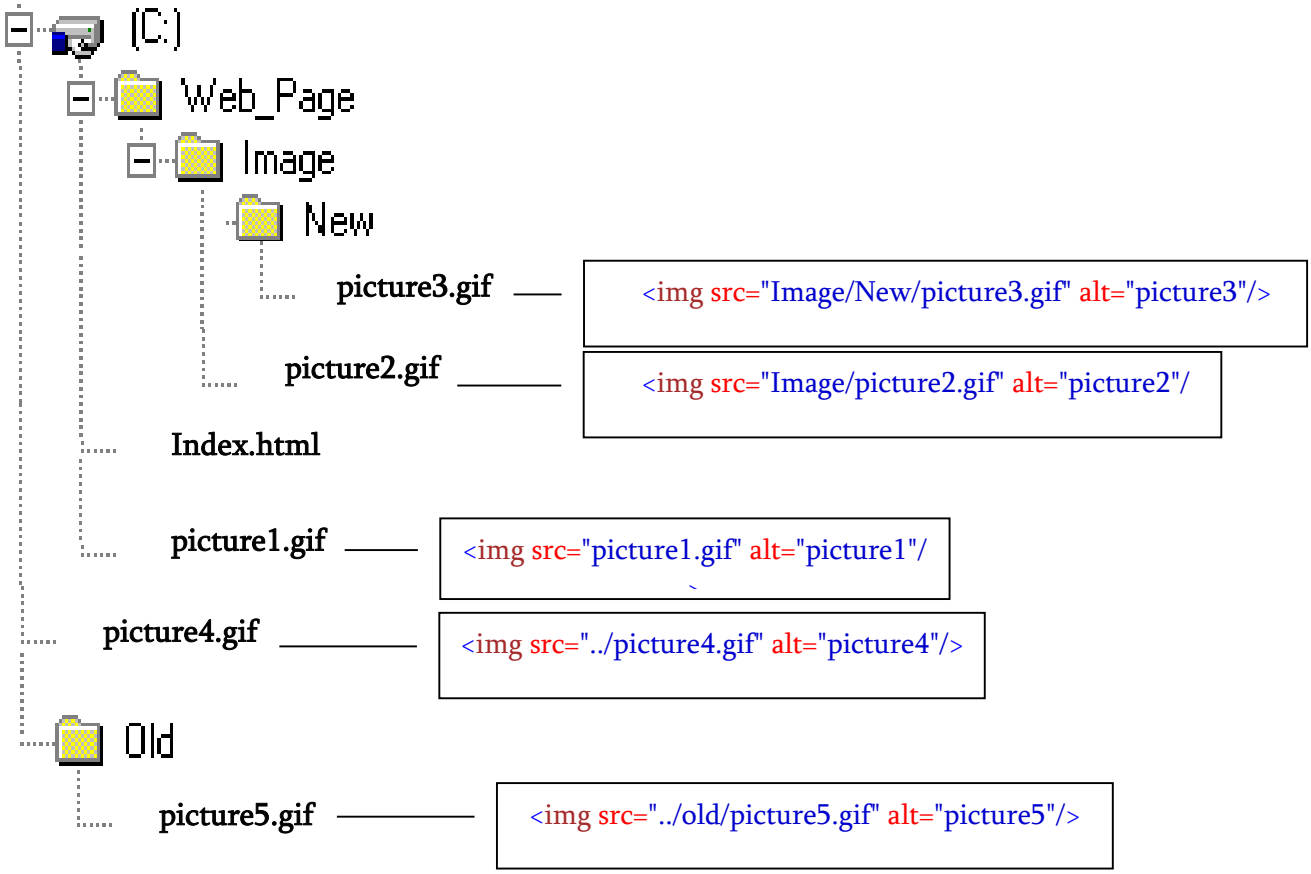
```
<img src="Image/New/picture3.gif" alt="picture3"/>
```

5. html დოკუმენტი მიმართავს საქაღალდის გარეთ არსებულ picture4.gif ფაილს (სურ.1.27), მაშინ მიეთითება **../ფაილის სახელი.**

```
<img src="../picture4.gif" alt="picture4"/>
```

6. html დოკუმენტი მიმართავს საქაღალდის გარეთ არსებულ სხვა Old საქაღალდის picture5.gif ფაილს (სურ.1.27), მაშინ მიეთითება **../საქაღალდის სახელი/ფაილის სახელი.**

```
<img src="../old/picture5.gif" alt="picture5"/>
```




სურ. 1.27 სურათის ასახვა მდებარეობის გათვალისწინებით

ვებ გვერდის ნაწილებად გასაყოფად გამოიყენება ჰორიზონტალური ხაზი <hr />, რომელსაც არ აქვს დამხურავი ტეგი (</hr>არასწორია) და იხურება გამხსნელ ტეგშივე.

### ბმულის ტიპები და პარამეტრები

ბმულები ერთერთი უმთავრესი საკითხის ვებ საიტზე. იგი უამრავ ვებგვერდს აერთიანებს ერთ ვებსაიტად. ვებგვერდები შესაძლებელია სხვადასხვა სერვერზეც კი იყოს განლაგებული, მაგრამ სწორად ორგანიზებული ბმულების საშუალებით მომხმარებელი აღიქვამს ერთ მთლიან საიტად.

არსებობს ორი ტიპის ბმული: **გარე ბმული,** რომელიც ერთმანეთთან აკავშირებს სხვადასხვა ობიექტებს (ვებგვერდები, სურათები, სხვადასხვა ტიპის ფაილები),გვეხმარება მათ შორის ნავიგაციაში და ქმნის ერთიან სივრცეს. **შიდა ბმული,** რომელიც გამოიყენება ერთი ვებგვერდის ფარგლებში ნავიგაციის ოპტიმიზაციისათვის.

html დოკუმენტში ბმულის ასახვა ხდება შემდეგი ტეგით:

```
<a href="მისამართი">ბმულის ტექსტი</a>
```

მაგ.

```
<a href="http://mes.gov.ge/">საქართველოს განათლებისა და მეცნიერების სამინისტრო</a>
<a href="about.html">ჩვენს შესახებ</a>
```

**href** ატრიბუტს ენიჭება იმ ობიექტის მისამართი, რომელზეც ვაპირებთ გადამისამართებას. თუ გადამისამართება ხდება რეალურ საიტზე მაშინ მიეთითება საიტის url-ს. სხვადასხვა ფაილზე გადამისამართების შემთხვევაში ფაილის მისამართს (html დოკუმენტიდან შესაბამის ფაილამდე სრულ გზა). მისამართის მითითების ხერხები დაწვრილებით იხილეთ წინა ქვეთავში - სურათის მისამართების მითითების ვარიანტები (სურ. 1.27).

ბმულზე გადასვლისას შესაბამისი ვებგვერდი ავტომატურად იხსნება იმავე ფანჯარაში. ახალ ფანჯარაში გახსნის შესაძლებლობას იძლევა **target** ატრიბუტი. თუ target ატრიბუტს მივანიჭებთ _blank მნიშვნელობას, მაშინ შესაბამისი ბმულის გახსნა მოხდება ახალ ფანჯარაში.

მაგ. `<a href="http://mes.gov.ge/" target="_blank">განათლებისა და მეცნიერების სამინისტრო</a>`

როცა ვებგვერდზე დიდი ზომის ინფორმაციაა განთავსებული და არ ეტევა მონიტორის ერთ არეში, ნავიგაციის გასაადვილებლად გამოიყენება შიდა ბმულები. ბმულის საშუალებით შეგვიძლია გადავაადგილდეთ იმავე დოკუმენტის სხვადასხვა ადგილას.

შიდა ბმულების ორგანიზების პირველ ეტაპზე უნდა მოვნიშნოთ ადგილი, სადაც ხდება ბმულის გადამისამართება.

მაგ.

```
<h2 id="Chapter1">პარაგრაფი1. .....</h2>
.
<h2 id="Chapter2">პარაგრაფი2. ....</h2>
.
<h2 id="Chapter3">პარაგრაფი3. ....</h2>
```

აღნიშნულ ადგილებზე გადამისამართება ხდება შემდეგი ტეგებით:

```
<ul>
  <li><a href="#Chapter1">პარაგრაფი1.</a></li>
  <li><a href="#Chapter2">პარაგრაფი2.</a></li>
  <li><a href="#Chapter3">პარაგრაფი3.</a></li>
</ul>
```

*პრაქტიკული დავალება*

შექმენით ვიკიპედიიდან თბილისის ისტორიის იდენტური ვებგვერდის სტრუქტურა (სურ. 1.29). შესაბამისი სურათებისა და ბმულების ასახვით.
https://ka.wikipedia.org/wiki/%E1%83%97%E1%83%91%E1%83%98%E1%83%9A%E1%83%98%E1%83%A1%E1%83%98


სურ. 1.28 თბილისის ისტორიის ვებგვერდი

აღნიშნული გვერდი სტილების გამოყენების გარეშე გამოიყურება შემდეგნაირად, რომლის იდენტური ვებგვერდი უნდა შექმნათ html–ის საშუალებით.

**ისტორია**

მთავარი სტატია : თბილისის ისტორია.

თბილისის ხედი XIX საუკუნეში

ლეგენდის თანახმად, თბილისის ტერიტორია ტყით ყოფილა დაფარული, ქართველ მეფეს (ერთ-ერთი ვარიანტით, ვახტანგ I გორგასალს) ნადირობის დროს შველი დაუჭრია, შველი ცხელ წყაროში განბანილა და განკურნებული გაქცევია მონადირეებს (სხვა ვარიანტით, მეფის მიმინო თავს დასცხრომია ხოხობს, ფრინველები ცხელ წყაროში ჩაცვივნულან და გაფუფქულან). ცხელი წყლის სამკურნალო თვისებებისა და ადგილის ხელსაყრელი მდებარეობის გამო მეფეს ტყე გაუკაფავს და ქალაქი გაუშენებია. „თბილისი“ — „თბილი“ (ძვ. ქართულად „ტფილი“) მინერალური წყაროების გამო უწოდეს ქალაქს. შემდგომში ამ ადგილზე გოგირდის აბაზოები გაშენდა. აღნიშნული ადგილი თბილისის ისტორიული უბანი — აბანოთუბანია.

**დაარსება**

აბანოთუბანი — თბილისის უძველესი უბანი

არქეოლოგიური გათხრებით დასტურდება, რომ თბილისის ტერიტორია დასახლებული ყოფილა ჯერ კიდევ ძვ. წ. IV ათასწლეულში. უძველესი წყაროსეული მოხსენიება განეკუთვნება IV საუკუნის II ნახევარს, როცა ამ ადგილებში მეფე ვარაზ-ბაკურის დროს ციხე ააგეს. IV საუკუნის დასასრულს თბილისი სპარსეთის მოხელის — პიტიახშის რეზიდენცია გახდა. V საუკუნის შუა წლებიდან ვვლავ ქართლის მეფეთა ხელში გადავიდა. ვახტან გორგასალმა აღადგინა და გააშენა, ამიტომ იგი მიჩნეულია ქალაქის დამაარსებლად. ზოგიერთი ისტორიკოსის მტკიცებით მეფე ვახტანგ გორგასალი (რომელიც V საუკუნის მეორე ნახევარში მეფობდა) სინამდვილეში ქალაქის აღორძინებასა და აღმშენებლობაშია პასუხისმგებელი, მისი დაფუძნების წაცვლად.

**დედაქალაქი**



თბილისი ფრანგი მოგზაური ჟან შარდენის მიხედვით, 1671

ვახტანგ გორგასლის მემკვიდრემ დაჩი I-მა უფკარმელმა (VI ს. დამდეგი) დაამთავრა ქალაქის ზღუდე-გალავნის აგება, რომელმაც ქალაქის საზღვრები განავრცო და მამის ანდერძის თანახმად, სატახტო ქალაქი მცხეთიდან თბილისში გადმოიტანა.
თბილისის უძველესი მოსახლეობა გაჩნდა გოგირდოვანი წყაროების უბანში (ახლანდელი გორგასლის მოედნის მომიჯნე ტერიტორია), სამხრეთ-აღმოსავლეთით ქალაქი იფარგლებოდა ყოფილი ორთაჭალის ბაღის მიდამოებით, ჩრდილოეთ-აღმოსავლეთით მდინარე მტკვარი საზღვრავდა, სამხრეთ-დასავლეთით — თაბორის ქედის კალთები, ჩრდილოეთ-დასავლეთით კი — წავკისისწყალი. IV საუკუნეშივე წარმოიშვა თბილისის მეორე უბანი კალა ციხითურთ, შემდგომში ქალაქი იზრდებოდა საკუთრივ თბილისისა და კალის ფარგლებს გარეთ, მტკვრის დინების აღმა. ზრდას ხელს უწყობდა მისი ხელსაყრელი გეოგრაფიული მდებარეობა. აქ გადიოდა მნიშვნელოვანი სავაჭრო გზები აღმოსავლეთ ამიერკავკასიისა და წინა აზიისკენ. თბილისი თანდათან შუა აღმოსავლეთის ერთ-ერთი მნიშვნელოვანი ცენტრი გახდა. მომიჯნავე სახელმწიფოთა სამხედრო-სტრატეგიულმა და ეკონომიკურმა ინტერესებმა გაზარდეს ინტერესი მისადმი. VI საუკუნის ბოლოდან იწყება საუკუნოვანი ბრძოლა თბილისისათვის.

სურ. 1.29 თბილისის ისტორიის ვებგვერდის html სტრუქტურა

## ცხრილის ტეგები და მისი პარამეტრები

ცხრილები გამოიყენება პროდუქციის ფასის, ვალუტის კურსის, ამინდის პროგნოზის, სტუდენტთა შეფასებების და სხვა სტატისტიკური მონაცემების ვებ გვერდზე ასახვისათვის.

ა)

| სახელი | გვარი | ასაკი |
|---|---|---|
| დავით | ბარბაქაძე | 6 |
| თამარ | ბარბაქაძე | 4 |

ბ) <table>

| | | | | |
|---|---|---|---|---|
| <tr> | <th>სახელი</th> | <th> გვარი </th> | <th> ასაკი </th> | </tr> |
| <tr> | <td> დავით </td> | <td> ბარბაქაძე </td> | <td> 6 </td> | </tr> |
| <tr> | <td> თამარ </td> | <td> ბარბაქაძე </td> | <td> 4 </td> | </tr> |

</table>

სურ. 1.30 ა) ცხრილის სახე ვებგვერდზე

ბ) ცხრილის ორგანიზება ტეგების საშუალებით

სურ. 1.30 ა–ზე მოცემული ცხრილის ამსახველი html კოდის გამოიყურება შემდეგნაირად.

```
<table width="100%" border="1">
  <tr>
    <th>სახელი</td>
    <th>გვარი</td>
    <th>საკი</td>
  </tr>
  <tr>
    <td>დავით</td>
    <td>ბარბაქაძე</td>
    <td>6</td>
  </tr>
  <tr>
    <td>თამარ</td>
    <td>ბარბაქაძე</td>
    <td>4</td>
  </tr>
</table>
```

ცხრილის ორგანიზებისათვის გამოიყენება შემდეგი ტეგები:

**<table>... </table>**– ცხრილი

**<tr>... </tr>**– სტრიქონი

**<td>... </td>**– უჯრა

**<th>... </th>**– სათაურის სტრიქონის უჯრები

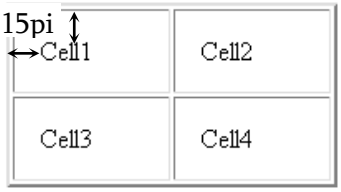
ინფორმაციის (ტექსტური, გრაფიკული და.ა.შ)შეტანა ხორციელდება უჯრებში.

ცხრილის ცარიელი უჯრის ასახვა შესაძლებელია <td> </td>ელემენტით.

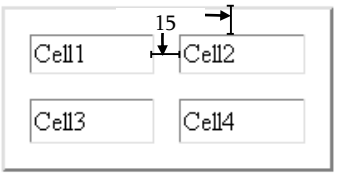
ცხრილის პარამეტრების განსაზღვრა შესაძლებელია შემდეგი ატრიბუტებით:

- **border** – რომლის მნიშვნელობა განსაზღვრავს ცხრილის ჩარჩოების სისქეს პიქსელებში. თუ ატრიბუტი border არ არის განსაზღვრული ცხრილის ჩარჩოები უხილავია.
- **width** – რომლის მნიშვნელობა განსაზღვრავს ცხრილის სიგრძის ზომას. თუ არ არის განსაზღვრული ცხრილის სიგრძე, მაშინ ის დამოკიდებული ხდება უჯრებში ჩაწერილი ინფორმაციის ზომაზე. width ატრიბუტის მნიშვნელობა შეიძლება მიეთითოს ორი ერთეულით: აბსოლუტური – პიქსელებში (width="540") ან ფარდობითი – პროცენტებში (width="70%"). პროცენტულად ცხრილის ზომის განსაზღვრა ხდება ეკრანის ზომის პროპორციულად და სხვადასხვა ეკრანზე მოხდება სხვადასხვა სიგრძის ცხრილის ასახვა.
  width ატრიბუტი შესაძლებელია გამოვიყენოთ უჯრის აღმწერ td ტეგში და განსაზღვრავს შესაბამისი უჯრის სიგრძეს.
- **height** ატრბუტი გამოიყენება სიმაღლის განსასაზღვრად, როგორც ცხრილის (table) ასევე სტრიქონის (tr) ან უჯრის (td) ტეგში. ერთ სტრიქონზე სხვადასხვა სიმაღლის უჯრის შექმნა შეუძლებელია, ამიტომ სიმაღლის განსაზღვრა საკმარისია სტრიქონის ერთერთ უჯრაში.
- **cellspacing** – ცხრილის უჯრებს შორის დაშორება (სურ.1.31. ბ).
- **cellpadding –** ცხრილის უჯრაში არსებულ ინფორმაციასა და მის ჩარჩოებს შორის დაშორება (სურ.1.31. ა).

ა)




ბ)



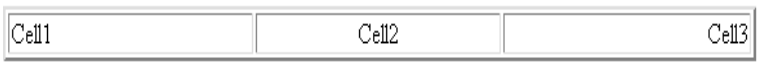

სურ. 1.31 ა) უჯრაში არსებულ ინფორმაციასა და მის ჩარჩოებს შორის დაშორება

ბ) ცხრილის უჯრებს შორის დაშორება

- **align –** უჯრაში არსებული ინფორმაციის ჰორიზონტალური მდებარეობის განსაზღვრა (სურ.1.32. ა). შესაძლო მნიშვნელობებია: left – მარცხნივ გასწორება, center – ცენტრირება, right – მარჯვნივ გასწორება
- **valign –** უჯრაში არსებული ინფორმაციის ვერტიკალური მდებარეობის განსაზღვრა (სურ.1.32. ბ). შესაძლო მნიშვნელობებია: top – ინფორმაციის უჯრის ზედა ნაწილში განთავსება, middle – შუაში და bottom – ქვედა ნაწილში განთავსება.

**align/valign** ატრიბუტების სტრიქონის განმსაზღვრელ ტეგში გამოყენების შემთხვევაში მდებარეობები გავრცელდება შესაბამისი სტრიქონის ყველა უჯრაზე.

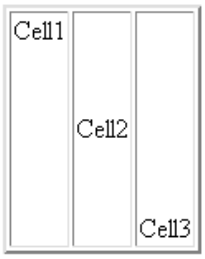
ა)




```
<table width="60%" border="1">
  <tr>
    <td align="left">Cell1</td>
    <td align="center">Cell2</td>
    <td align="right">Cell3</td>
```

```
  </tr>
</table>
```

გ)

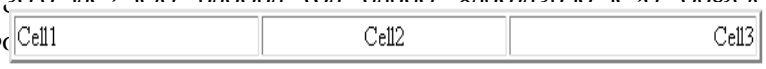



```
<table width="60%" border="1">
  <tr>
    <td align="top">Cell1</td>
    <td align="middle">Cell2</td>
    <td align="bottom">Cell3</td>
  </tr>
</table>
```

სურ. 1.32 უჯრაში ინფორმაციის მდებარეობა

ა) ჰორიზონტალური ბ) ვერტიკალური

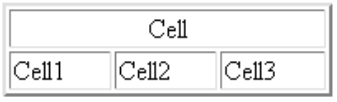
– **bgcolor** – უჯრაში ფერის ჩასხმა. მიეთითება ფერის შესაბამისი ექვსნიშნა კოდი. მაგ. #ffffff - თეთრი ფერი, #000000 - შავი ფერი, #ff0000 – წითელი და.ა.შ.

შენიშვნა: ცხრილის ეიზუალური სახისა და მასში განთავსებული ინფორმაციის მდებარეობის განსაზღვრა მართებულ  შვეობით არამედ css კასკადური სტილების ენის გამოყენებით. css თვისებები და მათი გამოყენების შესაძლებლობები დაწვრილებით აღწერილია შემდეგ თავში.



ცხრილების ორგანიზებისას ხშირად დგება უჯრების გაერთიანების საჭიროება, რომლის შესაძლებლობასაც იძლევა ატრიბუტები: colspan – ჰორიზონტალურად უჯრების გაერთიანება და rowspan – ვერტიკალურად უჯრების გაერთიანება. მათზე მინიჭებული მნიშვნელობა განსაზღვრავს გასაერთიანებელი უჯრების რაოდენობას.

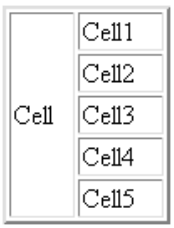
ა)




```
<table width="60%" border="1">
  <tr>
    <td colspan="3">Cell</td>
  </tr>
    <tr>
    <td>Cell1</td>
    <td>Cell2</td>
    <td>Cell3</td>
  </tr>
</table>
```

---

გ)




```
<table width="30%" border="1">
  <tr>
    <td rowspan="5">Cell</td>
        <td>Cell1</td>
  </tr>
```

```
<tr><td>Cell2</td> </tr>
<tr><td>Cell3</td> </tr>
<tr><td>Cell4</td> </tr>
<tr><td>Cell5</td> </tr>
</table>
```

სურ. 1.33 უჯრების გაერთიანებაა) ჰორიზონტალურად ბ) ვერტიკალურად

სურ. 1.34–ზე მოცემულია ვალუტის კურსის ცხრილის ნიმუში და მისი ბრაუზერში გამოსახვისათვის საჭირო html დოკუმენტი.

ვებგვერდი

| საქართველოს ბანკი BANK OF GEORGIA | ვალუტის კურსი | | 1 ლარი = | |
|---|---|---|---|---|
| | ყიდვა | გაყიდვა | ყიდვა | გაყიდვა |
| აშშ დოლარი | 2.375 | 2.455 | 0.421 | 0.407 |
| ევრო | 2.575 | 2.695 | 0.388 | 0.371 |
| გირვანქა სტერლინგი | 3.43 | 3.59 | 0.292 | 0.279 |

html დოკუმენტი

```html
<!DOCTYPE html>
<html>
<head>
<title>ვალუტის კურსი</title>
<meta charset="UTF-8">
</head>
<body>
<table width="80%" border="1">
<tr>
    <td rowspan="2" width="220"><img src="img/bog.jpg" alt="საქართველოს ბანკი"></td>
    <td colspan="2" bgcolor="#dddddd" align="center">ვალუტის კურსი</td>
    <td colspan="2" bgcolor="#dddddd" align="center">1 ღარი =</td>
</tr>
<tr bgcolor="#dddddd" align="center">
    <td>ყიდვა</td>
    <td>გაყიდვა</td>
    <td>ყიდვა</td>
    <td>გაყიდვა</td>
</tr>
<tr>
    <td><img src="img/usa.png" alt="">აშშ დოლარი</td>
    <td>2.375</td>
    <td>2.455</td>
    <td>0.421</td>
    <td>0.407</td>
</tr>
<tr>
    <td><img src="img/eur.png" alt="">ევრო</td>
    <td>2.575</td>
    <td>2.695</td>
    <td>0.388</td>
    <td>0.371</td>
</tr>
<tr>
    <td><img src="img/gbp.png" alt="">გირვანქა სტერლინგი</td>
    <td>3.43</td>
    <td>3.59</td>
    <td>0.292</td>
    <td>0.279</td>
</tr>
</table>

</body>
</html>
```

სურ. 1.34 ვალუტის კურსის ცხრილი

### *პრაქტიკული დავალება*

შექმენით  სურ. 1.35–ზე მოცემული ცხრილის ბრაუზერში წარმოსახვისათვის საჭირო html დოკუმენტი.

**თბილისის საერთაშორისო აეროპორტი**

| საერთაშორისო გამგზავრება | | | |
|---|---|---|---|
| თარიღი | დრო | ავიაკომპანია | მიმართულება |
| 09.01.2016 | 4:25 PM | Airzena Georgian Airlines | როსტოვი |
| 09.01.2016 | 5:10 PM | Ukraine International | კიევი |
| 09.01.2016 | 5:25 PM | S7 Airlines | სტამბული |
| 09.01.2016 | 6:15 PM | Turkish Airlines | |
| **ადგილობრივი გამგზავრება** | | | |
| თარიღი | დრო | ავიაკომპანია | მიმართულება |
| 10.01.2016 | 4:25 PM | Airzena Georgian Airlines | ბათუმი |

სურ. 1.35 ცხრილის პრაქტიკული ნიმუში

## 5.3. ვებგვერდზე ფორმების ასახვა

სწავლის შედეგის შესაბამისი თემატიკა

- ფორმის ელემენტები და მათი ტიპები
- ვებგვერდზე ფორმის ველების ასახვის ტეგები და მათი შესაბამისი პარამეტრები

ვებსაიტზე მომხმარებელთა ინტერაქტიულობის ორგანიზებისათვის გამოიყენება ფორმები. იგი იძლევა საშუალებას ვიზიტორისგან მივიღოთ სხვადასხვა სახის ინფორმაცია და დავუბრუნოთ პასუხი.

ფორმების ვებგვერდზე ასახვა ხდება html-ის საშუალებით, მაგრამ ინფორმაციის მიღება, დამუშავება, დაბრუნებისათვის გამოიყენება ვებ პროგრამირების ენები (PHP, Perl).

ამ თავში განვიხილავთ html-ის საშუალებით ვებგვერდზე სხვადასხვა ტიპის ფორმების ველების ასახვისა და მათი პარამეტრების გაწერის შესაძლებლობებს.

### ფორმის ელემენტები და მათი ტიპები

ნებისმიერი ფორმის საბაზო ტეგია <form>, რომელიც მოითხოვს დამხურავ ტეგს </form>. აღნიშნული ტეგის ატრიბუტები განსაზღვრავენ ფორმის დამუშავებისათვის აუცილებელ ინფორმაციას (სურ. 1.36).

```
<form action="action_page.php" method="GET" accept-charset="UTF-8" enctype="application/x-www-form
    urlencoded"name="test"target="_blank">
    .
    ფორმის ელემენტები
    .
</form>
```

სურ. 1.36. ფორმის საბაზო ტეგი

- **action** ატრიბუტის მნიშვნელობად მიეთითება ფაილის მისამართი/სახელი (მაგ. *.php ან *.asp), რომელშიც პროგრამულადაა აღწერილი თუ რა მოქმედებები უნდა შესრულდეს ფორმიდან მიღებულ მონაცემებზე. ამ ფაილს ქმნის ვებპროგრამისტი.
- **Method** ატრიბუტი უთითებს, თუ როგორ უნდა მოხდეს მონაცემების გადაცემა და მისი მნიშვნელობა დამოკიდებულია გადასაცემი მონაცემის ზომასა და მნიშვნელობაზე.
  **GET** ღია მეთოდის გამოყენებით მონაცემების გადაგზავნა ხდება სამისამართო ველიდან URL-ის საშუალებით. ამ მეთოდით შესაძლებელია მხოლოდ 255 სიმბოლოს გადაგზავნა. ეს მეთოდი გამოიყენება ძებნის დროს და ამ დროს თუ დააკვირდებით სამისამართო ველს შენიშნავთ თქვენს მიერ შეყვანილ ინფორმაციას. იგი გამოიყენება ვებგვერდის გამოძახების/ ძებნის შემთხვევაში.
  **POST** დახურული მეთოდი გამოიყენება დიდი ზომის ინფორმაციის სერვერზე გასაგზავნად. ასე გადაგზავნილი მონაცემები არ ჩანს სამისამართო ველში, რადგანაც ისინი მოთავსებულია შეტყობინების ტანში - body.
- **accept-charset** ატრიბუტით განისაზღვრება ფორმის ველების კოდირება. თუ ეს ატრიბუტი არ არის განსაზღვრული ავტომატურად გამოიყენება ვებგვერდისათვის განსაზღვრული კოდირება.
- **enctype** ატრიბუტის მნიშვნელობა განსაზღვრავს სერვერზე გადაგზავნილი ინფორმაციის კოდირების მეთოდს. სერვერისათვის მონაცემების უსაფრთხოდ გადაცემის მიზნით, ბროუზერი ახდენს მათ კოდირებას სპეციალური მეთოდით. სტანდარტულ კოდირებას წარმოადგენს - application/x-www-form-urlencoded. თუ ფორმა შეიცავს ფაილის არჩევა/მიბმის ტიპის ველს, მაშინ

გამოიყენება კოდირების მეთოდი - multipart/form-data; თუ ფორმის მონაცემების გადაეცემა ხდება ფოსტის საშუალებით, მაშინ გამოიყენება - text/plain (ამ შემთხვევაში action ატრიბუტში მითითება mailto:საფოსტო მისამართი).

– **name** ატრიბუტით ფორმა მენიჭება უნიკალური სახელი, რომელის საშუალებითაც ხდება ფორმის ელემენტებთან წვდომა. ამავე დანიშნულებით შესაძლებელია გამოვიყენოთ ატრიბუტი **id.**

Form ტეგის ზემოთ განხილული ყველა ატრიბუტი ემსახურებოდა ფორმის მონაცემების დამუშავებას და განკუთვნილია ვებპროგრამისტისთვის.

– **tarjet** ატრიბუტით განისაზღვრება რომელ ფანჯარაში გაიხსნას სერვერიდან დაბრუნებული ინფორმაცია. target="_blank" ინფორმაცია გაიხსნება ახალ ფანჯარაში, target="_self" - იმავე ფანჯარაში. თუ ატრიბუტი tarjet არ არის მითითებული, მაშინ მიღებული ინფორმაცია ავტომატურად გაიხსნება იმავე ფანჯარაში.

## ვებგვერდზე ფორმის ველების ასახვის ტეგები და მათი შესაბამისი პარამეტრები

ფორმის ყველაზე მნიშვნელოვანი ელემენტია <input type="ველის ტიპი" name="ველის სახელი">

**name** ატრიბუტის მნიშვნელობა ველს ანიჭებს უნიკალურ სახელს. მომხმარებლის მიერ შესაბამის ველში შეყვანილი ინფორმაცია ველის სახელით, ცვლადის სახით გადაეცემა შესაბამის ფუნქციას/ფაილს სადაც ხდება მისი დამუშავება. ამ ატრიბუტის მნიშვნელობას იყენებს ვებპროგრამისტი და აუცილებელია გათვალისწინებული იქნას კოდის წერის ეთიკა და ლოგიკურად იქნას შერჩეული.

**type** ატრიბუტის მნიშვნელობა განსაზღვრავს ფორმის ველის ტიპს.

ცხრილი 1. 2 ველის ტიპები

| type ატრიბუტის მნიშვნელობა | ველის ტიპები |
|---|---|
| text | ტექსტის შესაყვანი ველი |
| password | პაროლის შესაყვანი ველი |
| submit | ღილაკი, ფორმის მონაცემების გასაგზავნად |
| button | ღილაკი |
| reset | ღილაკი ფორმის გასუფთავებისათვის |
| image | სურათის ველი |
| radio | გადამრთველი |
| checkbox | ალმით მოსანიშნი |
| File | ფაილის ასატვირთი ველი |

### ტექსტის/პაროლის შესაყვანი ველი და ღილაკის ტიპები

სურ. 1.36-ზე მაგალითისათვის მოცემულია ავტორიზაციის ფორმის ნიმუში, რომელიც შედგება სამი ტიპის ველისაგან:

1. <input type="text" name="username"> - ტექსტის შესაყვანი ველი
2. <input type="password" name="psw"> - პაროლის შესაყვანი ველი. ველის თავისებურებიდან გამომდინარე მასში შეყვანილი სიმბოლო დაფარულია (სისტემიდან გამომდინარე გამოისახება მუქი წერტილებით ან ფიფქებით) და პაროლის შეყვანისას შეუძლებელია მისი დანახვა.
3. <input type="submit" value="Submit"> - ღილაკი, რომელიც ფორმაში არსებულ მონაცემებს გადაუგზავნის სერვერზე არსებულ დამმუშავებელ ფაილს (action.php).

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<form action="action.php" method="GET" enctype="application/x-www-form urlencoded"name="login">
  მომხმარებელი:<br />
  <input type="text" name="username"><br />
  პაროლი:<br />
  <input type="password" name="psw"><br />
  <input type="submit" value="Submit">
</form>
```

მომხმარებელი:
eka.chikashua
პაროლი:
••••••••
Submit

სურ. 1.36 ავტორიზაციის ფორმის ნიმუში1

ტექსტური/პაროლის შესაყვანი ველის ტეგში შეგვიძლია გამოვიყენოთ ორი ატრიბუტი:

- **Size** რომლის მნიშვნელობაც განსაზღვრავს ჩარჩოს სიგრძეს (ავტომატურად size="20"). ტექსტური ველის სიგრძე განისაზღვრება ერთმანეთის გვერდზე მდგომი მონო სიგანის (monospace font-ერთნაირი სიგანის მქონე სიმბოლოებიანი შრიფტი) შრიფტის მქონე სიმბოლოების რაოდენობით. თუ შრიფტის ზომა იცვლება სტილების საშუალებით თავისთავად შესაბამისად იცვლება ველის სიგრძეც. size ატრიბუტი განსაზღვრავს ველის სიგრძეს (სურ. 1.37) და არა მასში შესატანი სიმბოლოების რაოდენობას. თუ მონაცემთა შესატან ველში შეტანილი სიმბოლოების რაოდენობა ველის სიგრძეს აღემატება, მაშინ პირველად აკრეფილი ტექსტის გადაფარვა ხდება.
- **Maxlenght** - განსაზღვრავს ველში შესაყვანი სიმბოლოების დასაშვებ მაქსიმალურ რაოდენობას. მითითებულ მნიშვნელობაზე მეტი სიმბოლოს შეტანის მცდელობის დროს ბრაუზერი ხმოვან სიგნალს გამოსცემს და ზედმეტი სიმბოლოს შეტანის საშუალებას არ იძლევა. თუ მოცემული ატრიბუტი არ იქნება მითითებული, მაშინ მისი მნიშვნელობა არ არის განსაზღვრული.

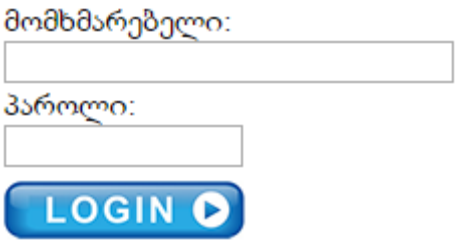
ღილაკის შექმნისას უნდა გავითვალისწინოთ მისი დანიშნულება/ფუნქცია:

- type="button" - ჩვეულებრივი ღილაკი
- type="submit" - ღილაკი, რომელიც ფორმაში არსებულ მონაცემებს გადაუგზავნის სერვერზე არსებულ დამმუშავებელ ფაილს
- type="reset" - ღილაკი, რომლის საშუალებითაც ხდება ფორმის მონაცემების გასუფთავება და სტანდარტულ მნიშვნელობებზე დაბრუნება

ღილაკის შექმნისას **value** ატრიბუტზე მინიჭებული მნიშვნელობა აისახება ტექსტის სახით ღილაკზე (სურ. 1.36). შესაძლებელია ღილაკზე სურათის გამოსახვა. ამ შემთხვევაში type="image" და აუცილებელია ატრიბუტში src მიეთითოს სურათის შესაბამისი ფაილის მისამართი (სურ. 1.37). სასურველია alt ატრიბუტია გამოყენებაც, რათა სურათის არ ჩატვირთვის შემთხვევაში მომხმარებელი მიხვდეს ღილაკის დანიშნულებას.

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<form action="action.php" method="GET" enctype="application/x-www-form urlencoded"name="login">
  მომხმარებელი:<br />
  <input type="text" name="username" size="30"><br />
  პაროლი:<br />
  <input type="password" name="psw" size="15"><br />
  <input type="image" src="login.png" alt="LogIn">
</form>
```




სურ. 1.37 ავტორიზაციის ფორმის ნიმუში2

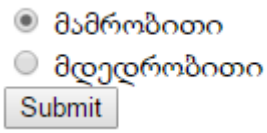
### გადამრთველები/რადიო ღილაკები

გადამრთველი/რადიო ღილაკების ტიპის ველის (type="radio") თავისებურება იმაში მდგომარეობს, რომ ის მომხმარებელს შემოთავაზებული რამდენიმე ვარიანტებიდან მხოლოდ ერთი მონაცემის არჩევის საშუალებას აძლევს (სურ. 1.38). გადამრთველების შემთხვევაში ველის სახელი name="gender" ერთი და იგივე უნდა იყოს, რადგან ისინი ფაქტიურად ერთ ელემენტს წარმოადგენენ, ხოლო მნიშვნელობა ენიჭებათ სხვადასხვა value="male" და value="gender" და სერვერზე გადაიგზავნება მომხმარებლის მიერ არჩეული მნიშვნელობა.

Checked ატრიბუტი განსაზღვრავს რომელი მნიშვნელობა იყოს ავტომატურად მონიშნული. იგი მოსახერხებელია, ისეთი პასუხების მოსანიშნად, რომელსაც ხშირად ირჩევენ მომხმარებლები. ამ ატრიბუტის მითითების გარეშე არცერთი მნიშვნელობა არ იქნება ავტომატურად მონიშნული.

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<form action="action.php" method="GET" enctype="application/x-www-form urlencoded"name="genderForms">
  <input type="radio" name="gender" value="male" checked>
  მამრობითი<br />
  <input type="radio" name="gender" value="female">
მდედრობითი <br />
  <input type="submit" value="Submit">
</form>
```




სურ. 1.38 გადამრთველი ველის ნიმუში

გადამრთველი/რადიო ღილაკის სურ. 1.38-ზე განხილული ნიმუშის მიხედვით შექმნისას, მომხმარებელმა სასურველი ვერსიის მოსანიშნად მაუსის მაჩვენებელი აუცილებლად უნდა დააჭიროს ფორმის აღმნიშვნელ წრეს/ღილაკს (შესაბამის ტექსტზე დაჭერით ველი არ ინიშნება).
ფორმასთან მომხმარებლის მუშაობის გასაადვილებლად გამოიყენება label ელემენტი, რომელიც ფორმის ვიზუალურ მხარეს არ ცვლის, მაგრამ ამ შემთხვევაში სასურველი ვერსიის მონიშვნა შესაძლებელია შესაბამის ტექსტზე მაუსის მაჩვენებლის დაჭერითაც.
input ელემენტში, რომელთანაც უნდა დაკავშირდეს label ელემენტში აღწერილი ტექსტი ვამატებთ id ატრიბუტს, რომლის მნიშვნელობა label ელემენტში მითითებული for ატრიბუტის მნიშვნელობის იდენტური უნდა იყოს.

html დოკუმენტი

```
<form action="action.php" method="GET" enctype="application/x-www-form
urlencoded"name="genderForms">
  <input type="radio" name="gender" id="male" value="male">
<label for="male">მამრობითი</label><br/>
  <input type="radio" name="gender" id="female" value="female">
<label for="female">მდედრობითი </label><br/>
  <input type="submit" value="Submit">
</form>
```

სურ. 1.39 გადამრთველი ველის ნიმუში label ელემენტის გამოყენებით

მომხმარებლის ფორმის აღქმის გასამარტივებლად, შესაძლებელია ფორმის ველების დაჯგუფება fieldset ელემენტის საშუალებით. fieldset ელემენტი აჯგუფებს ფორმაში განთავსებულ ველებს და შემოსაზღვრავს მათ ჩარჩოთი. ჯგუფის სათაურის განსაზღვრა კი შესაძლებელია legend ელემენტით (სურ. 1.40).

html დოკუმენტი

```
<form action="action.php" method="GET" enctype="application/x-www-form
urlencoded"name="genderForms">

  <fieldset>
  <legend>სქესი:</legend>

<input type="radio" name="gender" id="male" value="male">
<label for="male">მამრობითი</label><br/>
  <input type="radio" name="gender" id="female" value="female">
<label for="female">მდედრობითი </label><br/>
  <input type="submit" value="Submit">

</fieldset>
</form>
```

ვებგვერდი


სურ. 1.40 გადამრთველის ველების დაჯგუფება

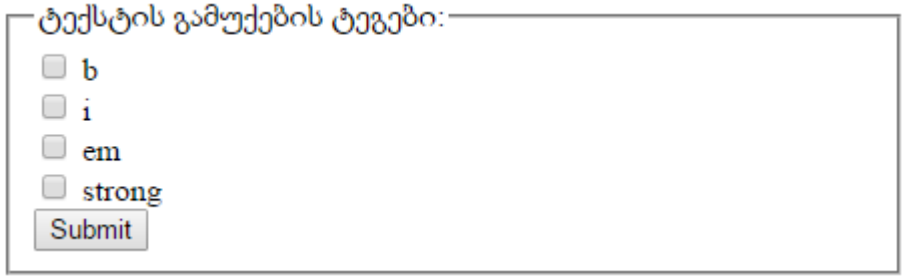
### ალმით მოსანიშნი

ალმით მოსანიშნი ველი (type="checkbox") მომხმარებელს ასევე სთავაზობს არჩევანის გაკეთების საშუალებას, მაგრამ ზემოთ განხილული გადამრთველებისაგან განსხვავებით, ამ შემთხვევაში მომხმარებელს შეუძლია რამდენიმე ვარიანტის ერთდროულად მონიშვნა. გადამრთველების მსგავსად, ალმით მოსანიშნი ფორმის შემთხვევაშიც, ველის სახელი name=" fontWeight" ერთი და იგივე უნდა იყოს, რადგან ისინი ფაქტიურად ერთ ელემენტს წარმოადგენენ, ხოლო მნიშვნელობა ენიჭებათ სხვადასხვა value="b", value="i", value="em" და name="strong" და სერვერზე გადაიგზავნება მომხმარებლის მიერ არჩეული მნიშვნელობა.

html დოკუმენტი

```
<form action="action.php" method="GET" enctype="application/x-www-form
urlencoded"name="fontWeight">
  <fieldset>
  <legend>ტექსტის გამუქების ტეგები:</legend>
<input type="checkbox" name="fontWeight" id="b" value="b">
<label for="b">b</label><br/>
  <input type="checkbox" name="fontWeight" id="i" value="i">
<label for="i">i</label><br/>
  <input type="checkbox" name="fontWeight" id="em" value="em">
<label for="em">em</label><br/>
  <input type="checkbox" name="fontWeight" id="strong" value="strong">
<label for="strong">strong</label><br/>
  <input type="submit" value="Submit">
</fieldset>
</form>
```

ვებგვერდი



სურ. 1.41 გადამრთველის ველების დაჯგუფება

**!შენიშვნა**: ფორმის ველებში, რომლიდანაც შეიძლება რამდენიმე მნიშვნელობის სერვერზე ერთდროული გადაგზავნა ატრიბუტის name მნიშვნელობად გამოიყენება მასივები.

### ფაილის ასატვირთი ველი

ხშირად იქმნება ფაილის სერვერზე ატვირთვის საჭიროება. მომხმარებლებმა შეიძლება ატვირთონ სხვადასხვა ტიპის ფაილები, მაგ. გრაფიკული, ვიდეო ან აუდიო ფაილები.

ფაილის ასატვირთი ველის შესაქმნელად ატრიბუტს type ენიჭება მნიშვნელობა file, ხოლო accept ატრიბუტის მნიშვნელობით განისაზღვრება ასატვირთი ფაილის ტიპი.

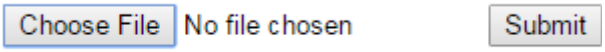
```
<input type="file" name="picture" accept="file_extension">
<input type="file" name="picture" accept="image/*">
<input type="file" name="picture" accept="audio/*">
<input type="file" name="picture" accept="video/*">
<input type="file" name="picture" accept="media_type">
```

ფაილის ასატვირთი ველის უჩნდება ღილაკი Choose File, რომელზე დაჭერაც იძლევა ასატვირთი ფაილის შერჩევის საშუალებას.

html დოკუმენტი

```
<form action="action.php" method="GET" enctype="applicati
on/x-www-form urlencoded"name="fileUpload">
<input type="file" name="picture" accept="image/*">
 <input type="submit" value="Submit">
</form>
```

ვებგვერდი




სურ. 1.42 ფაილის ასატვირთი ველის ნიმუში

### ჩამოშლადი ჩამონათვლის ველები

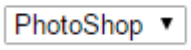
მომხმარებლისათვის შეთავაზებული მნიშვნელობიდან სასურველის არჩევის საშუალებას იძლევა ჩამოშლადი ველები. გადამრთველებისა და ალმით მოსანიშნი ველისაგან განსხვავებით ჩამოშლადი ველი საშუალებას იძლევა კომპაქტურად განვალაგოთ მასში შემავალი პუნქტები და წინასწარ განსაზღვროთ ველის რამდენი პუნქტი იყოს მომხმარებლისათვის ხილული.

ჩამოშლადი ველის შესაქმნელად გამოიყენება select ელემენტი, რომელშიც ჩამონათვალის პუნქტები აღიწერება option ტეგის საშუალებით. option ელემენტის value ატრიბუტს ენიჭება მნიშვნელობა, რომელიც გადაეცემა სერვერს შესაბამისი პუნქტის არჩევის შემთხვევაში. selected ატრიბუტი განსაზღვრავს ავტომატურად მონიშნულ პუნქტს (სურ. 1.43).

html დოკუმენტი

ვებგვერდი

```
<form action="action.php" method="GET" enctype="application/x-
www-form urlencoded"name="Programs">
<select>
  <option value="Word">Word</option>
  <option value="Excel">Ecxel</option>
  <option value="Photoshop" selected>Photoshop</option>
  <option value="Illustrator">Illustrator</option>
</select>
</form>
```




სურ. 1.43 ჩამოშლადი ველის ნიმუში

როცა არცერთ option ელემენტში არ არის მითითებული selected ატრიბუტი, ავტომატურად პირველი პუნქტი ჩანს ფორმაში, თუ გვსურს თავდაპირველად ცარიელი იყოს ფორმა შესაძლებელია პირველ პუნქტად დაემატოს ცარიელი Option ელემენტი <option value=" "> </option>
ჩამოშლადი ველის პუნქტების სისტემატიზება, რათა მომხმარებელს გაუადვილდეს მუშაობა, შესაძლებელია optgroup ელემენტის საშუალებით, რომელიც label ატრიბუტის საშუალებით ჩამონათვალის პუნქტებს უქმნის სათაურებს (სურ. 1.44).

| html დოკუმენტი | ვებგვერდი |
|---|---|

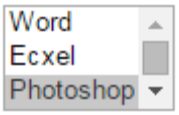
```
<form action="action.php" method="GET" enctype="application/x-
www-form urlencoded"name="Programs">
<select>
<optgroup label="Microsoft office">
  <option value="Word">Word</option>
  <option value="Excel">Ecxel</option>
</optgroup>
<optgroup label="Adobe">
  <option value="Photoshop" selected>Photoshop</option>
  <option value="Illustrator">Illustrator</option>
</optgroup>
</select>
</form>
```


სურ. 1.44 ჩამოშლადი ველის პუნქტების სისტემატიზირების ნიმუში

ზემოთგანხილული ჩამოშლადი ველის მაგალითებში მომხმარებლისათვის ველის ჩამოშლამდე ხილული არის მხოლოდ ერთი პუნქტი. თუმცა size ატრიბუტის საშუალებით შესაძლებელია განისაზღვრო ხილვადი ელემენტების რაოდენობა. თუ size ატრიბუტის მნიშვნელობა ჩამოშლადი ველის პუნქტებზე ნაკლებია ავტომატურად ჩნდება ვერტიკალური გადაფურცვლის ბილიკი (სურ.1.45).

| html დოკუმენტი | ვებგვერდი |
|---|---|

```
<form action="action.php" method="GET" enctype="application/x-
www-form urlencoded"name="Programs">
<select value="3">
  <option value="Word">Word</option>
  <option value="Excel">Ecxel</option>
  <option value="Photoshop" selected>Photoshop</option>
  <option value="Illustrator">Illustrator</option>
</select>
</form>
```

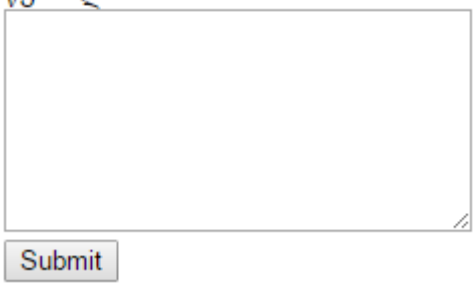



სურ. 1.45 ჩამოშლადი ველის ნიმუში, როცა სამი პუნქტია ხილული

შენიშვნა: select ელემენტში multiple ატრიბუტის დამატება <select multiple> იძლევა რამოდენიმე პუნქტის ერთდროულად მონიშვნის საშუალებას (ctrl კლავიშის დახმარებით).

### ტექსტური არე

დიდი ზომის ტექსტების შესაქმნელად გამოიყენება ელემენტი textarea. ველის ზომა დამოკიდებულია ატრიბიტებზე cols - სიგრძე  და rows - სიმაღლე.

| html დოკუმენტი | ვებგვერდი |
| --- | --- |
| <form action="action.php" method="GET" enctype="application/x-www-form urlencoded"name="mail"><br>  წერილი:<br /><br>  <textarea cols="30" rows="10">  </textarea>  <br /><br>  <input type="submit" value="Submit"><br></form> | წერილი:<br>  <br>Submit |

სურ. 1.46 ტექსტური არის ფორმის ნიმუში

# 6. საიტის სტილებით გაფორმება - css

## 6.1. ვებსაიტის საბაზო ელემენტების გაფორმება სტილებით

**მიმდინარე პარაგრაფის თემატიკა**

- სტილების მნიშვნელობა და გამოყენების სფერო
- დოკუმენტის სტილებით გაფორმების მეთოდები
- css–ის კოდის წერის სინტაქსი
- css კოდის წერის ეთიკა
- ძირითადი სელექტორები
- სელექტორების თვისებისა და მნიშვნელობის არსი
- Class და ID სელექტორების მნიშვნელობა და გამოყენების ასპექტები

### სტილების მნიშვნელობა და გამოყენების სფერო

ვინაიდან HTML -ის ენა მოკლებულია იმ შესაძლებლობებს, რომ გავაფორმოთ დოკუმენტის ვიზუალური მხარე ისე როგორც ეს თანამედროვეობას შეესაბამება, ამისათვის WEB-გვერდების ვიზუალიზაციაში აუცილებლობა წარმოიშვა შეექმნათ დამატებითი ენა რომელსაც ეწოდა CSS(Cascading Style Sheet) ანუ კასკადური სტილების ენა. CSS კასკადური სტილების ენა საშუალებას გვაძლევს დოკუმენტის ვიზუალიზაცია მომხმარებლისთვის იყოს კომფორტული, ადვილად აღსაქმელი და რაც ყველაზე მთავარია დოკუმენტი ყველა მოწყობილობაზე აისახებოდეს ჩვენთვის სასურველი ფორმით. CSS ენის გამოჩენამ WEB ტექნოლოგიების სივრცეში, მისცა დეველოპერებს საშუალება შეექმნათ უფრო მიმზიდველი და ეფექტური საიტები.

აღნიშნული ენის დანერგვამ მკვეთრად გამიჯნა ერთმანეთისგან HTML დოკუმენტის სტრუქტურული ნაწილი მისი ვიზუალური ნაწილისგან. როდესაც ამ ენის შექმნამდე ვიყენებდით ათასგვარ ხრიკს, რომ დოკუმენტი ვიზუალურად დაგვეფორმატებინა ჩვენი სურვილის და მიხედვით, ახლანდელ დროში ეს პრობლემა აღმოიფხვრა CSS ენის წყალობით. ანუ კიდევ ერთხელ აღვნიშნოთ, რომ HTML კოდს ვიყენებთ რეალურად დოკუმენტის სტრუქტურის შექმნაში და CSS კოდს ვიყენებთ ცალსახად მხოლოდ მის ვიზუალურ ფორმირებაში და მიმზიდველი ეფექტების შექმნისთვის. აქვე ერთ გარემოებას გავუსვათ ხაზი, CSS ენა არავითარ შემთხვევაში არ წარმოადგენს HTML ენის შემცვლელ ალტერნატიულ ტექნოლოგიას. როგორც ზემოთ აღვნიშნეთ ეს ორი კოდირების ენა ერთმანეთს მკვეთრად ემიჯნება თავისი დანიშნულების მიხედვით.

ახლა მოდით აღვწეროთ თუ რის გამო არის CSS-ი HTML კოდირების ენისგან განსხვავებული დოკუმენტის ვიზუალიზაციის ნაწილში. როგორც წესი მოგეხსენებათ HTML ენაში დოკუმენტის სტრუქტურა იქმნება ტეგების საშუალებით და თითოეული დოკუმენტის ელემენტს, იქნება ეს გრაფიკული გამოსახულება თუ ტექსტური ელემენტი, გარკვეული ატრიბუტის (პარამეტრის) მეშვეობით ენიჭება დაშვებული მნიშვნელობები, როგორიცაა სიგანე, სიმაღლე, ტექსტის ფერი და ასე შემდეგ. რაც შეეხება CSS-ის კოდირების ენას მთელი მისი შექმნის შინაარსი გამოიხატება იმაში, რომ ერთი

მითითებების ნაკრები შეგვიძლია გამოვიყენოთ კასკადურად ერთი ჯგუფის ქვეშ გაერთიანებული ტეგებისთვის. თუ რა აერთიანებს ამ თაგებს ერთი ჯგუფის ქვეშ ამაზე ქვემოთ დაწვრილებით ვისაუბრებთ.

ქვემოთ მოყვანილ 3.1. ცხრილში გთავაზობთ გაეცნოთ იმ განსხვავებებს რაც ანსხვავებს ერთმანეთისგან CSS ენის ვერსიებს, თუმცა აქვე დავაზუსტოთ, რომ ვერსიებს შორის მხოლოდ ატრიბუტიკის ნაწილშია სხვაობა და არავითარი პრინციპული ან ლოგიკური სხვაობა არ არსებობს.

მინიშნება: თუ თქვენ დაეუფლებით CSS1 -ის გამოყენების პრინციპებს, არ იფიქროთ, რომ ამ ენის მომდევნო ვერსიების შესასწავლად გარკვეული მნიშვნელოვანი ძალისხმევა დაგჭირდებათ

ცხრილი 3.1 CSS ენის ვერსიების შესაძლებლობები

| ვერსია | მიღების თარიღი | შესაძლებლობები |
|---|---|---|
| CSS1 | 01.1996 | • ფურცელზე ელემენტის ასახვის წესის მართვა<br>• ელემენტის და ტექსტის ურთიერთმიმართება<br>• ელემენტის ზომების მართვა<br>• შიდა და გარე დაძვრების მართვა<br>• ცხრილებში ვერტიკალზე განლაგების მართვა<br>• ელემენტის საზღვრების სტილის მართვა<br>• სიების დაფორმატების მართვა<br>• ტექსტის ფერის და ფონის მართვა<br>• შრიფტის პარამეტრების მართვა<br>• ტექსტის თვისებების მართვა<br>• სტრიქონებს შორის ინტერვალების მართვა |
| CSS2 | 05.1998 | CSS1-ის შესაძლებლობანი და დამატებითი სიახლეები:<br>• ტექსტის მიმართულების მართვა<br>• ელემენტის პოზიციონირების მართვა<br>• უბნების ხილვადობის მართვა<br>• საზღვრებიდან გასული ელემენტების მართვა<br>• კურსორის გარეგნული სახის მართვა<br>• ფენების მართვა<br>• ელემენტის ზღვრული ზომების მართვა<br>• ცხრილის უჯრედებს შორის დაცილებები<br>• ელემენტის ცალკეული საზღვრების მართვა<br>• ცხრილის ელემენტების ზომების მართვა<br>• ბრჭყალების სტილის მართვა<br>• ბეჭდვისას კონტენტის მართვა<br>• ბგერის ხმამაღლობის, პაუზების მართვა |
| CSS2.1 | 09.2009 | • მოხდა CSS2-ში დაშვებული შეცდომების გასწორება და ზოგიერთი მომენტის დაზუსტება, მათ შორის პერსპექტივის |
| CSS3 | | • მომრგვალებული კუთხეების მხარდაჭერა<br>• გრადიენტული საზღვრების მხარდაჭერა |

| | | |
|---|---|---|
| | | • ელემენტის ჩრდილების მართვა<br>• არასტანდარტული შრიფტების გამოყენების შესაძლებლობა<br>• მომხმარებლის ბლოკების ზომების მართვა<br>• ტექსტის სვეტებად დაყოფა<br>• და ზოგი სხვა გრაფიკული ეფექტი |

**კითხვები თვითშეფასებისთვის:**

- შესაძლებელია თუ არა CSS1 ის მეშვეობით, სიების დაფორმატება და მისი ვიზუალიზაცია?
- დოკუმენტში შეგვიძლია თუ არა განვათავსოთ ელემენტები ფენებად?
- რამდენად ურთიერთქმედებს CSS-ი HTML-ით შექმნილი ცხრილების უჯრებთან და მათ შიგთავსთან?

## დოკუმენტში სტილების შემოტანა და მათი გამოყენების მეთოდები

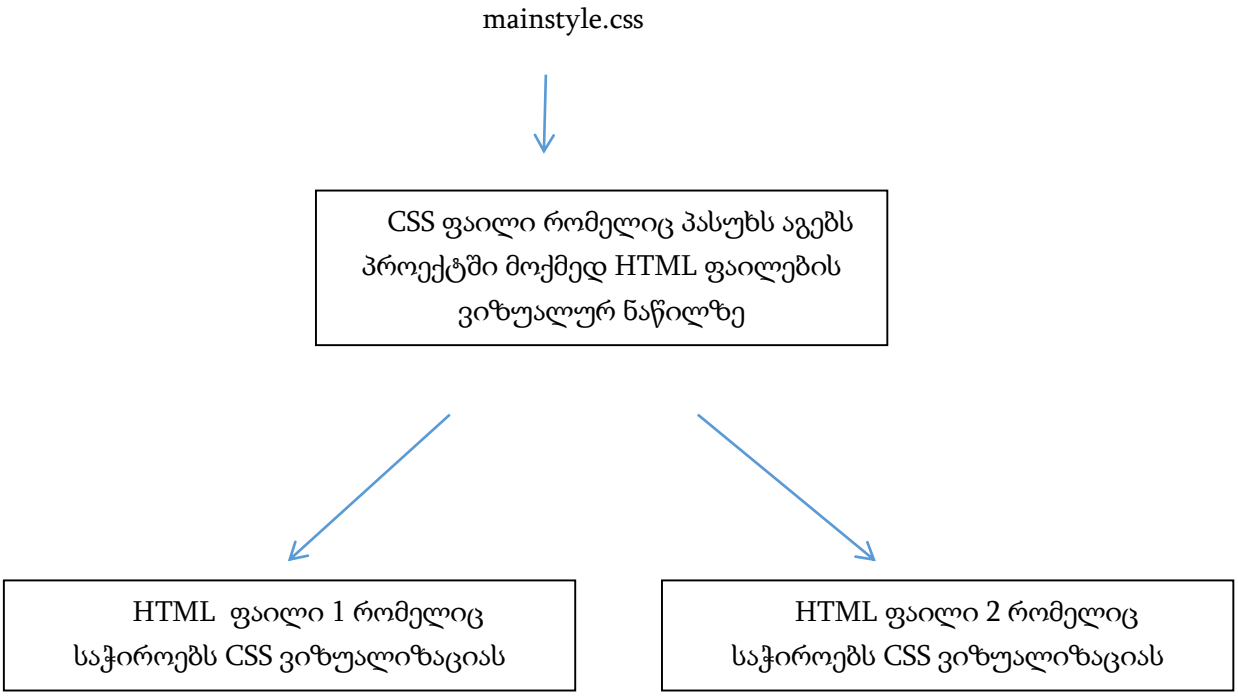
HTML დოკუმენტში არსებობს სტილის შემოტანის 3 მეთოდი, ესენი გახლავთ

1. სტილის დოკუმენტში იმპორტირება - External style sheet
2. სტილის დოკუმენტის თავის (head) განყოფილებაში ჩაშენება - Internal style sheet
3. ხაზოვანი სტილი - Inline style

**სტილის დოკუმენტში იმპორტირება** - External style sheet

HTML დოკუმენტში CSS სტილის ჩანერგვის ერთი მეთოდი გახლავთ ცალკე CSS ფაილის შექმნა მაგალითად (mainstyle.css) მისი შემდგომი იმპორტირებისთვის. გთხოვთ ყურადღება მიაქციოთ ფაილის გაფართოებას .css ანუ მსგავსი ტიპის ფაილები, როგორც წესი იქმნება გლობალურად მთელი პროექტის ძირითადი ვიზუალური სტილის შესაქმნელად. მასში ვათავსებთ ხოლმე, პროექტის ძირითადი შრიფტის ზომას, ფერს, სათაურების სტილებს და ასე შემდეგ. მოკლედ რომ ვთქვათ ასეთი იმპორტირებისთვის გამიზნული CSS ფაილი ემსახურება სრული პროექტის ვიზუალიზაციას და მის საბაზო ეფექტებს. პრინციპი შემდეგია: იქმნება css გაფართოების ფაილი, ხოლო შემდგომ ამ ფაილს იმპორტირებით ვაშენებთ ჩვენთვის სასურველ HTML დოკუმენტში. ქვემოთ წარმოგიდგენთ პრინციპს და სინტაქსს თუ როგორ გამოვიყენოთ ზემოთხსენებული მიდგომა CSS ფაილთან მიმართებაში.

3.1 სურათზე მოცემული სქემიდან ნათლად გამოიხატება, რომ ერთი CSS ფაილი შეგვიძლია გამოვიყენოთ რამდენიმე ფაილში, რაც იმას ნიშნავს, რომ მთელი პროექტისთვის წინასწარ მოფიქრებული და გათვლილი ფონტის ზომა, დოკუმენტის ფონის ფერი ან მისი გრაფიკული ფონის მართვა და ასე შემდეგ შეგვიძლია ერთი ფაილიდან.




სურ. 3.1. სტილის დოკუმენტში იმპორტირების სტრუქტურა

რაც შეეხება კოდის ნაწილს, მოცემული კოდის ფრაგმენტი აღწერს თუ როგორ შეგვიძლია გარე CSS ფაილის იმპორტირება რომელიმე HTML დოკუმენტში.

```
<head>
<link rel="stylesheet" type="text/css" href="mainstyle.css">
</head>
```

აქვე გავაკეთოთ მინიშნება იმის შესახებ, რომ იმპორტირებულ mainstyle.css ფაილში გაწერილი აბსოლუტურად ყველა ატრიბუტი/თვისება იქონიებს ზემოქმედებას HTML ფაილის ვიზუალურ ნაწილზე, რომელშიც გამოვიძახეთ კონკრეტული mainstyle.css ფაილი იმპორტირების მეთოდით. ანუ დაიმპროტირებულ ფაილში თუ გაგვაჩნია მითითება, რომ ფონტის ფერი იყოს წითელი მთელი დოკუმენტისთვის ან რომელიმე კონკრეტული ობიექტისთვის ჩათვალეთ, რომ ეს ასეც იქნება თუ რა თქმა უნდა ყველაფერი სწორად გვაქვს ორგანიზებული.

სტილის ფაილის იმპორტირების შემთხვევაში საჭიროა ყურადღებით გავითვალისწინოთ ჩვენი პროექტის ფაილური წყობა, ანუ რეალურად თაგს <LINK> ატრიბუტში HREF ენიჭება არა ფაილის სახელი მხოლოდ, არამედ ფაილის URL (Universal Reosource Location) მისამართი.

მაგ. თუ სტილის ფაილებისათვის გამოყოფილის ცალკე საქაღალდე styles, რომელშიც არის მოთავსებული mainstyle.css ფაილი, მაშინ ამ ფაილის html დოკუმენტში ექსპორტირებისას უნდა მიეთითოს საქაღალდის სახელი styles.

```
    <head>
<link rel="stylesheet" type="text/css" href="styles/mainstyle.css">
</head>
```

**სტილის დოკუმენტის თავის განყოფილებაში ჩაშენება** - Internal style sheet

რეალურად რომ მივუდგეთ ამ საკითხს, ანუ არჩევანის წინაშე რომ დავდგეთ რომელი მეთოდი ჯობია, სტილის იმპორტირება თუ მისი დოკუმენტის სტრუქტურაში ჩაშენება? ცალსახა პასუხი ამაზე არ არსებობს, ყოველივე გამომდინარეობს პროექტის სპეციფიკიდან, ანუ უკეთესი არჩევანის გაკეთება ამ ორ ვარიანტს შორის დამოკიდებულია პროექტის სტრუქტურაზე და ზოგადად არ შეიძლება განისაზღვროს ამ ორიდან რომელი უფრო ოპტიმალურია.

სტილის დოკუმენტის თავში ჩაშენება ხორციელდება მოცემული კოდის ფრაგმენტის საშუალებით.

```
    <head>
<style type="text/css">

    ... ამ ნაწილში ხდება სტილის თვისებების აღწერა ...

    </style>
</head>
```

გთხოვთ ყურადღება მიაქციოთ, რომ ეს თაგი კონტეინერია ანუ მოითხოვს შესაბამის დახურვას. ასეთი გზით ერთსადაიმავე დოკუმენტში სტილის ჩაშენება შეგვიძლია უთვალავი რაოდენობით, საჭიროების და მიხედვით.

აქვე დავაზუსტოდ, რატომ მაინცდამაინც დოკუმენტის თავში ვაშენებთ სტილს? როგორც წესი დოკუმენტის ბრაუზერში ვიზუალური ასახვა იწყება ბრაუზერის მიერ თაგი <BODY>-ის წაკითხვის შემდეგ, ანუ რადგან სტილი პასუხს აგებს დოკუმენტის ვიზუალურ ნაწილზე უმჯობესია ბრაუზერმა ჯერ წაიკითხოს ის CSS ატრიბუტები და მნიშვნელობები რომელიც აფორმირებს ჩვენს ვიზუალურ ნაწილს და მხოლოდ ამის შემდგომ დაიწყოს დოკუმენტის ბრაუზერში ასახვა.

**ხაზოვანი სტილი** - Inline style

ხაზოვანი სტილი გამოყენების სახელდახელო მეთოდს წარმოადგენს CSS ატრიბუტების თაგში ჩაშენება. სტილის მსგავსი მიდგომით გამოყენება საკმაოდ მოუქნელია და მხოლოდ იშვიათ შემთხვევებში გამოიყენება. მისი გამოყენების სინტაქსი შემდეგნაირია:

```
<img src="image.jpg" alt=" " style="width:300px;height:200px;" />
```

აღნიშნული მაგალითი გამოიყენება კონკრეტულად ერთი თაგისთვის, ამ შემთხვევაში გრაფიკული გამოსახულებისთვის რომელსაც CSS ატრიბუტებით ვანიჭებთ სიმაღლეს (200px) და სიგანეს (300px) პიქსელებში.

## CSS ის წერის სინტაქსი

როგორც მარკირების და პროგრამირების ყველა ენას გააჩნია თავისებური წერის სინტაქსი, ასევე გახლავთ CSS ენაც. საკმარისია უმნიშვნელოს სინტაქსი დაირღვეს და ჩათვალეთ, რომ ვეღარ მივიღებთ იმ ეფექტს რაც ჩანაფიქრში გვქონდა. ანუ სინტაქსის ძირითადი პრინციპი შემდეგნაირია: ვირჩევთ სელექტორს (ამაზე მოგვიანებით გვექნება საუბარი), შემდეგ ვირჩევთ კონკრეტულ მის თვისებას და ვანიჭებთ ნებისმიერ დასაშვებ მნიშვლობას.

```
სელექტორის სახელი { თვისება1: მნიშვნელობა; თვისება2: მნიშვნელობა; }
```

მაგალითად: გრაფიკული გამოსახულება გვინდა რომ იყოს 200px სიგანის და 300px სიმაღლის

```
img { height: 300px;  width: 200px; }
```

ზემოხსენებულ მაგალითში img გახლავთ სელექტორი, ანუ კონკრეტულად თაგი რომელსაც გამოსახულება შემოაქვს დოკუმენტზე, ფიგურულ ფრჩხილებში კი ვირჩევთ მის კონკრეტულ პარამეტრებს და ვანიჭებთ მნიშვნელობებს, ანუ მაგალითად პარამეტრი height-ის არჩევის შემდგომ ვწერთ ორწერტილს, რაც მინიჭებას ნიშნავს და ამის შემდგომ ამ პარამეტრს ვანიჭებთ კონკრეტულ მნიშვნელობას ამ შემთხვევაში 300px-ს და ვუკონკრეტებთ რომ საზომი ერთეული არის px ანუ (Pictre Single Element) ყველაზე უმცირესი წერტილი მონიტორის რეზოლუციაზე (pixel). პირველ პარამეტრზე მნიშვნელობის მინიჭების შემდეგ იწერება წერტილმძიმე და შესაძლებელია სხვა პარამეტრის გამოყენება.

აქვე გავაკეთოთ პატარა მაგრამ მნიშვნელოვანი მინიშნება, როგორც ყველა კოდირების სინტაქსი ითვალისწინებს უშუალოდ კოდში კომენტარის ჩანაწერის წარმოებას, ასევე CSS იც არ გახლავთ გამონაკლისი. CSS ის კოდში კომენტარი იწერება შემდეგნაირად

```
/* ეს გახლავთ CSS კომენტარი */
```

**კითხვები თვითშეფასებისთვის:**

- რამდენად მისაღებია ერთი CSS ფაილის სხვადასხვა HTML დოკუმენტთან გამოყენება?
- აუცილებელია თუ არა ჩავაშენოთ CSS-ი HTML დოკუმენტის თავის განყოფილებაში?

- HTML დოკუმენტში CSS სტილის შემოტანის რამდენი მეთოდი არსებობს?

## CSS კოდის წერის ეთიკა

ზემოთმოყვანილი მაგალითიდან ჩანს თუ როგორ შეგვიძლია CSS სინტაქსის გამოყენება HTML დოკუმენტში, თუმცა აქვე უნდა აღინიშნოს, ვრცელი პროექტების ფორმირების დროს ძალიან მნიშვნელოვანია დავიცვათ გარკვეული უკვე კარგად აპრობირებული წერის მანერა და ეთიკა, ვინაიდან ვრცელი პროგრამული კოდების გარჩევის დროს მარტივი იყოს მათი თვალისთვის და გონებისთვის აღქმა. ზემოთმოყვანილი მაგალითი დამეთანხმებით თვალისთვის უფრო გარჩევადი იქნებოდა ქვევით მოცემული ფორმით რომ ჩაგვეწერა. ანუ სელექტორი და ყოველი მომდევნო თვისება ახალი სტრიქონიდან ტაბულაციით მარჯვნივ შეწეული რომ დავწეროთ.

```
img                                                    {
    height:                                            300px;
    width:                                             200px;
    }
```

**კითხვები თვითშეფასებისთვის:**

- იმოქმედებს თუ არა CSS -ის წერის ეთიკის დარღვევა HTML დოკუმენტის ვიზუალურ ნაწილზე?
- რატომ ვცდილობთ დავიცვათ CSS -ის წერის ეთიკა?

## სელექტორები და მათი ზოგადი გამოყენების ცნებები

ტერმინი „სელექტორი“ CSS-თან მიმართებაში გამოიყენება დოკუმენტში კონკრეტული ელემენტის პარამეტრების (ატრიბუტების) მნიშვნელობის სამართავად. დოკუმენტში კონკრეტულ ელემენტს შეგიძლია მივმართოთ სხვადასხვა მიმართვის წესის გამოყენებით. ესენია:

1. მივმართოთ ელემენტს თაგის სახელით
2. მივმართოთ ელემენტს ID-ის მეშვეობით
3. მივმართოთ ელემენტს წინასწარ გამოცხადებული კლასის მეშვეობით

განვიხილოთ ზემოთჩამოთვლილი სელექტორების სპეციფიკაცია და მათი გამოყენების ასპექტები.

**ელემენტზე თაგის სახელით მიმართვა** მსგავსი მიმართვა გამოიყენება კასკადურად დოკუმენტში კონკრეტული თაგის მიმართ ინსტრუქციის გადასაცემად.

მაგ:

```
p                                                    {
                                        font-size:          15px;
}
```

აღნიშნული კოდის ნაწილი მიუთითებს, რომ დოკუმენტში ყველა თაგი სახელად  უნდა იყოს 15 პიქსელის ზომის ფონტით.

**კონკრეტულ ელემენტს მისი ID -ის მეშვეობით მიმართვა.** ID უნიკალური უნდა იყოს დოკუმენტში (ანუ არ უნდა მეორდებოდეს).

```
#paragraph                                           {
                                        font-size:          15px;
}
```

ზემოთ მოცემული კოდიდან ჩანს რომ id სელექტრზე მიმართვისათვის ვიყენებთ # დიეზის სიმბოლოს. მოცემული მითითება აღნიშნავს, რომ მხოლოდ ის ელემენტი დოკუმენტში რომელსაც გააჩნია id="paragraph" აისახება 15 პიქსელის ზომის ფონტით.

მაგ. Hello World!

**დოკუმენტის ელემენტზე კლასის მეშვეობით მიმართვა.** წინასწარ გამოცხადებული გარკვეული პარამეტრები შეგვიძლია გამოვიძახოთ ნებისმიერ დოკუმენტის ელემენტთან

```
.myStyle                                             {
                                        font-size:          15px;
}
```

ზემოთ მოცემული კოდიდან ჩანს რომ კლასზე მიმართვისათვის ვიყენებთ . წერტილს. აღნიშნული მითითება უზრუნველყოფს, რომ CSS თვისებამ იმოქმედოს კონკრეტულად იმ ტაგზე რომელსაც ატრიბუტად გააჩნია CLASS და მის მნიშვნელობად მინიჭებული აქვს ჩვენთვის სასურველი კლასის სახელი.

მაგ. Hello World!

*კითხვები თვითშეფასებისთვის:*

- რას წარმოადგენს ტერმინი „სელექტორი“ CSS ენაში ?
- რამდენი ტიპის სელექტორი არსებობს CSS ენაში?

- ჩამოთვალეთ თითოეული სელექტორის თვისებები
- ჩამოთვალეთ თითოეული სელექტორის გამოყენების წესი

## 6.2. ვებსაიტის ტექსტური ელემენტების გაფორმება

**მიმდინარე პარაგრაფის თემატიკა**

- ზომის ერთეულები
- ვებგვერდის ფონის პარამეტრები და მნიშვნელობები
- შრიფტის თვისებები და მათი მნიშვნელობები
- ვებფონტების შესაძლებლობები
- ვებფონტების გენერირების საშუალებები
- ვებფონტების ასახვის შესაძლებლობები
- ტექსტის თვისებები და მნიშვნელობები
- ბმულების  თვისებები და გამოყენების შესაძლებლობები

### ზომის ერთეულები და მათი გამოყენების ასპექტები

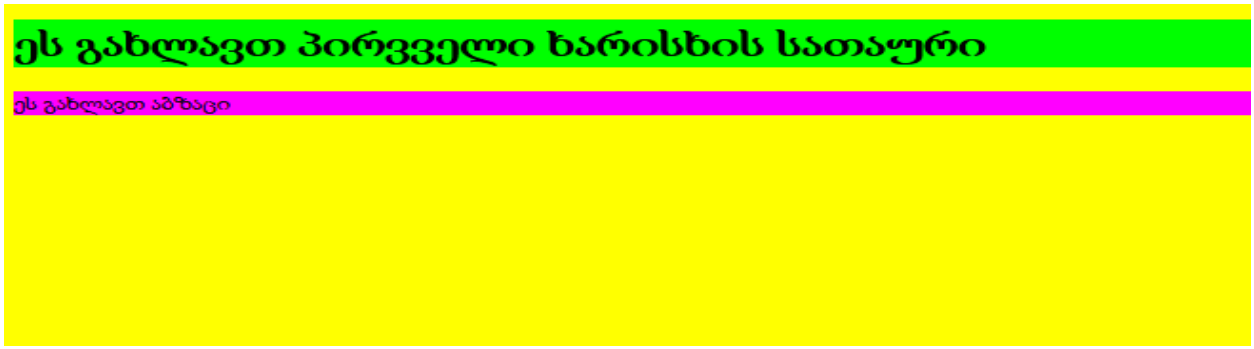
WEB გვერდის შესაქმნელად, მასზე ობიექტების განთავსება და ზომების მითითება შესაძლებელია სხვადასხვა საზომი ერთეულებით:

| | |
|---|---|
| cm | სანტიმეტრი |
| mm | მილიმეტრი |
| In | ინჩი ( 1ინჩი = 96პიქსელი (px) = 2.54 სანტიმეტრი |
| px | პიქსელი (1 პიქსელი  = 1/96 ედი  1ინჩი) |
| pt | პოინტი (1პოინტი =1/72-ი 1 ინჩი) |

თუმცა პრაქტიკამ გვაჩვენა, რომ ყველაზე მოსახერხებელი, ზემოთ ჩამოთვლილი საზომი ერთეულებიდან პიქსელებია (PX). ვინაიდან ყველა მოწყობილობის მონიტორის რეზოლუცია იზომება პიქსელებში.

background-color: ფერი – განსაზღვრავს ფონის ფერს ელემენტისათვის

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
    background-color: yellow;
}

h1
{
    background-color: #00ff00;
}

p
{
    background-color: rgb(255,0,255);
}
</style>
</head>
<body>

<h1>ეს გახლავთ პირველი ხარისხის სათაური</h1>
<p>ეს გახლავთ აბზაცი</p>

</body>
</html>
```




სურ. 3.2. სხვადასხვა ფონის ელემენტები

ზემოთმოყვანილ მაგალითში გამოყენებულია სწორედ ის სელექტორები რომელზეც წინა ქვეთავში გვქონდა საუბარი. დოკუმენტს აშკარა მითითებით გადავცემთ დოკუმენტის ტანის ფერს (yellow - ყვითელი) ანუ გლობალურად ფონის ფერს, პირველი დონის სათაურის ფონის ფერს (მწვანე – 00ff00) და აბზაცის ფონის ფერს (ვარდისფერს - rgb(255,0,255)). ზემოთ აღწერილი მაგალითიდან გამომდინარე, რამდენი აბზაციც გვექნება დოკუმენტში და რამდენი პირველი დონის სათაურიც <h1> ყველა მათგანი გაიზიარებს ზემოაღწერილ სტილს.

background-image: url("ფაილის სახელი") - უზრუნველყოფს გრაფიკული ფონის შექმნას კონკრეტული ელემენტისთვის.

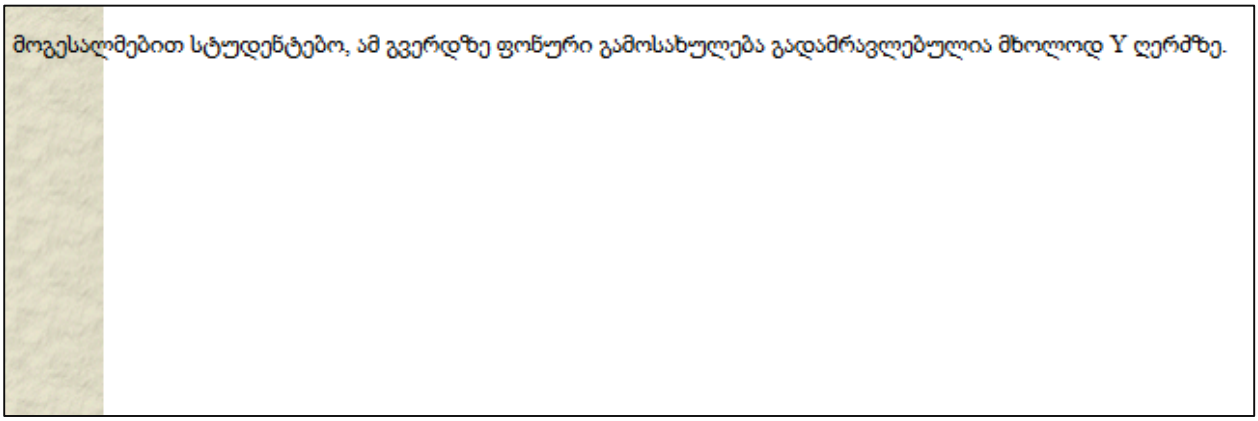
ზემოთ მოყვანილ მაგალითს თუ დააკვირდებით, ნახავთ რომ დოკუმენტის ფონზე შემოტანილი არის ერთი გრაფიკული გამოსახულება (სურ. 3.3), რომელიც მოზაიკის წესით არის გადამრავლებული (სურ. 3.4). ე.ი. როდესაც ფონად ვუთითებთ დოკუმენტის რომელიმე ელემენტს გრაფიკულ გამოსახულებას ის ავტომატურად მრავლდება X და Y ღერძზე უსასრულოდ, თუმცა ქვემოთ მოყვანილ მაგალითებში ვიხილავთ, რომ CSS ის მეშვეობით ამ ქმედების კონტროლიც შეგვიძლია

შენიშვნა: თუ ფონური გამოსახულების ფაილი და html დოკუმენტი არ მდებარეობს ერთ საქაღალდეში, ფაილზე მიმართვისათვის საჭიროა მიეთითოს სრული მისამართი.

background-repeat - განსაზღვრავს ფონური გამოსახულების განმეორებადობას

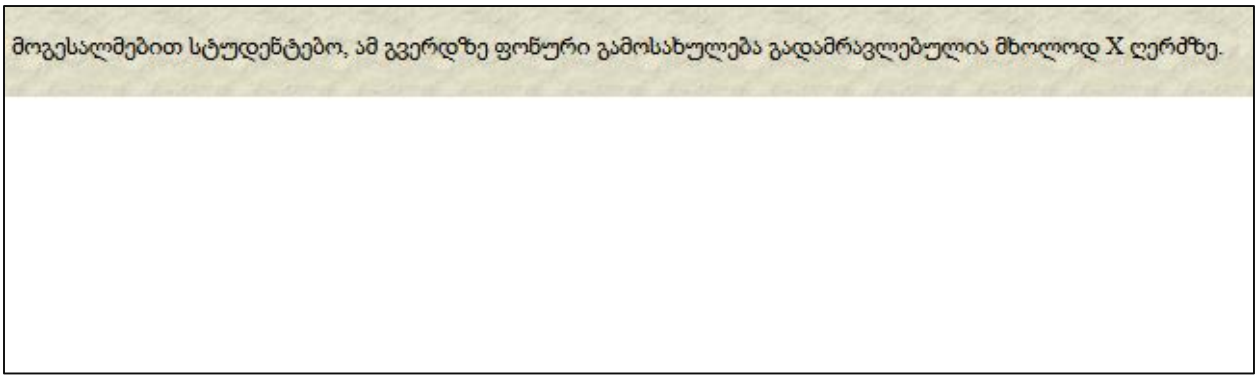
- background-repeat: repeat-y - ფონის გამოსახულების y ღერძზე განმეორადობა;
- background-repeat: repeat-x - ფონის გამოსახულების x ღერძზე განმეორადობა;
- background-repeat: no-repeat - ფონის გამოსახულება განმეორდება.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("paper.gif");
    background-repeat: repeat-y;
}
</style>
</head>
<body>

<p>მოგესალმებით სტუდენტებო, ამ გვერდზე ფონური გამოსახულება გადამრავლებულია მხოლოდ Y
ღერძზე.</p>

</body>
</html>
```




სურ. 3.5. ფონის გამოსახულების y ღერძზე განმეორადობა

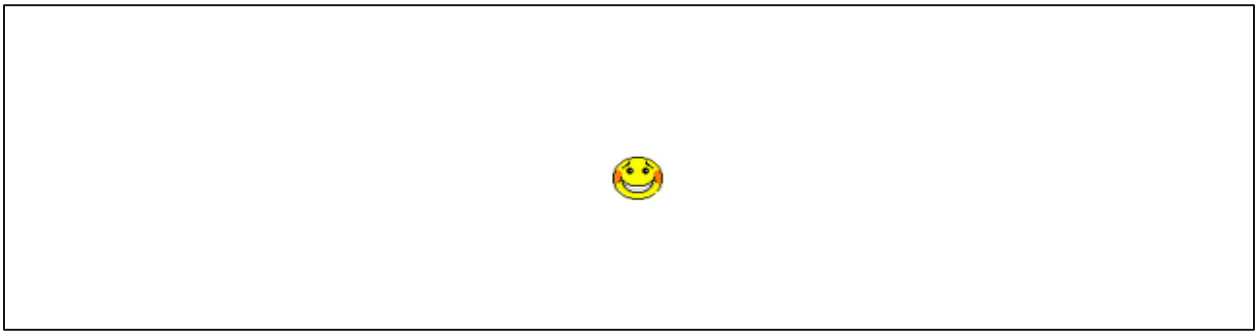
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("paper.gif");
    background-repeat: repeat-x;
}
</style>
</head>
<body>

<p>მოგესალმებით სტუდენტებო, ამ გვერდზე ფონური გამოსახულება გადამრავლებულია მხოლოდ X ღერძზე.</p>

</body>
</html>
```




სურ. 3.6. ფონის გამოსახულების x ღერძზე განმეორადობა

background-position - განსაზღვრავს ფონის სურათის პოზიციის დაწყებას

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url('smiley.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
}
</style>
</head>

<body>
</body>

</html>
```



სურ. 3.7. ფონის გამოსახულების პოზიციის განსაზღვრა

შესაძლებელია გამოვიყენოთ ფონის განსაზღვრის გაერთიანებული ვერსია, სადაც მნიშვნელობები მიეთითება შემდეგი თანმიმდევრობით:

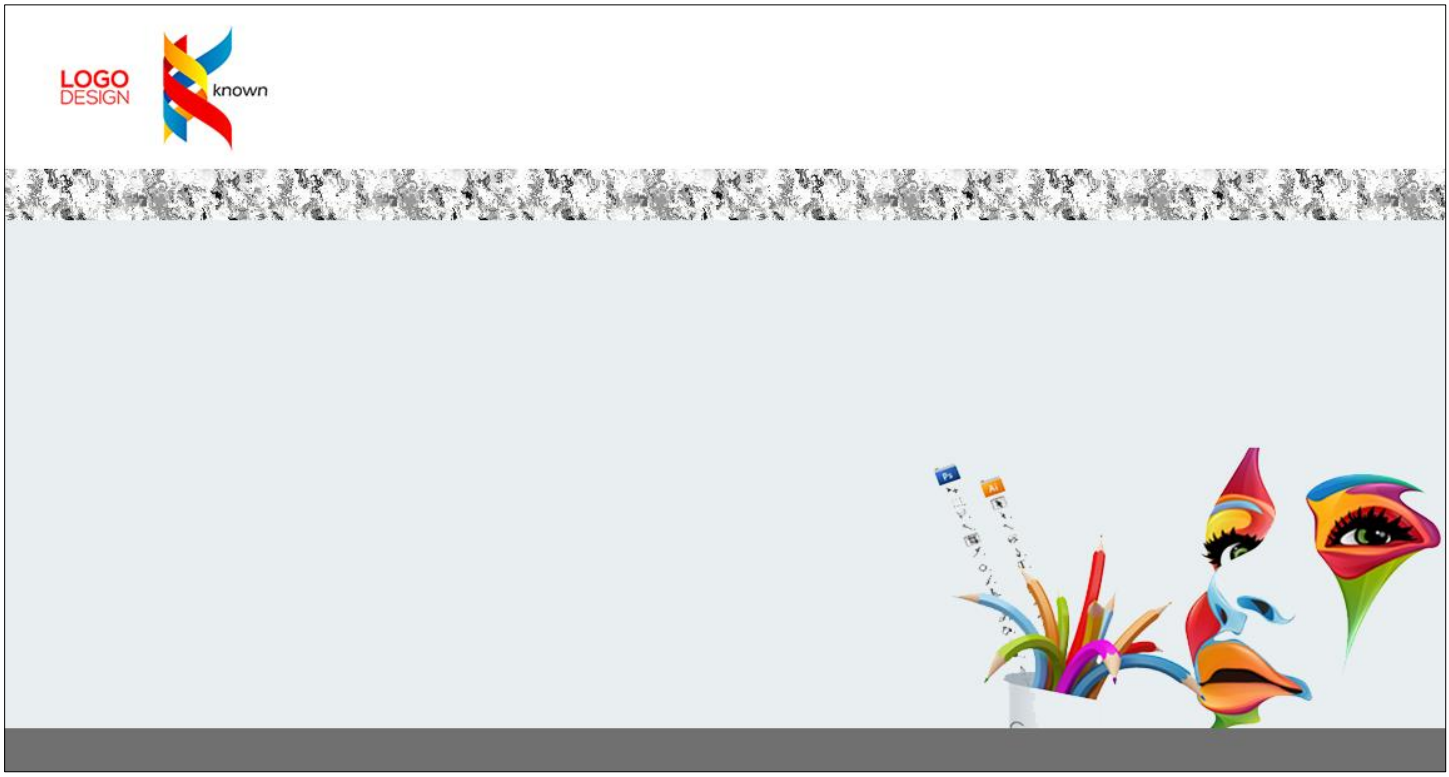
1. ფონის ფერი
2. გრაფიკული გამოსახულება
3. გამოსახულების განმეორებადობა
4. გამოსახულების ასახვის პოზიცია

```
 body {
background: #ffffff url("img_tree.png") no-repeat right top;
}
```

შენიშვნა: თუ რომელიმე მნიშვნელობა არ არის მითითებული, ბრაუზერი გამოიყენებს ამ თვისების ავტომატურ მნიშვნელობას.

მაგალითისათვის განვიხილოთ სურ. სურ. 3.8-ზე მოცემული ვებგვერდის გაფორმება ფონური ელემენტებით.

ვებგვერდი დაყოფილია ოთხ ძირითად ნაწილად: header - თავი, menu- მენიუ, main - ძირითადი ნაწილი, footer - ბოლო. შესაბამისი ბლოკური ელემენტები უნდა განისაზღვროს html დოკუმენტში და აღიწეროს ცალკეულის ფონური პარამეტრები. სურ. 3.9- ზე მოცემულია შესაბამისი გრაფიკული გამოსახულებები.




სურ. 3.8. ვებგვერდის გაფორმება ფონური ელემენტებით

menubg.png


logo.png

mainbg.png

სურ. 3.9. მოცემული გრაფიკული გამოსახულებები

```
<!DOCTYPE html>
<html>
<head>
<title>ფონები</title>
<meta charset="UTF-8">
<style>
#header{
    background:#ffffff;
    }
#menu{
    background:url(menubg.png); repeat-x;
    height:35px;
    }
#main{
    background:#e9eff1 url(mainbg.png) no-repeat right bottom;
    height:400px;
    }
#footer{
    background:#707070;
    height:30px;
    }
</style>
</head>
<body>

<div id="header">
    <img src="logo.png" alt=""/>
</div>

<div id="menu"> </div>

<div id="main"> </div>

<div id="footer"> </div>

</body>
</html>
```

სურ. 3.10. ფონის თვისებების გამოყენების ნიმუში

*კითხვები თვითშეფასებისთვის:*

- შეიძლება თუ არა დოკუმენტზე გამოვიყენოთ გრაფიკული ფონური გამოსახულება?
- შესაძლებელია თუ არა ფონური გრაფიკული გამოსახულების პოზიციის კონტროლი?
- ავტომატურად , როგორ  გადანაწილდება დოკუმენტში გრაფიკული ფონი?
- შეგვიძლია თუ არა მივუთითოთ გრაფიკულ ფონს თუ როგორ გადანაწილდეს დოკუმენტზე?

## შრიფტის თვისებები და მათი მნიშვნელობები

შრიფტთან მუშაობა ამთავითვე დავიხსომოთ: მომხმარებლის ეკრანზე აისახება მხოლოდ ის შრიფტები, რომლებიც დაყენებულია მის კომპიუტერზე, შესაბამისად, ჯობია, "ეგზოტიკური" შრიფტების გამოყენებისაგან თავი შევიკავოთ ან უკიდურეს შემთხვევაში, თუ მიგვაჩნია, რომ ეს აუცილებელია, ასეთი შრიფტებით გაკეთებული ჩანაწერები გრაფიკულ ფორმატში გადავიყვანოთ (შევნიშნავთ, რომ ნათქვამი არ ეხება უნიკოდს).

ქვემოთ ჩამოთვლილია სტილების შემნახველ კასკადურ ცხრილებში გამოყენებული ის ატრიბუტები, რომლებიც ემსახურება შრიფტის მართვას:

- **font-family** – შრიფტის სახეს განსაზღვრავს.
- **Font-style** – სტილის აღმნიშვნელი
- **font-size** – განსაზღვრავს შრიფტის ზომას (აბსოლუტური ან ფარდობითი სახით ზემოთ ჩამოთვლილი საზომი ერთეულების გამოყენებით)

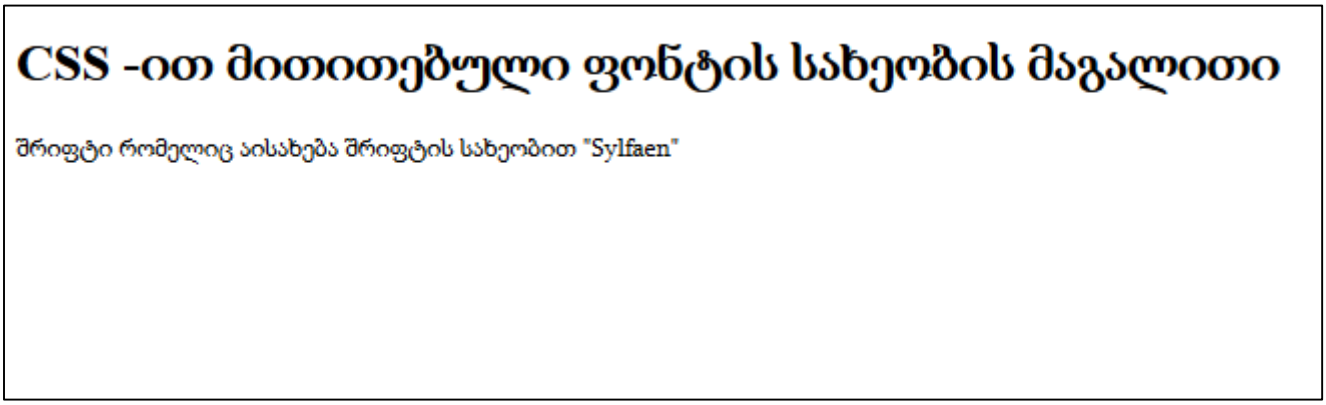
მაგალითად: font-size: 12px, font-size: 100%, font-size: 2.5em;

განვიხილოთ თითოეული ზემოთ ჩამოთვლილი შრიფტთან სამუშაოდ ჩამოთვლილი ატრიბუტი.

**font-family**- როგორც უკვე აღვნიშნეთ მოცემული ატრიბუტით შეგვიძლია განვსაზღვროთ თუ რომელი შრიფტი გამოიყენოს დოკუმენტმა ტექსტის ასახვის დროს. ვინაიდან ზემოთ უკვე გავუსვით ხაზი იმ გარემოებას, რომ მომხმარებელი შრიფტს დაინახავს მხოლოდ იმ შემთხვევაში თუ ეს შრიფტი მის კომპიუტერში უკვე არსებოს, ვცდილობთ დოკუმენტის ტექსტის გადმოსაცემად შევარჩიოთ სტანდარტული ტიპის შრიფტები, ანუ საუბარი არის იმ ტიპის შრიფტებზე, რომლებიც ავტომატურად მოყვება ჩვენთვის ცნობილ და პოპულარულ ოპერაციულ სისტემებს.

მაგალითად: შემდეგი კოდი გვაჩვენებს თუ როგორ უნდა ავსახოთ ტექსტი Sylfaen-ურის ტიპის შრიფტით ანუ ყველასათვის ნაცნობი ქართული შრიფტით.

```
<!DOCTYPE html>
<html>
<head>
<title>დოკუმენტის სათაური</title>
<style>
 p
{
  font-family:Sylfaen;
}

</style>
</head>
<body>

<h1>CSS -ით მითითებული ფონტის სახეობის მაგალითი</h1>
<p>შრიფტი რომელიც აისახება შრიფტის სახეობით "Sylfaen"</p>

</body>
</html>
```




სურ. 3.11. შრიფტის სახეობის გამოყენების ნიმუში

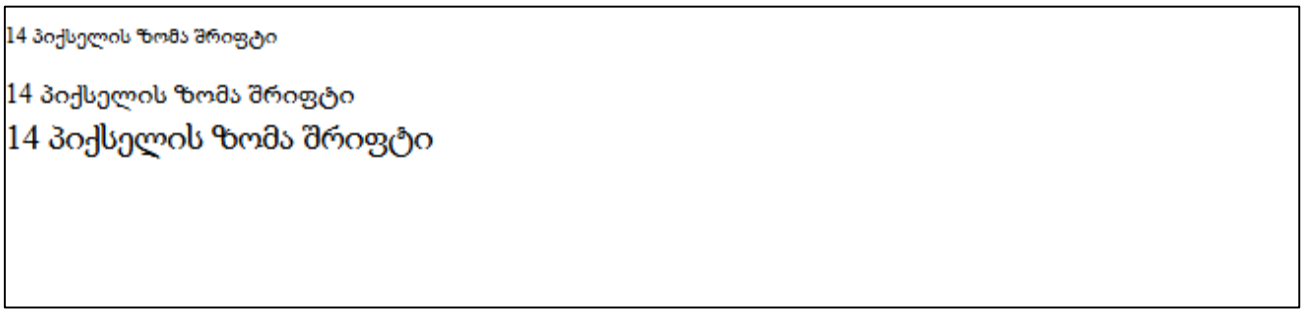
აქვე გვინდა პატარა მინიშნება გავუკეთოთ იმ გარემოებას, რომ შრიფტის სახელი ზუსტად ისე უნდა იყოს მითითებული ამ თვისების მნიშვნელობაში რეგისტრის მიხედვით როგორც შრიფტს სინამდვილეში ჰქვია.

**font style** – სტილის აღმნიშვნელი, რაც შეეხება მოცემულ თვისებას მან შეიძლება მაგალითად მიიღოს შემდეგი მნიშვნელობები:

- normal;
- italic;

**font-size** – განსაზღვრავს შრიფტის ზომას ჩვენთვის სასურველი საზომი ერთეულით

```
<!DOCTYPE html>
<html>
<head>
<title>დოკუმენტის სათაური</title>
<style>
 p
{
  font-size:14px
}
 div
{
  font-size:18px
}

span
{
  font-size:22px
}

</style>
</head>
<body>
<p>14 პიქსელის ზომა შრიფტი</p>
<div>14 პიქსელის ზომა შრიფტი</div>
<span>14 პიქსელის ზომა შრიფტი</span>

</body>
</html>
```




სურ. 3.12. შრიფტის  თვისებების გამოყენების ნიმუში

### ვებ ფონტების შესაძლებლობები

ვებ ფონტები ჩვეულებრივი სტანდარტული ფონტებისაგან (შრიფტებისაგან) განსხვავებით ვებ გვერდებზე გამოიყენება არასტანდარტული შრიფტებით ტექსტის ასახვის მიზნით. ანუ თუ ჩვენ გვინდა ჩვენს მიერ შექმნილ ვებ პროექტში გამოვიყენოთ რაიმე განსხვავებული შრიფტი, რომელიც რეალურად შესაძლებელია, რომ მომხმარებელს არ ჰქონდეს კომპიუტერზე დაინსტალირებული მაშინ დასახმარებლად მივმართავთ სწორედ ვებ ფონტებს. ვებ ფონტების მთელი ხიბლი და მისი გამოყენების მოხერხებულობა იმაში მდგომარეობს, რომ იგი იტვირთება ბრაუზერში ვებ გვერდთან ერთად (დინამიურად) ასეთი მიდგომით ჩვენ და მომხმარებელიც გარანტირებულია, რომ შრიფტი ნებისმიერ შემთხვევაში ისეთი სახით აისახება ბრაუზერში როგორც ეს ჩვენ (ავტორებს) გვქონდა გათვლილი. თუმცა არსებობს ერთი ნიუანსი რომელიც ძალიან საყურადღებოა, უნდა გავითვალისწინოთ ის მნიშვნელოვანი

ფაქტიც, რომ ვებ ფონტებს ყველა ბრაუზერი ერთნაირად არ აღიქვამს და ამ დეტალმა შეიძლება გაუთვალისწინებელი პრობლემა შეგვიქმნას ბრაუზერების გარკვეულ ვერსიებთან. მოდით აქვე დავაზუსტოთ, ვებ ფონტებს გააჩნია 4 ფორმატი, სწორედ ეს იწვევეს ბრაუზერებთან მიმართებაში ვებ ფონტების მოქნილად გამოყენების საშუალებას ვინაიდან პრობლემის აღმოფხვრაში სწორედ ოთხივე ფორმატის ფონტის შემოტანა გვიწყობს ხელს.

ეს ფორმატებია:

- TTF
- WOFF
- EOT
- SVG

ანუ იმისათვის რომ უზრუნველვყოთ ჩვენი ვებ პროექტი ვებ ფონტების სწორად ასახვისთვის, ამისათვის საჭირო გახლავთ ჯერ დავაგენერიროთ ეს შრიფტები სპეციალური ინსტრუქციის მიხედვით და შემდგომ განვათავსოთ იგი სერვერზე ვინაიდან მოგვეცეს საშუალება საჭიროების და მიხედვით მივაკითხოთ მის URL მისამართს და უზრუნველვყოთ ვებ გვერდთან სინქრონულად მათი ჩატვირთვა მომხმარებლის კომპიუტერში.

### ვებ ფონტების გენერირების საშუალებები

რაც შეეხება ვებ ფონტის (შრიფტის) გენერირების პროცესს, ეს საკმაოდ მარტივია, ძალიან ბევრი ონლაინ რესურსის მოპოვება შეიძლება მსგავსი ოპერაციის შესასრულებლად. პროცედურა დაახლოებით ყველგან ერთმანეთის ანალოგიურია, თქვენ, თქვენს კომპიუტერში უკვე არსებული შრიფტი უნდა ატვირთოთ რომელიმე შრიფტ-გენერატორ საიტზე და ის თქვენს შრიფტს დაგიბრუნებთ უკვე ვებფონტის ფორმატში. საცდელად შეგიძლიათ გამოიყენოთ ეს რესურსი: https://www.web-font-generator.com მოცემულ ვებ რესურსზე შესვლისთანავე მიყევით ძალიან მარტივ ინსტრუქციას.

### ვებ ფონტების ასახვის შესაძლებლობები

რაც შეეხება ვებ ფონტების ბრაუზერში დოკუმენტზე ასახავს, ამისათვის არსებობს CSS ის ენაში სპეციალურად განკუთვნილი დეკლარაცია- ეს გახლავთ შემდეგი სახის ჩანაწერი @font-face, რომელიც უზრუნველყოფს დინამიურად ვებ ფონტების ჩატვირთვას ვებ გვერდთან ერთად. მისი სრული გამოყენების სინტაქსი და წესი შემდეგნაირია:

```
<!DOCTYPE html>
<html>
<head>
<style>
@font-face {
    font-family: myFirstFont;
    src: url(web_font_file.woff);
}

div {
    font-family: myFirstFont;
}
</style>
</head>
<body>

<div>დინამიური ვებ ფონტი.</div>

<p><b>შენიშვნა:</b> ინტერნეტ ექსპლორერი 8 და უფრო ადრეული ვერსიები არ აღიქვამს WOFF ფორმატის
ვებ ფონტებს. ის აღიქვამს მხოლოდ EOT ფორმატის ვებ ფონტებს.</p>

</body>
</html>
```

დინამიური ვებ ფონტი.

**შენიშვნა:** ინტერნეტ ექსპლორერი 8 და უფრო ადრეული ვერსიები არ აღიქვამს WOFF ფორმატის ვებ ფონტებს. ის აღიქვამს მხოლოდ EOT ფორმატის ვებ ფონტებს.

სურ. 3.13 ვებ. შრიფტის  თვისებების გამოყენების ნიმუში

### ვებ ფონტების შესაძლებლობები

მოგეხსენებათ ნებისმიერ ვებ გვერდზე ყველაზე მეტად ინფორმაციის გადმოცემა აქტუალურია ტექსტური სახით. ამიტომ CSS ში ტექსტის დამუშავების და მათი ფორმატირების მდიდარი საშუალებები გახლავთ ჩადებული. ჩვენ ახლა თითოეულ მათგანს ჩამოვთვლით და აღვწერთ.

| თვისება | მნიშვნელობა | აღწერა |
|---|---|---|
| color | თექვსმეტობითი ათვლის სისტემით მინიჭებული მნიშვნელობა, RGB (Red Green Blue) მნიშვნელობა და ფერის სიტყვიერი წარმოდგენა | აღნიშნული თვისების მნიშვნელობით შეგვიძლია შევცვალოთ ტექსტის ფერი კონკრეტული ელემენტისთვის |
| text-align | Left, right, center, justify | მოახდენს ტექსტის ჰორიზონტალურ ხაზზე გასწორების უზრუნველყოფას, |

| | | |
|---|---|---|
| | | მარცხნივ, მარჯვნივ, ცენტრში ან ჩაასწორებს ტექსტს ორივე კიდეზე მარცხნიდან და მარჯვნიდან |
| Text-decoration | Underline, overline, line-through, none | ტექსტს გამოიტანს ქვევიდან ხაზგასმულს, ან ზემოდან ხაზგასმულს, ან გადახაზულს და ან საერთოდ მოხსნის დეკორაციას. ბოლო მნიშვნელობა ძალიან ხშირად გამოიყენება ბმულებთან რათა ბმული არ აისახოს ქვემოდან ხაზგასმული |
| Text-transform | Capitalize, uppaercase, lowercase | ახდენს ტექსტის ტრანსფორმაციას, პირველი მნიშვნელობა ანიჭებს წინადადებაში ტექსტის ყველა სიტყვის პირველ სიმბოლოს მაღალ რეგისტრს, მეორეს აბსოლიტურად ტექსტის ყველა სიმბოლო გადაყავს მაღალ რეგისტრში, უკანასკნელი კი ყველა სიმბოლოს გადაიყვანს დაბალ რეგისტრში |
| Text-indent | მნიშვნელობა საზომ ერთეულებში | გამოიყენება ბლოკის დონის ტაგებთან და განსაზღვრავს ტაგის ტექსტის პირველი ხაზის აბზაცად გამოყოფას იმ ინტერვალით რასაც მივუთითებთ |
| Letter-spacing | მნიშვნელობა საზომ ერთეულებში | განსაზღვრავს სიმბოლოებს შორის დაშორებას მითითებული ინტერვალით |
| Line-height | მნიშვნელობა საზომ ერთეულებში | განსაზღვრავს ტექსტის ხაზებს შორის დაშორებას მითითებული ინტერვალით |
| Direction | Ltr, rtl | მიუთითებს ტექსტის მიმართულებას (მარცხნიდან მარჯვნივ ან მარჯვნიდან მარცხნივ) |
| Word-spacing | მნიშვნელობა საზომ ერთეულებში | განსაზღვრავს სიტყვებს შორის დაშორებას მითითებული ინტერვალით |

**ბმულების თვისებები და გამოყენების შესაძლებლობები**

როგორც HTML-ის ენის სპეციფიკაციიდან მოგეხსენებათ, ბმული ბრაუზერისთვის იყოფა კატეგორიებად, ასე ვთქვათ:

- ბმული რომელიც ჯერ მომხმარებელს არ უნახავს -LINK
- ბმული რომელიც გააქტიურებულია -ACTIVE LINK
- ბმული რომელიც უკვე მომხმარებელმა დაათვალიერა ანუ მოახდინა ამ ბმულით გარკვეულ URL მისამართზე მიმართვა VISITED LINK
- ბმული რომელზეც მაუსის კურსორს მივიტანთ -HOVER (იმართება მხოლოდ CSS-ის საშუალებით)

სინამდვილეში ამ ბმულების ისტორიას ინახავენ ბრაუზერები, რომლებიც შესაბამისად ბმულის სტატუსიდან გამომდინარე ასახავენ მათ სხვადასხვა ფერით. აგრეთვე ბმულის გამოყოფა დოკუმენტზე ბრაზუერების მიერ ხდება მათი ქვემოდან ხაზგასმული სტილით(UNDERLINE).

უმეტეს შემთხვევაში გამომდინარე მოსაზრებიდან, რომ ჩვენი პროექტის დიზაინი მოითხოვს მეტ დახვეწილობას და ვიზუალურად ჩვენებურად წარმოჩენას, CSS -ი გვაძლევს საშუალებას უფრო მოქნილად ვმართოთ ამ ბმულების ვიზუალური თვისებები.

ბმულის თვისებების შეცვლა ხდება გლობალურად ყველა კატეგორიაზე ერთად

```
a
{
  color: #ff0000;
}
```

ვინაიდან უკვე აღვნიშნეთ, რომ ბმული ბრაუზერისთვის არსებობს კატეგორიზირებული სახით, აქედან გამომდინარე თითოეულ მათგანს რომ მივმართოთ საჭიროა გამოვიყენოთ ქვესელექტორები. მაგალითად, თუ ჩვენ ვაპირებთ გარკვეული თვისებები შევუცვალოთ მხოლოდ მომხმარებლის მიერ უკვე ნანახ ბმულს(VISITED LINK), მაშინ მასთან მიმართვის სინტაქსი შემდეგნაირი იქნება:

```
a:visited
{
  color: #ff0000;
}
```

მოცემული კოდი უთითებს ბრაუზერს, რომ ლინქი რომელიც ვიზიტორს უკვე რეალიზებული აქვს ანუ ნანახი მისთვის ბრაუზერი ამ ბმულის ფერს ასახავს წითელი შრიფტის სახით.

HTML ენისაგან განსხვავებით CSS ის საშუალებით შეგვიძლია აგრეთვე ვმართოთ ბმული მაუსის კურსორის მიტანის მომენტში, ანუ ვცვალოთ მისი თვისებები. მაგალითად

```
a:hover
{
  color: #ff0000;
 background-color:#00ff00;
 font-size:20px;
}
```

მოყვანილი მაგალითის თანახმად, ბმულზე, რომელზეც მომხმარებელი მიიტანს მაუსის კურსორს, აღნიშნული ბმულის ტექსტის ფერი გახდება წითელი, ფონის ფერი გაუხდება მწვანე და შრიფტის ზომა აისახება 20 პიქსელის ზომით.

ახლა კი ვნახოთ სრული მაგალითი თუ როგორ მივაკითხოთ შესაბამისი ქვესელექტორების მეშვეობით ყველა კატეგორიის ბმულს

```
a:link /* ბმული რომელზეც ჯერ არ დაუწკაპუნებია მომხმარებელს  */
{
  color: #ff0000;
}
a:active /* ბმული რომელიც აქტიურია */
{
  color: #ff0000;
}
a:visited /* ბმული რომელიც უკვე დაათვალიერა მომხმარებელმა და ამ ისტორიას ინახავს ბრაუზერი  */
{
  color: #ff0000;
}

a:hover /* ბმული რომელზეც მაუსის მიტანისას უნდა შეიცვალოს მისი თვისება  */
{
  color: #ff0000;
}
```

*კითხვები თვით შეფასებისთვის:*

- დასაშვებია თუ არა გამოვიყენოთ საიტზე არასტანდარტული შრიფტები?
- რამდენი ნაირსახეობის ბმული არსებობს HTML დოკუმენტში?
- როგორ შეიძლება ბმულზე მაუსის კურსორის მიტანისას შევცვალოთ შრიფტის თვისებები?
- ჩამოთვალეთ დინამიურად გენერირებადი შრიფტების ფორმატები.

## 6.3. ვებსაიტის  ელემენტების გაფორმება

**მიმდინარე პარაგრაფის თემატიკა**

- ხაზოვანი და ბლოკური ელემენტების გამოყენების თავისებურებები
- ხაზოვანი და ბლოკური ელემენტების თვისებები
- სიის თვისებები და მნიშვნელობები

### ხაზოვანი და ბლოკური ელემენტების გამოყენების თავისებურებები

მოგეხსენებათ HTML დოკუმენტის სტრუქტურაში ერთმანეთისგან მკაცრად არიან გამიჯნულები ხაზის დონის ელემენტები(ხაზოვანი INLINE ELEMENTS) და ბლოკის დონის ელემენტები (ბლოკური BLOCK LEVEL ELEMNTS). ამ ორი ჯგუფის ტიპის ტაგებს გააჩნიათ საკუთარი თვისებები. მოდით აქვე აღვნიშნოთ ამ თუ რა პრინციპული განსხვავება არის ბლოკურ და ხაზოვან ელემენტებს შორის.

ხაზოვანი ელემენტი (INLINE ELEMENT) დოკუმენტზე აისახება როგორც მიმდინარე ხაზის ელემენტი, ანუ რამდენიმე ხაზოვანი ელემენტის გამოყენების შემთხვევაში ჩვენ მივიღებთ მიმდინარე ხაზში ჩამწკრივებული ელემენტების კასკადს. მაგალითად:

```
<!DOCTYPE html>
<html>
<head>
  <title>დოკუმენტის ტიტული</title>
</head>

<body>
 <b>ხაზოვანი ელემენტი მსხვილი შრიფტით</b>
 <i>ხაზოვანი ელემენტი დახრილი შრიფტით</i>
 <u>ხაზოვანი ელემენტი ხაზგასმული შრიფტით</u>
</body>
</html>
```

მოცემული სამივე ელემენტი აისახება ერთმანეთის გვერდიგვერდ ერთ ჰორიზონტალურ სივრცეში (ხაზში). სწორედ ამიტომაც მსგავს ელემენტებს უწოდეს ხაზოვანი ტიპის ელემენტები.

**ხაზოვანი ელემენტი მსხვილი შრიფტით** *ხაზოვანი ელემენტი დახრილი შრიფტით* ხაზოვანი ელემენტი ხაზგასმული შრიფტით

სურ. 3.14. ხაზოვანი ელემენტების გამოყენების ნიმუში

რაც შეეხება ბლოკური ტიპის ელემენტებს (BLOCK LEVEL ELEMENTS) მათ პრინციპულად სხვა თვისება გააჩნიათ, ანუ თუ ჩვენი სურვილის მიხედვით არ მივუთითეთ მათ გარკვეული CSS თვისებები და მათი მნიშვნელობები ავტომატურად ისინი განლაგდებიან დოკუმენტზე რა თქმა უნდა თანმიმდევრობით, მხოლოდ ერთმანეთის ქვეშ ახალ ახალი ხაზიდან. ანუ თითოეული ბლოკური დონის ელემენტი დოკუმენტზე მოიცავს დოკუმენტის სიგანის მთელ ჰორიზონტალურ სივრცეს, როგორც ბლოკს. ამიტომაც ეწოდებათ მათ ბლოკური დონის ელემენტები. როგორც წესი ბლოკური დონის

ელემენტებს ხაზოვანი ელემენტებისგან განსხვავებით ვიყენებთ ხოლმე დოკუმენტის სტრუქტურულად აგებისთვის ვინაიდან ჩვენი სურვილის მიხედვით სწორად გავანაწილოთ დოკუმენტზე ელემენტები და მოქნილად ვმართოთ მათი განლაგება ურთიერთ მიმართ.

```
<!DOCTYPE html>
<html>
<head>
  <title>დოკუმენტის ტიტული</title>
</head>
<style type="text/css">
 #div1
  {
    background-color:#0000ff;
  }
#div2
  {
    background-color:#00ff00;
  }
</style>
<body>
 <div id="div1">პირველი ბლოკური ტიპის ელემენტი</div>
 <div id="div2">მეორე ბლოკური ტიპის ელემენტი</div>

</body>
</html>
```

პირველი ბლოკური ტიპის ელემენტი

მეორე ბლოკური ტიპის ელემენტი

სურ. 3.15. ბლოკური ტიპის ელემენტების გამოყენების ნიმუში

### ხაზოვანი და ბლოკური ელემენტების თვისებები

ხაზოვანი ტიპის ელემენტები თავის თვისებებით ვერ შეიძლება მოვაქციოთ ერთ ჯგუფში, ვინაიდან მათ ყველას თავისი სპეციფიკის მიხედვით ერთმანეთისგან განსხვავებული თვისებები გააჩნიათ. მაგალითად ხაზოვანი ტიპის ელემენტია ტაგი <img /> ანუ გრაფიკული გამოსახულება, რომლის თვისებებიც რადიკალურად განსხვავდება ასეთივე ხაზოვანი ტიპის ტაგისაგან <b>.

რაც შეეხება ბლოკური ტიპის ელემენტებს მათ გააჩნიათ საერთო თვისებები, როგორიც გახლავთ, ბლოკის სიგანე, სიმაღლე, ფონის ფერი და ასე შემდეგ. რა თქმა უნდა ქვემოთ ყველა ამ თვისებას განვიხილავთ დეტალურად.

| თვისება | მნიშვნელობა | აღწერა |
|---|---|---|
| Background-color | თექვსმეტობითი ათვლის სისტემით მინიჭებული მნიშვნელობა, RGB (Red Green Blue) მნიშვნელობა და ფერის სიტყვიერი წარმოდგენა | აღნიშნული თვისების მნიშვნელობით შეგვიძლია შევცვალოთ ბლოკური ტიპის ელემენტისთვის ფონის ფერი |
| Border-color | თექვსმეტობითი ათვლის სისტემით მინიჭებული მნიშვნელობა, RGB (Red Green Blue) მნიშვნელობა და ფერის სიტყვიერი წარმოდგენა | აღნიშნული თვისების მნიშვნელობით შეგვიძლია შევცვალოთ ბლოკური ტიპის ელემენტისთვის ჩარჩოს ფერი |
| Border-style | None, dotted, dashed, solid, double, groove, ridge, inset, outset | აღნიშნული თვისების მნიშვნელობით შეგვიძლია შევცვალოთ ბლოკური ტიპის ელემენტისთვის ჩარჩოს სტილი, ანუ მისი სხვადასხვა სტილით წარმოდგენა |
| Border-size | მნიშვნელობა საზომ ერთეულებში (px, cm) და ასე შემდეგ | აღნიშნული თვისების მნიშვნელობით შეგვიძლია შევცვალოთ ბლოკური ტიპის ელემენტისთვის ჩარჩოს სისქე |
| Width | მნიშვნელობა საზომ ერთეულებში (px, cm) და ასე შემდეგ | გამოიყენება ბლოკის დონის ტაგებთან და განსაზღვრავს მის სიგანეს |
| Height | მნიშვნელობა საზომ ერთეულებში (px, cm) და ასე შემდეგ | გამოიყენება ბლოკის დონის ტაგებთან და განსაზღვრავს მის სიმაღლეს |
| Position | Absolute, relative მნიშვნელობა საზომ ერთეულებში(px, cm) და ასე შემდეგ<br><br>(top, left, right, bottom) | გადავცემთ ბლოკის დონის ელემენტს თუ როგორ უნდა განთავსდეს დოკუმენტზე, როგორც აბსოლუტური პოზიციის მქონე ელემენტი თუ როგორც ფარდობით. |
| z-index | მთელი რიცხვითი მნიშვნელობა | მივუთითებთ მრავალშრიანი დოკუმენტის არსებობის შემთხვევაში თუ რომელ შრეზე განთავსდეს არსებული ბლოკის დონის ელემენტი. |
| Float | Left, right | გადასცემს ბლოკის დონის ელემენტს თუ რომელი მხრიდან |

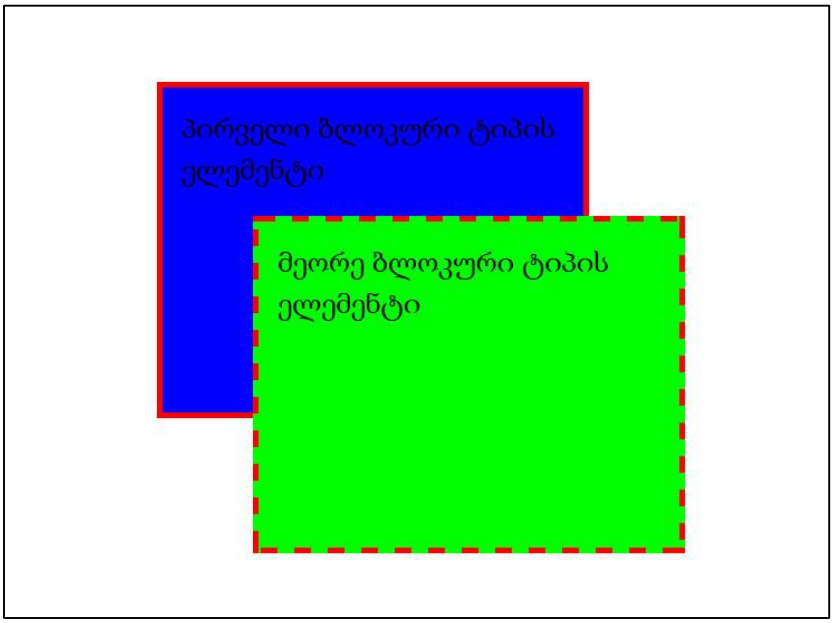
| | | |
|---|---|---|
| | | განთავსდეს მომდევნო ელემენტის მიმართ |
| Overflow | Visible, hidden, scroll, auto | განსაზღვრავს ბლოკური ტიპის ელემენტის მიდამოში, როგორ აისახოს მისი შიგთავსი. |
| Margin | მნიშვნელობა საზომ ერთეულებში(px, cm) და ასე შემდეგ და აგრეთვე შეგვიძლია მივანიჭოთ მნიშვნელობა auto რომელიც უზრუნველყოფს ბლოკის დონის ელემენტის ცენტრში გასწორებას ჰორიზონტალურ სიბრტყეზე. | განსაზღვრავ ბლოკური ტიპის ელემენტის მინდვრებს მის გარშემო, ანუ რამდენად უნდა იყოს ის დაშორებული სხვა ელემენტებისგან |
| Padding | მნიშვნელობა საზომ ერთეულებში(px, cm) და ასე შემდეგ | განსაზღვრავს ბლოკური ტიპის ელემენტის შიგთავსის დაშორებას მისი კიდეებიდან |
| Display | None, block, inline | განსაზღვრავს თუ როგორ უნდა აისახოს ბლოკის დონის ელემენტი, იყოს დამალული, ჩანდეს დოკუმენტზე თუ აისახოს როგორც ხაზოვანი ელემენტი. |

ზემოთ მოყვანილი ცხრილიდან ჩანს თუ რა თვისებები გააჩნიათ ბლოკის დონის ელემენტებს, მიუხედავად ამისა მაინც გვინდა ყურადღება გავამახვილოთ მათ რამდენიმე თვისებაზე რომელთა გამოყენებასაც გარკვეული ნიუანსების გათვალისწინება სჭირდება.

**თვისება position:** რომელსაც ენიჭება მნიშვნელობები, absolute და relative, ანუ დოკუმენტზე ყველა ელემენტი გაჩუმების პრინციპით (Default) პოზიცია იკავებს მის წინა ელემენტთან ფარდობითად. თუმცა ჩვენ შეგვიძლია ბლოკის ტიპის ელემენტს მივუთითოთ, რომ ის მიუხედავად ყველაფრისა განთავსდეს კონკრეტულ აბსოლუტურ პოზიციაზე. ასეთ შემთხვევაში მას უნდა გადავცეთ კიდევ დამატებითი ინსტრუქციები, თუ რომელი პოზიცია დაიკავოს მან. ანუ ხსენებულ შემთხვევაში ჩვენ ვაკონკრეტებთ ბლოკის ტიპის ელემენტს თუ ბრაუზერის ფანჯრის რომელი კიდის მიმართ განთავსდეს გარკვეულ პოზიციაზე :top (ზედა კიდე), left (მარცხენა კიდე), right (მარჯვენა კიდე), bottom (ქვედა კიდე)

**თვისება z-index:** ხშირად საჭიროება მოითხოვს ხოლმე დოკუმენტზე რამდენიმე ელემენტი განთავსებული იყოს შროვანი განლაგებით, ანუ რომელიღაც ელემენტი ან ელემენტის გარკვეული ნაწილი ამოფარებული იყოს მასზე მაღალი პრიორიტეტის მქონე ელემენტს, ან პირიქით. ასეთ შემთხვევაში თამამად შეგვიძლია მოვიშველიოთ თვისება z-index , სწორედ ეს თვისება განსაზღვრავს ელემენტების განლაგებას შრეებად.

```
<!DOCTYPE html>
<html>
<head>
  <title>დოკუმენტის ტიტული</title>
</head>
<style type="text/css">
 #div1
  {
   position:absolute;
   top:50px;
   left:70px;
   height:150px;
   width:200px;
   border-style:solid;
   border-color:#ff0000;
   border-width:3px;
   background-color:#0000ff;
   margin:20px;
   padding:10px;
   z-index:1;
  }
#div2
  {
   position:absolute;
   top:120px;
   left:120px;
   height:150px;
   width:200px;
   border-style:dashed;
   border-color:#ff0000;
   background-color:#00ff00;
   margin:20px;
   padding:10px;
   z-index:2;
  }
</style>
<body>
 <div id="div1">პირველი ბლოკური ტიპის ელემენტი</div>
 <div id="div2">მეორე ბლოკური ტიპის ელემენტი</div>

</body>
</html>
```




სურ. 3.16.ბლოკური ტიპის ელემენტის გაფართოებული თვისებების გამოყენების ნიმუში

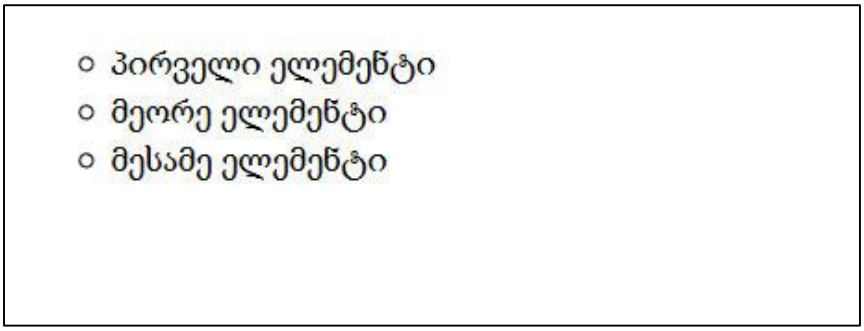
**კითხვები თვით შეფასებისთვის:**

- რა სხვაობაა ხაზოვანსა და ბლოკურ ტიპის ელემენტებს შორის?
- შეგვიძლია თუ არა ბლოკური ტიპის ელემენტი გამოვიყენოთ, როგორც ხაზოვანი ტიპის ელემენტი?
- ჩამთვალეთ შესაძლებლობები თუ როგორ შეგვიძლია მივანიჭოთ პოზიცია ბლოკური ტიპის ელემენტს
- რომელი თვისება განსაზღვრავს ბლოკური ტიპის ელემენტების, ურთიერთ გადაფარვის შემთხვევაში მათ შრეებად განლაგებას?

## სიის ტიპები და მნიშვნელობები

HTML-დან ჩვენ უკვე ვიცით რომ არსებობს სიის სხვადასხვა ტიპი, მათი გაფორმება ჩვენ შეგვიძლია CSS-ის საშუალებით. რეალურად სიის გაფორმების კარგი საშუალება გვაქვს, რომ იგი გავაფორმოთ გრაფიკული მარკირებით ნაცვლად სტანდარტული მარკირების ტიპისა. გარდა ამისა CSS -ი შესაძლებლობას გვაძლევს ასევე ვმართოთ სიის სტანდარტული მარკირების ტიპები. სიის მარკირების მართვის სინტაქსი CSS-ის მეშვეობით შემდეგნაირია:

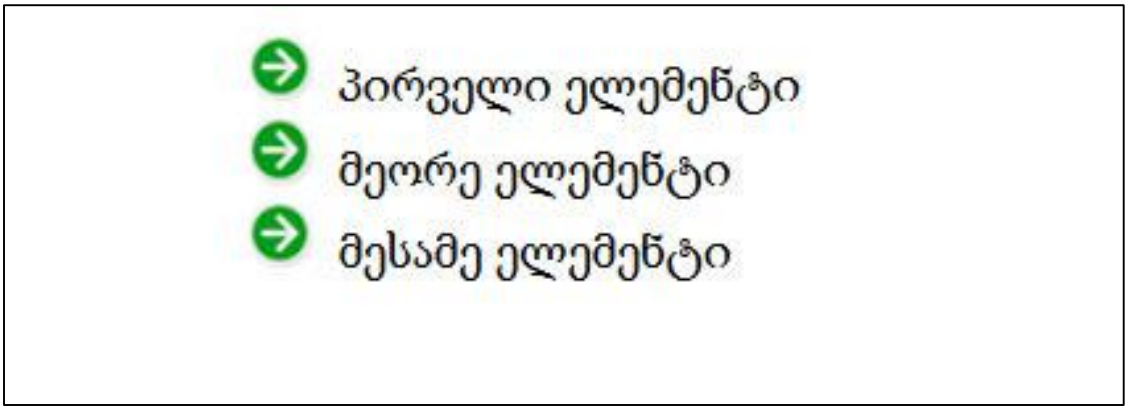
```
<!DOCTYPE html>
<html>
<head>
  <title>დოკუმენტის ტიტული</title>
</head>
<style type="text/css">

ul
{
 list-style-type: circle;
}

</style>

<body>

 <ul>
  <li>პირველი ელემენტი</li>
  <li>მეორე ელემენტი</li>
  <li>მესამე ელემენტი</li>
</ul>

</body>
</html>
```




სურ. 3.17.მარკირებული სიის გამოყენების ნიმუში

როგორც მოგახსენეთ სიის მარკირების ტიპი ასევე შეძლება იყოს გრაფიკული გამოსახულება. ქვემოთ მოცემულ მაგალითში CSS-ის მეშვეობით ჩვეულებრივი HTML სია გაფორმებული გვაქვს გრაფიკული მარკერებით. მთავარი არის, რომ CSS-ში სწორად მივუთითოთ გზა ( URL მისამართი) იმ გრაფიკული გამოსახულებისა რომელმაც უნდა გააფორმოს ჩვენს მიერ შექმნილი სია.

```
<!DOCTYPE html>
<html>
<head>
  <title>დოკუმენტის ტიტული</title>
</head>
<style type="text/css">

ul
{
 list-style-image: url("listicon.gif");
}

</style>

<body>

 <ul>
  <li>პირველი ელემენტი</li>
  <li>მეორე ელემენტი</li>
  <li>მესამე ელემენტი</li>
</ul>

</body>
</html>
```




სურ. 3.18.გრაფიკული მარკერებით გაფორმებული სიის გამოყენების ნიმუში

*კითხვები თვით შეფასებისთვის:*

- რამდენნაირი სახეობის სია არსებობს HTML - ში?
- რამდენად შესაძლებელია სიის გრაფიკულად გაფორმება?
- რა ტიპის მარკირება შეგვიძლია მივანიჭოთ სიებს?

## 6.4. შექმნილი ნამუშევრის ხარისხის უზრუნველყოფა W3C სტანდარტების შესაბამისად

**მიმდინარე პარაგრაფის თემატიკა**

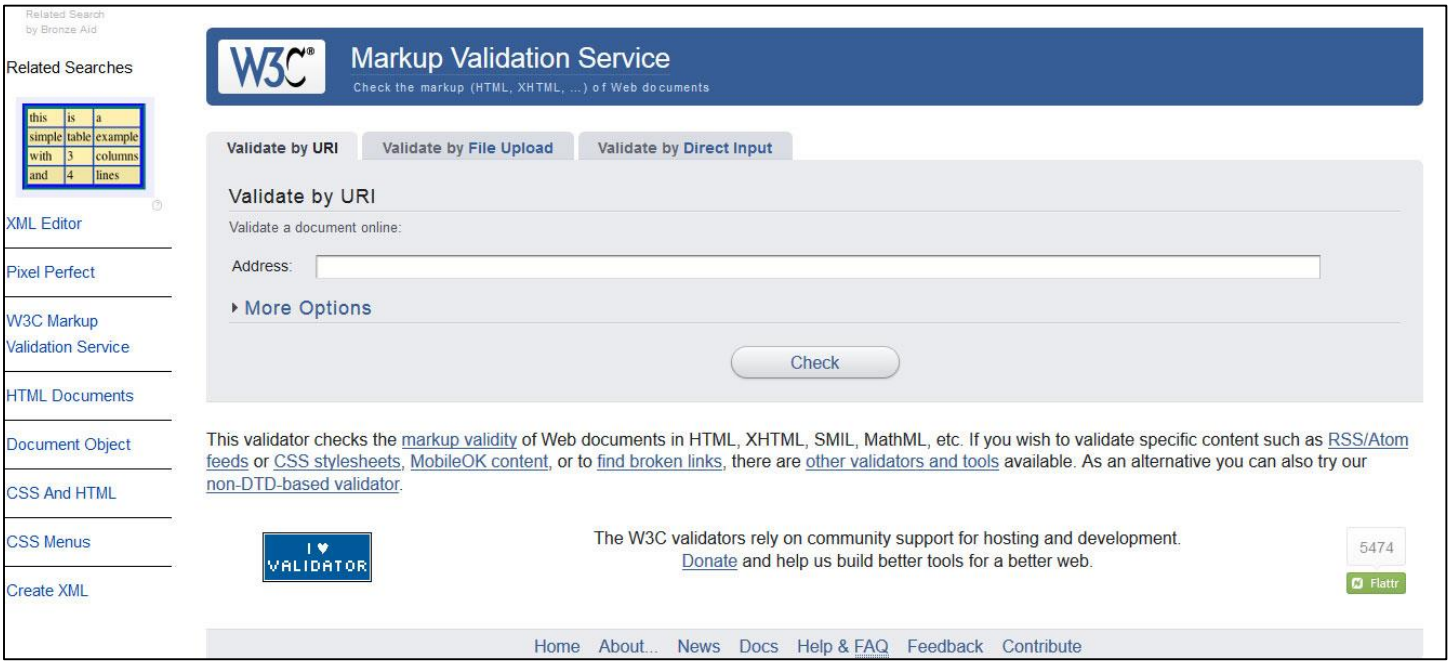
- W3C სტანდარტი და ვალიდაციის ინსტრუმენტი

ახლა შევეხოთ ისეთ თემას როგორიც არის WEB გვერდის (რესურსის) სტანდარტიზაცია. რას ნიშნავს სტანდარტიზაცია და ვალიდური WEB პროექტი? მოგეხსენებათ ინტერნეტ სივრცეში ნებისმიერი WEB გვერდის დათვალიერება ხორციელდება ბრაუზერების მეშვეობით. ვინაიდან წამყვანი ბრაუზერები, როგორებიც არიან Firefox, Google Chrome, Internet Explorer, Opera და ასე შემდეგ ერთიდაიგივე დოკუმენტს, ანუ მასში გაწერილ კოდს, იქნება ეს HTML თუ CSS კოდი აღიქვამენ ერთმანეთისგან უმნიშვნელოდ მაგრამ მაინც განსხვავებულად. სწორედ ამ მიზეზის გამო ორგანიზაციამ W3C (WORLD WIDE WEB CONSORTIUM) -მა ანუ ორგანიზაციამ რომელიც ქმნის ინტერნეტ სტანდარტებს გადაწყვიტა WEB კოდირების ენები მოექცია გარკვეულ ერთ სტანდარტში. ანუ თუ ჩვენ მივყვებით W3C კონსორციუმის სტანდარტებს და ჩვენს პროექტებს ვქნით მათ მიერ დადგენილი სტანდარტის მიხედვით გამოვა, რომ ჩვენი საიტი აბსოლუტურად გამართულად იმუშავებს ყველა ზემოთ ჩამოთვლილ ბრაუზერში. აქვე აღვნიშნოთ, რომ მსგავსი პროექტის შექმნა და ყველა ბრაუზერზე მორგება საკმაოდ

შრომატევადი საქმეა, ვინაიდან პროექტის ფორმირების პროცესში უნდა გავითვალისწინოთ ის გარემოება, როგორიცაა ელემენტარულად ჩვენი ყველა WEB დოკუმენტის სხვადასხვა ბრაუზერში ტესტირება. სწორედ ამ პრობლემის გადასაწყვეტად შეგვიძლია მივმართოთ W3C კონსორციუმის ასე ვთქვათ WEB გვერდის ვალიდატორს. რა არის ვალიდატორი? ვალიდატორი გახლავთ ონლაინ პროგრამული უზრუნველყოფა, რომელსაც შეუძლია ჩვენს მიერ შექმნილი პროექტის კოდი შეამოწმოს უკანასკნელი სტანდარტის მიხედვით ვალიდურობაზე.

იმისათვის რომ დავრწმუნდეთ არის თუ არა ჩვენს მიერ დაწერილი კოდი ვალიდური და შეესაბამება თუ არა ის თანამედროვე სტანდარტებს, პირველ რიგში საჭიროა ვეწვიოთ W3C კონსორციუმის ოფიციალურ ვებ-გვერდს შემდეგ ინტერნეტ მისამართზე : https://validator.w3.org სწორედ აღნიშნულ ვებ-მისამართზე გახლავთ განთავსებული ის ინსტრუმენტი (ვალიდატორი), რომლის შესახებაც ზემოთ გვქონდა საუბარი.

აღნიშნული ვებ-გვერდის ვიზიტისას იხილავთ შემდეგნაირ ინტერფეისს:



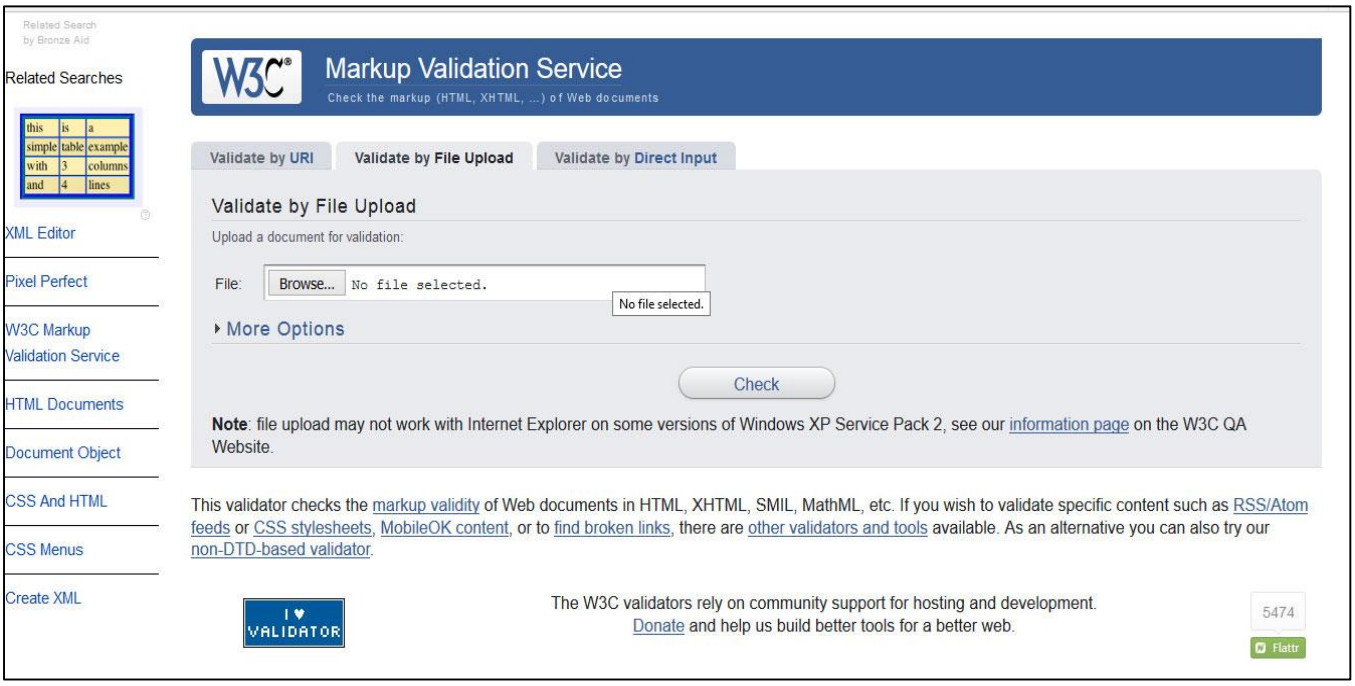

სურ. 3.19.W3C-ს ვალიდაციის ინსტრუმენტის გამოყენების ნიმუში

აღნიშნული ვალიდატორის ინტერფეისში შენიშნავთ, რომ გაგაჩნიათ თქვენი კოდის ვალიდურობაზე შემოწმების სამი გზა. ეს გახლავთ

1. Validate by URI
2. Validate by File Upload
3. Validate by Direct Input

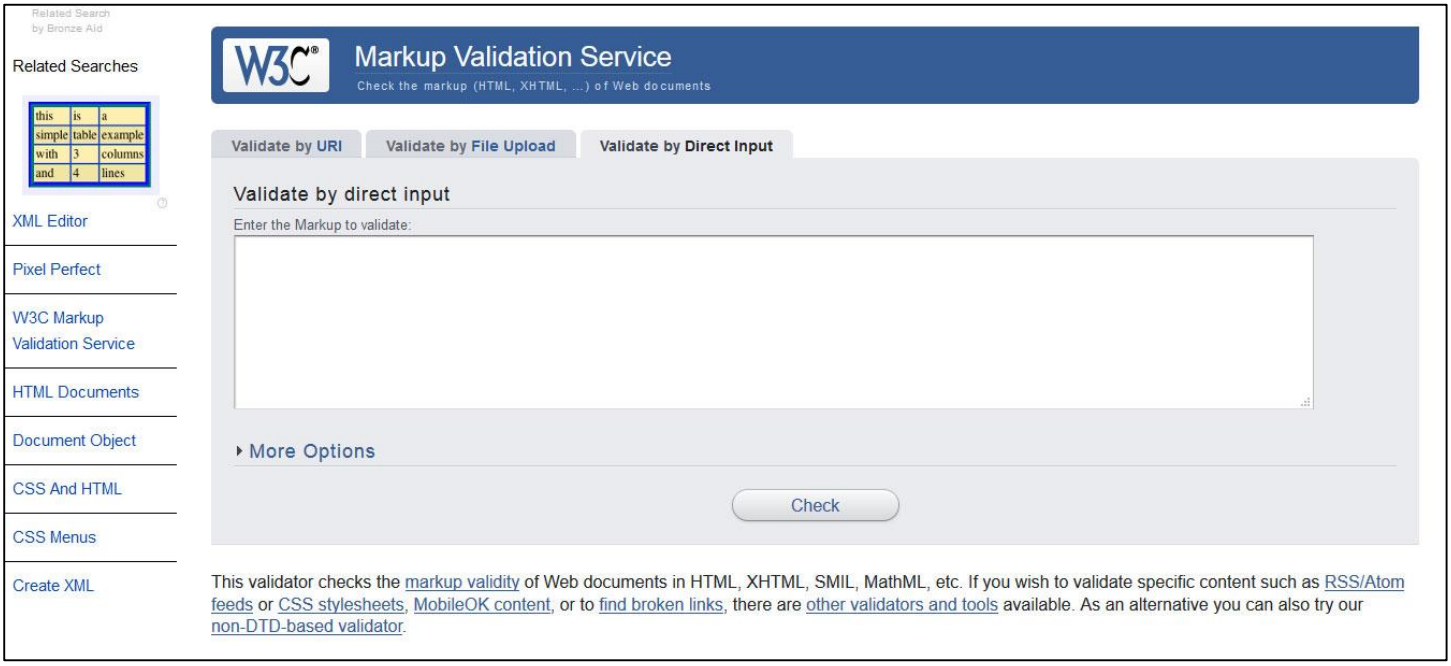
პირველი ვარიანტი, ანუ Validate by URI კოდის ვალიდურობის შესამოწმებლად გამოგადგებათ მხოლოდ მაშინ, როდესაც თქვენი პროექტი ანუ კოდები უკვე გარკვეულ ინტერნეტ სერვერზეა განთავსებული და გააჩნიათ ინტერნეტ მისმართი. ასეთ შემთხვევაში ვალიდატორის Address-ის ველში მიუთითებთ სწორედ ამ მისამართ და ვალიდატორიც შეამოწმებს თქვენს ფაილს ვალიდურობაზე.

Validate by File Upload, ეს მეთოდი მისი სათაურით უკვე თავისთავად მეტყველებს თავის თავზე. ანუ აირჩევთ ნებისმიერ სავალიდაციოდ გამზადებულ ფაილს, ატვირთავთ ვალიდატორის მეშვეობით და ვალიდატორი დაგიბრუნებთ პასუხს, ვალიდურია თუ არა თქვენი შექმნილი კოდი.




სურ. 3.20.W3C-ს ვალიდაციის ინსტრუმენტის გამოყენების ნიმუში

და ბოლოს, ყველაზე პრაქტიკულად გამოსაყენებელი ვარიანტი ეს გახლავთ მესამე ვარიანტი. Validate by Direct Input, ანუ ამ შემთხვევაში ყველაზე მარტივად , რედაქტორიდან, სადაც ვქმნით ჩვენს კოდს ვაკოპირებთ კოდის ტექსტს, ვსვამთ ვალიდატორის ფანჯარაში და ვღებულობთ პასუხს თუ რამდენად ვალიდურია ჩვენი კოდი თანამედროვე სტანდარტებთან მიმართებაში.




სურ. 3.21.W3C-ს ვალიდაციის ინსტრუმენტის გამოყენების ნიმუში

*კითხვები თვით შეფასებისათვის:*

- რისთვის არის საჭირო ვალიდაციის ინსტრუმენტის გამოყენება?
- რას ემსახურება ვალიდაციის ინსტრუმენტი?
- რამდენი გზა არსებობს ვალიდაციის ინსტრუმენტის გამოსაყენებლად?

# 7. ვებ მარკირების გაფართოებული შესაძლებლობები (HTML5, CSS3)

## 7.1. ვებსაიტის ძირითადი სტრუქტურის აგება HTML 5-ის ბრძანებების გამოყენებით

**მიმდინარე პარაგრაფის თემატიკა**

- HTML 5 ის მიმოხილვა
- რა სიახლეებია HTML 5 ში
- რომელი ტაგები აღარ გამოიყენება HTML 5 ში
- HTML 5 ის მეშვეობით დოკუმენტის ძირითადი სტრუქტურის შექმნა

### HTML 5-ის მიმოხილვა

პირველ რიგში აღვნიშნოთ ის ფაქტი, რომ HTML 5 გაცილებით მოქნილი არის გამოყენებაში და რაც მთავარია უკვე სტანდარტიზებულია რაც იმას ნიშნავს, რომ მისი აღქმა ბრაუზერებისთვის პრობლემას არ წარმოადგენს. თუმცა აქვე გვინდა მივანიშნოთ, რომ HTML -ის ნებისმიერი ვერსიის გამოყენებისას ყოველთვის აუცილებელია ჩვენს მიერ შექმნილი კოდის ტესტირება სხვადასხვა ბრაუზერში, ვინაიდან პრაქტიკამ აჩვენა ბრაუზერების მიერ HTML კოდის არაერთგვაროვნად აღქმის შესაძლებლობა. გამონაკლისი ამ შემთხვევაში რა თქმა უნდა არც HTML 5 გახლავთ. HTML ის ხსენებულ ვერსიაში საკმაოდ არს ცვლილებები შეტანილი. ჩვენ შევეცდებით აღვწეროთ თუ რა ცვლილებებია მასში, რა დაემატა და რომელი ტაგები იქნა ამოღებული გამოყენებიდან. ყველაზე მნიშვნელოვანი და პრაქტიკაში ადვილად გამოსაყენებელი რაც არის, ეს გახლავთ HTML 5-ის კოდის წერის შემოკლებული სინტაქსი. მაგალითად როგორიცაა დოკუმენტის სტრუქტურის შექმნა, გარკვეული მულტიმედიური და ფორმის ელემენტების გაცილებით მარტივად გამოყენება, გრაფიკასთან მუშაობა და ასე შემდეგ.

### რა სიახლეებია HTML 5 -ში?

პირველ რიგში გავამახვილოთ ყურადღება ისეთ მნიშვნელოვან საკითზე როგორიც გახლავთ HTML დოკუმენტის სტრუქტურის ფორმირება, კერძოდ დოკუმენტის ტიპის დეკლარირება. მაგალითად: თუ HTML 4.01 ვერსიაში დოკუმენტის ტიპის დეკლარირება გვიწევდა შემდეგნაირი სინტაქსით როგორიც არის

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

HTML 5 -ში ეს ამოცანა გაცილებით გამარტივებულია და ჩანაცვლებულია შემდეგი დეკლარირების ტიპით

<!DOCTYPE html>

აქვე ხაზი გავუსვათ იმ ფაქტს, რომ დოკუმენტის ტიპის დეკლარირებისას სიმბოლოების რეგისტრს არავითარი მნიშვნელობა არ გააჩნია ბრაუზერებისთვის. ანუ ზემოთ მოყვანილი ჩანაწერი და მომდევნო ჩანაწერი იდენტურები არიან ბრაუზერებისთვის სტანდარტის მიხედვით

<!DOCTYPE HTML>

ზუსტად იგივე შეგვიძლია აღვნიშნოთ დოკუმენტის ენობრივ კოდირებასთან დაკავშირებით. HTML 5 ის სტანდარტით გაცილებით მარტივად შეგვიძლია დოკუმენტის კოდირების ტიპის მითითება. მაგ:

<meta charset="UTF-8>

თუმცა გვინდა ყურადღება გავამახვილოთ იმ ფაქტზე, რომ როდესაც HTML 5 ის კოდირების დეკლარირებას არ ვახდენთ მაშინ ავტომატურად ანუ გაჩუმების პრინციპით კოდირების ტიპი ბრაუზერების მიერ აღიქმება როგორც UTF-8.

რაც შეეხებოდა დოკუმენტის სტრუქტურის დეკლარირებას ამაზე უკვე ზემოთ ვისაუბრეთ. ახლა შევეხოთ იმ თემას თუ რა პრინციპულად განსხვავებული ელემენტები არის დამატებული HTML 5ის მარკირების ენაში. ესენი გახლავთ:

- **სემანტიკური ტიპის ელემენტები:** <header>, <article>, <section> <footer>.
- **ფორმის ელემენტები:** number, date, time, calendar, range
- **გრაფიკული ტიპის ელემენტები:** <svg>, <canvas>
- **მულტიმედიური ტიპის ელემენტები:** <audio>, <video>.

რა თქმა უნდა ყველა ზემოთ ჩამოთვლილ ელემენტს დეტალურად განვიხილავთ და ვნახავთ მათ თვისებებს პრაქტიკულ მაგალითებზე.

### რომელი ტაგები აღარ გამოიყენება HTML 5 -ში?

როგორც მოგეხსენებათ HTML მარკირების ენა მუდმივ ფორმირებას განიცდის და სულ დამუშავების პროცესში არის. გარკვეული წლების მანძილზე ცდილობდნენ მის სტანდარტიზაციაზე და უნივერსალიზაციაზე. სწორედ ასეთი მცდელობების გამო ხდება მისი ფორმირება. მაგალითად HTML 4 ვერსიაში გადაწყვიტეს გამოეყენებინათ ახალი ტიპის ტექსტის ლოგიკური ფორმატირების ტაგები. ასევე იყო დავა და საუბარი დარჩენილიყო თუ არა სტანდარტის მიხედვით FRAME-ები. HTML 5ის მარკირების ენაში ამ თემატიკასთან დაკავშირებით საკმაოდ მკვეთრი ცვლილებებია შეტანილი, კერძოდ: ამოღებულია გამოყენებიდან შემდეგი ტაგები:

| |
|---|
| <acronym> |
| <applet> |
| <basefont> |
| <big> |
|  |

| |
|---|
| <dir> |
| <font> |
| <frame> |
| <frameset> |
| <noframes> |
| <strike> |
| <tt> |

ქვემოთ მოყვანილია მაგალითი თუ რა მარტივი არის HTML 5 ის მეშვეობით დოკუმენტის სტრუქტურის შექმნა და ძირითადი სტრუქტურის შემადგენელი ელემენტების დეკლარირება:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8
<title>დოკუმენტის ტიტული</title>
</head>
<body>

</body>
</html>
```

სურ. 3.22.HTML 5 -ის დოკუმენტის სტრუქტურის დეკლარირების გამოყენების ნიმუში

*კითხვები თვით შეფასებისთვის:*

- რა ძირეული ცვლილებები არის შეტანილი HTML5 ის ენაში HTML4 ისგან განსხვავებით?
- არსებობს თუ არა ისეთი ტაგები რომლებიც აღარ გამოიყენება HTML5 -ში და თუ არსებობს ჩამოთვალეთ ისინი.
- აუცილებელი არის თუ არა ბრაუზერებისთვის დოკუმენტის სწორად დეკლარირება?
- რა განსხვავებაა HTML5 სა და HTML 4-ს შორის დოკუმენტის დეკლარირების სპეციფიკაში?

## HTML5 ის ბლოკური ტიპის ტაგების განხილვა

როგორც ზემოთ უკვე აღვნიშნეთ HTML5 -ის ვებ მარკირების ენას დაემატა ბლოკური ტიპის ტაგები, ქვემოთ განვიხილავთ შემდეგ ტაგებს:

- <header>
- <article>
- <section>

- <footer>

**ტაგი - <header>**

აღნიშნული ტაგი როგორც წესი გამოიყენება ხოლმე ვებ გვერდზე ზედა ნაწილში ინფორმაციის განსანთავსებლად, მაგალითად ლინქების პანელის ასაწყობად, სათაურების ჩასასმელად ან უბრალოდ ლოგოს შემოსატანად დოკუმენტში. ტაგი <header> დოკუმენტში შეგვიძლია გამოვიყენოთ რამოდენიმეჯერ სხვადასხვა ბლოკების შექმნის მიზნით.

```
<!DOCTYPE html>
<html>
<body>

  <header>
    <h1>პირველი ხარისხის სათაური</h1>
    <h3>მესამე ხარისხის სათაური</h3>
    <p>უბრალოდ აბზაცი</p>
  </header>


</body>
</html>
```

პირველი ხარისხის სათაური

მესამე ხარისხის სათაური

უბრალოდ აბზაცი

სურ. 3.23.ტაგი <header>-ის გამოყენების ნიმუში

**ტაგი -<article>**

მოცემულ ტაგს მისი ლოგიკური სახელიდან გამომდინარე ვიყენებთ ხოლმე გარკვეული ბლოგების, სტატიების, ან კომენტარების ასახვისთვის ვებ გვერდზე.

```
<!DOCTYPE html>
<html>
<body>

<article>
    <p>სტატიის ან ბლოგის ტექსტი</p>
</article>

</body>
</html>
```

სტატიის ან ბლოგის ტექსტი

სურ. 3.24.ტაგი <article>-ის  გამოყენების ნიმუში

**ტაგი - <section>**

ამ ტაგის მეშვეობით დოკუმენტზე შეგვიძლია გამოყოთ გარკვეული სექციები მსგავსად აბზაცებისა. მისი გამოყენება დოკუმენტში შეგვიძლია განუსაზღვრელი რაოდენობით.

```
<!DOCTYPE html>
<html>
<body>

<section>
    <p>დოკუმენტზე არსებული პირველი სექცია გარკვეული ინფორმაციით</p>
</section>

<section>
    <p>დოკუმენტზე არსებული მეორე სექცია გარკვეული ინფორმაციით</p>
</section>


</body>
</html>
```

დოკუმენტზე არსებული პირველი სექცია გარკვეული ინფორმაციით

დოკუმენტზე არსებული მეორე სექცია გარკვეული ინფორმაციით

სურ. 3.25.ტაგი <section>-ის  გამოყენების ნიმუში

**ტაგი - <footer>**

ეს ტაგი ემსახურება დოკუმენტზე განთავსდეს ისეთი ტიპის ინფორმაცია როგორებიცაა მაგალითად:

- ინფორმაცია საავტორო უფლების შესახებ
- კოპი რაიტი
- საკონტაქტო ინფორმაცია
- საიტის რუქა

და ასე შემდეგ.

აღნიშნული ტაგი გამოიყენება აგრეთვე ჩვენს მიერ ზეოთ ხსენებულ ტაგ <section>- თან

```
<!DOCTYPE html>
<html>
<body>

<footer>
  <p>საიტი შექმნილია www.gtu.edu.ge -ს მიერ</p>
  <p>ყველა საავტორო უფლება დაცულია</p>
</footer>

</body>
</html>
```

საიტი შექმნილია www.gtu.edu.ge -ს მიერ

ყველა საავტორო უფლება დაცულია

სურ. 3.26.ტაგი <footer>-ის  გამოყენების ნიმუში

HTML 5 -ის ვებ მარკირების ენაში ასევე დამატებულია გრაფიკასთან მოქნილად სამუშაო ტაგები

- <canvas>
- <svg >

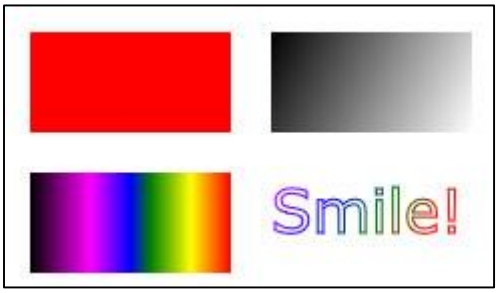
განვიხილოთ თითოეული მათგანი:

**ტაგი - <canvas>**

ის წარმოადგენს  ვებ დოკუმენტზე გრაფიკული გამოსახულების დინამიურად შექმნის საშუალებას, რომელიმე სკრიპტის ენით. როგორც წესი JavaScript-ის მეშვეობით.

იმისათვის, რომ შეგექმნათ ზოგადი წარმოდგენა ამ ტაგის შესაძლებლობების შესახებ მოგვყავს პატარა მაგალითი.

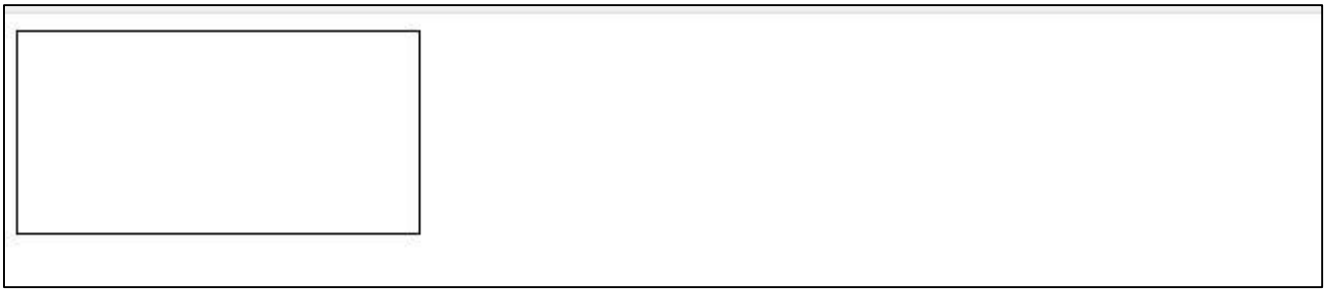
ქვემოთ ასახული ოთხივე გრაფიკული გამოსახულება შექმნილია დინამიურად ტაგი <canvas> ის საშუალებით.



სურ. 3.27.ტაგი <canvas>-ის  გამოყენების ნიმუში

ტაგი <canvas> დოკუმენტზე წარმოადგენს ოთხკუთხედ ელემენტს, რომელსაც გაჩუმების პრინციპით არ გააჩნია ჩარჩო, არ არის ბლოკური ტიპის ტაგი და შეგვიძლია ვუცვალოთ სიგანე და სიმაღლე.

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
თქვენს ბრაუზერს არ გააჩნია ტაგი canvas-ის მხარდაჭერა.
</canvas>

</body>
</html>
```

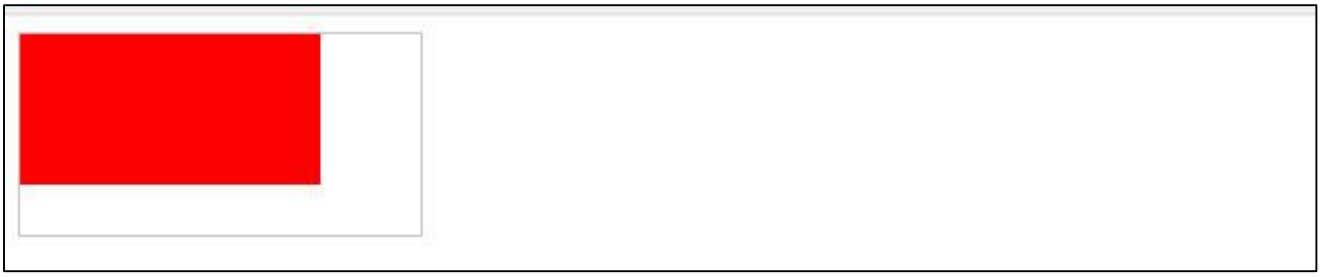


სურ. 3.28.ტაგი <canvas>-ის  გამოყენების ნიმუში შესაბამისი თვისებებით

არსებული ტაგი თავისთავად შიგთავსის გარეშე აისახება, იმისათვის , რომ გრაფიკული შიგთავსი ავსახოთ მის კონტენტად აუცილებელია გამოვიყენოთ JavaScrip-ის ენა.

ქვემოთ მოყვანილ უმარტივეს მაგალითში დემონსტრირებულია თუ როგორ შეიძლება JavaScript-ის მეშვეობით  canvas-ის შიგთავსში გეომეტრიული ფიგურის კონკრეტულად ოთხკუთხედი ჩახატვა.

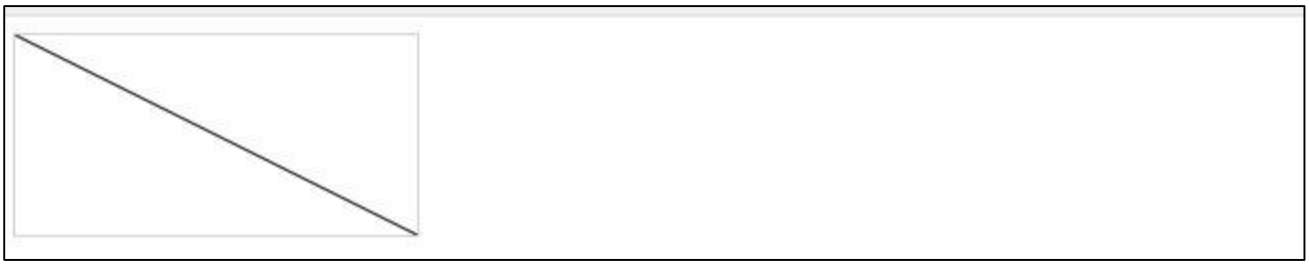
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100"style="border:1px solid #c3c3c3;">
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
</script>

</body>
</html>
```



სურ. 3.29.ტაგი <canvas>-ის  გამოყენების ნიმუში შესაბამისი თვისებებით

მაგალითი თუ როგორ შეგვიძლია ჩავხატოთ ხაზი.

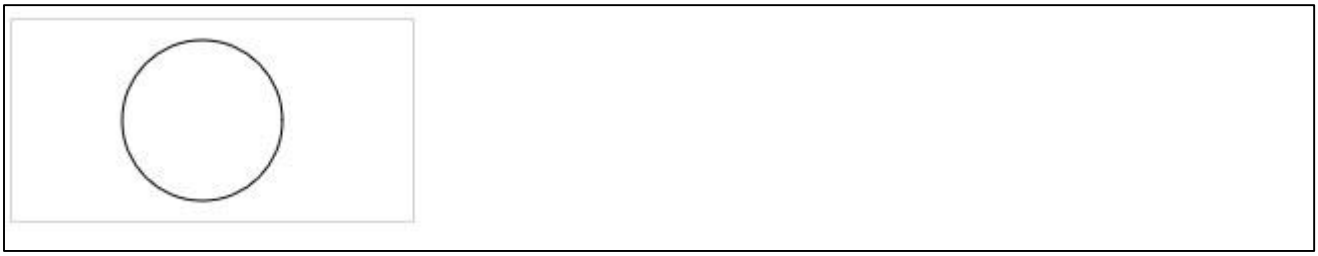
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>

</body>
</html>
```



სურ. 3.30.ტაგი <canvas>-ის მეშვეობით ხაზის გავლების გამოყენების ნიმუში

წრეწირის  ჩახატვა:

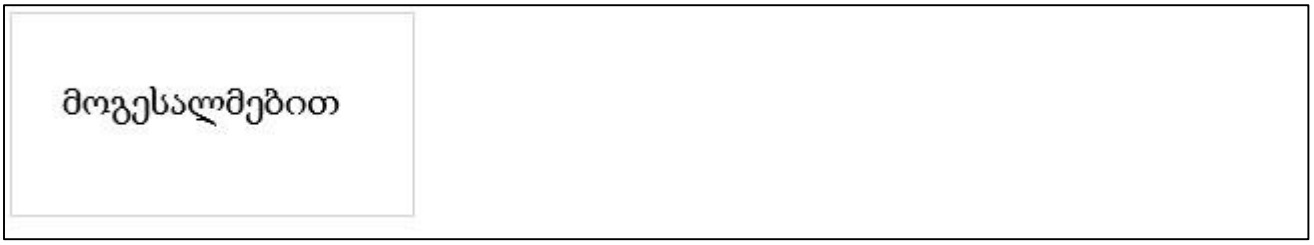
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```



სურ. 3.31.ტაგი <canvas>-ის მეშვეობით ხაზის წრეწირის ჩახატვის გამოყენების ნიმუში

ტექსტის ჩახატვა:

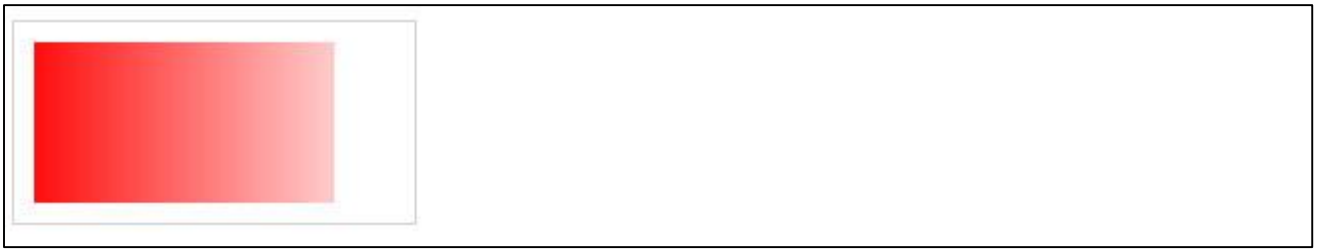
```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "20px Sylfaen";
ctx.fillText("მოგესალმებით",25,50);
</script>

</body>
</html>
```




სურ. 3.32.ტაგი <canvas>-ის მეშვეობით ტექსტის ფორმირების გამოყენების ნიმუში

გრადაციის შექმნა:

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// ქმნის გრადიენტს
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
// აფერადებს ფიგურას გრადაციული ფერით
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>

</body>
</html>
```



სურ. 3.33.ტაგი <canvas>-ის მეშვეობით გრადაციული ფონის გამოყენების ნიმუში

**ტაგი - <svg>** (Scalable Vector Graphics) მასშტაბირებადი ვექტორული გრაფიკა.

აღნიშნული ტაგი გამოიყენება ვებ გვერდზე გრაფიკული კომპონენტების შესაქმნელად მსგავსად ტაგი <canvas>-ისა. მისი მეშვეობით შეგვიძლია შევქმნათ გეომეტრიული ფიგურები, გრაფიკული გამოსახულებები და ტექსტები ჩვენი ვებ გვერდისთვის.

მოდით გავაკეთოთ ახლა შედარება რით განსხვავდებიან ერთმანეთისგან ტაგები <svg> და <canvas>.

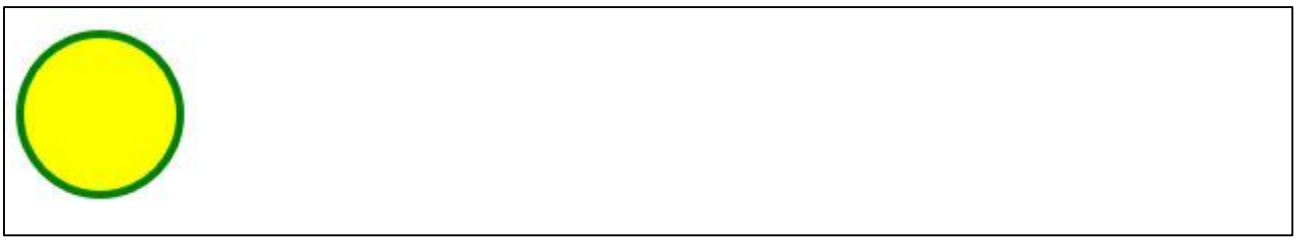
- SVG არის ენა, რომელიც აღწერს 2D განზომილებიან გრაფიკას XML-ის ფორმატით
- Canvas დინამიურად ხატავს 2D განზომილებიან გრაფიკას და იყენებს JavaScript-ს
- SVG დაფუძნებულია XML-სტრუქტურაზე, რაც იმას ნიშნავს, რომ მისი მეშვეობით შექმნილ ელემენტებთან გვაქვს წვდომა JavaScript-ის მეშვეობით, აგრეთვე გვაქვს საშუალება გამოვიყენოთ JavaScript ხდომილებები (Events).
- SVG ის მიერ შექმნილი გამოსახულება ბრაუზერში ინახება როგორც ობიექტი. მაგალითად თუ დოკუმენტში რომელიმე ობიექტის პოზიციას ვცვლით დინამიურად ის ავტომატურად თავიდან გადაიხატება ახალ პოზიციაზე.
- Canvas-ით შექმნილი ფიგურა იხატება პიქსელ-პიქსელ. Canvas-ის მიერ დახატული ფიგურა ბრაუზერის მიერ უკვე დავიწყებულია, ეს იმას ნიშნავს, რომ მისი მაგალითად პოზიციის ცვლილების შემთხვევაში ფიგურის თავიდან დახატვის ფუნქცია უნდა შესრულდეს.

ქვემოთ მოყვანილ ცხრილში აღწერილია SVG-ს და Canvas- შორის სხვა მნიშვნელოვანი განსხვავებები.

| Canvas | SVG |
|---|---|
| დამოკიდებულია ეკრანის რეზოლუციაზე | არ არის დამოკიდებული ეკრანის რეზოლუციაზე |
| არ აქვს ხდომილებების მხარდაჭერა (Events) | აქვს ხდომილებების მხარდაჭერა (Events) |
| კარგად მორგებულია და მოსახერხებელი გამოყენებადი დინამიურ გრაფიკული თამაშების შესაქმნელად | არ არის მორგებული გრაფიკული თამაშების აპლიკაციებზე |
| გამოსახულების დამუშავების სუსტი შესაძლებლობები | დიდი გამოსახულებების კარგი შესაძლებლობები როგორიცაა მაგალითად Google Map |

ახლა კი ქვემოთ ვიხილოთ პრაქტიკული მაგალითები ტაგი <svg>-ს გამოყენებით.
SVG-ს მეშვეობით დახატული გაფერადებული წრე

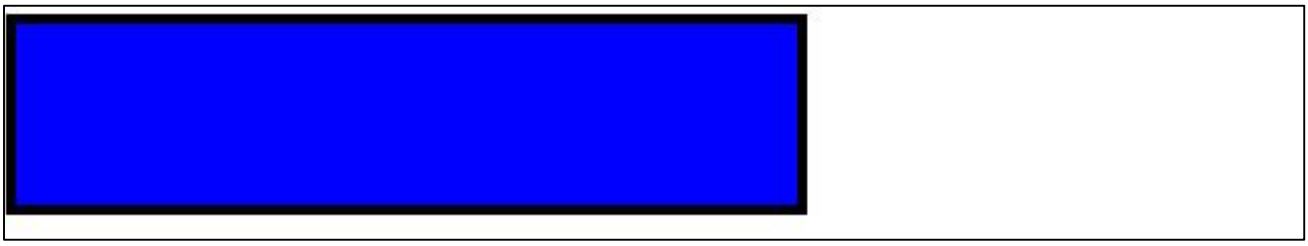
```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```



სურ. 3.34.ტაგი <svg>-ის მეშვეობით გაფერადებული წრის გამოყენების ნიმუში

SVG-ს მეშვეობით დახატული გაფერადებული ოთხკუთხედი

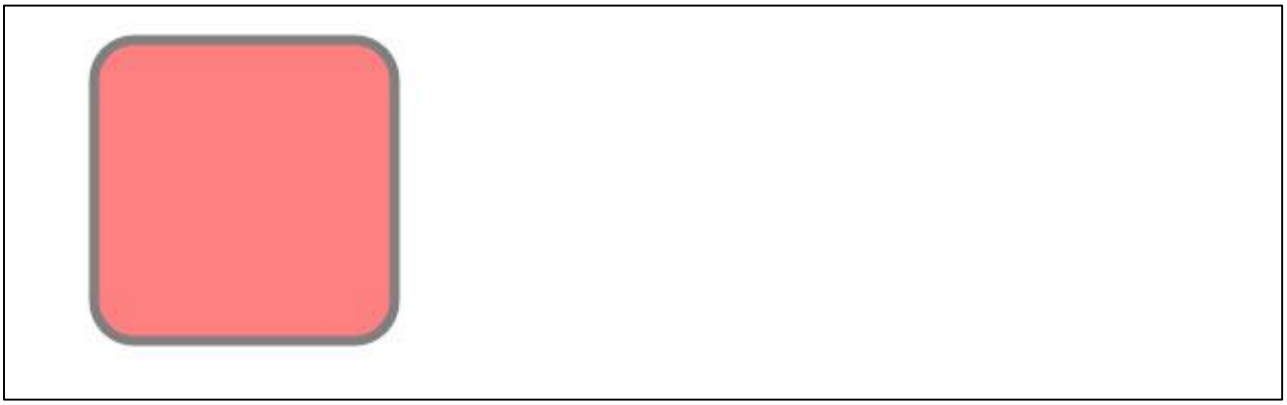
```
<!DOCTYPE html>
<html>
<body>

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
</svg>

</body>
</html>
```



სურ. 3.35.ტაგი <svg>-ის მეშვეობით გაფერადებული ოთხკუთხედი გამოყენების ნიმუში

SVG-ს მეშვეობით დახატული გაფერადებული  ოვალურ კუთხოვანი ოთხკუთხედი

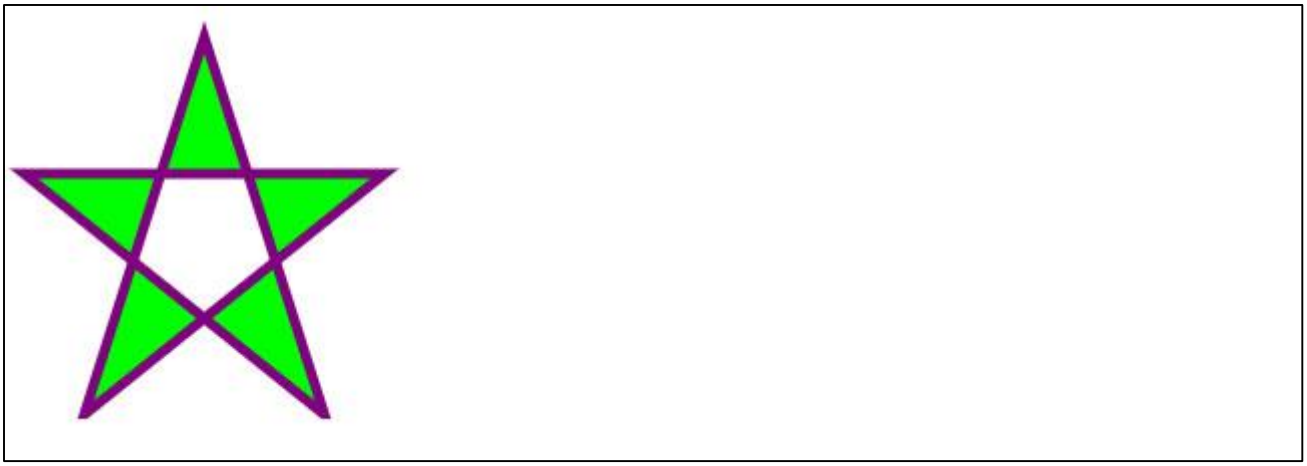
```
<!DOCTYPE html>
<html>
<body>

<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>

</body>
</html>
```



სურ. 3.36.ტაგი <svg>-ის მეშვეობით გაფერადებული ოვალურ კუთხეებიანი ოთხკუთხედის გამოყენების ნიმუში

SVG-ს მეშვეობით დახატული ნაწილობრივ გაფერადებული მრავალკუთხედი

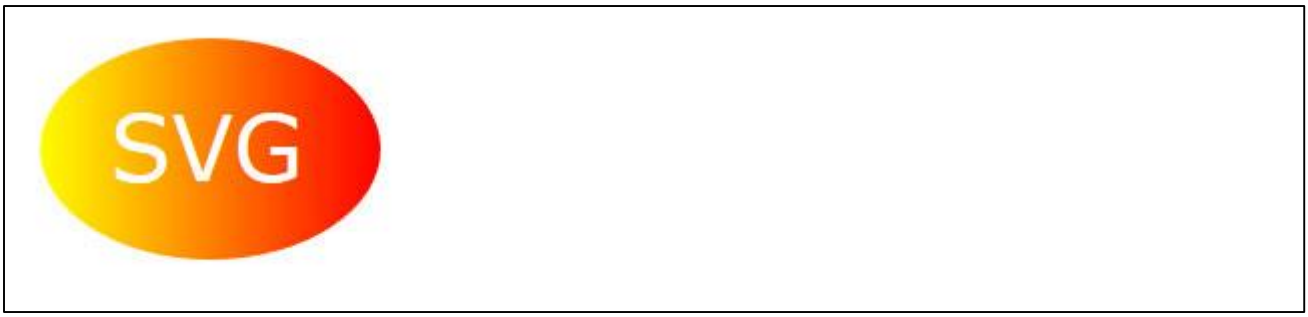
```
<!DOCTYPE html>
<html>
<body>

<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>

</body>
</html>
```



სურ. 3.37.ტაგი <svg>-ის მეშვეობით მრავალკუთხედის გამოყენების ნიმუში

SVG-ს მეშვეობით დახატული მარტივი ლოგო.

```
<!DOCTYPE html>
<html>
<body>

<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%"
      style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%"
      style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana"
  x="50" y="86">SVG</text>
</svg>

</body>
</html>
```




სურ. 3.34.ტაგი <svg>-ის მეშვეობით მარტივი ლოგოს შექმნის ნიმუში

*კითხვები თვით შეფასებისთვის:*

- რომელი ახალი ბლოკური ტიპის ელემენტები გააჩნია HTML--ის ენას?
- ჩამოთვალეთ მათი სპეციფიკაცია და გამოყენების წესები
- შეიძლება თუ არა ყველა ახალი ბლოკური ტიპის ელემენტის მრავალჯერადად დოკუმენტში გამოყენება?
- რომელი ტაგებით ვაფორმირებთ დოკუმენტში დინამიურ გამოსახულებას?
- დაასახელეთ მათ შორის განსხვავება
- დაასახელეთ რომელი უფრო პრიორიტეტულია მათგან და რით განსხვავდებიან ერთმანეთისგან

## 7.2. ვებსაიტზე მულტიმედია ელემენტების გამოყენება

**მიმდინარე პარაგრაფის თემატიკა**

- რას წარმოადგენ მულტიმედიური  ელემენტები, მოკლე მიმოხილვა
- მულტიმედიური ელემენტების ფორმატები და სპეციფიკა
- WEB-ზე გამოსაყენებლად რეკომენდებული ვიდეო ფაილების ფორმატები
- WEB-ზე გამოსაყენებლად რეკომენდებული აუდიო ფაილების ფორმატები
- ვებსაიტზე მულტიმედიური ელემენტის შემოტანა
- ვებსაიტზე მულტიმედიური ელემენტის თვისებები

### რას წარმოადგეს მულტიმედიური ელემენტები? მოკლე მიმოხილვა

საიტზე მულტიმედიურ ელემენტებს ვუწოდებთ ყველა იმ ელემენტს, რომელიც არ გამოისახება ტექსტის სახით. ანუ მულტიმედიური ელემენტები არსებობს როგორც

- ვიდეო
- აუდიო
- გრაფიკული
- ანიმაცია

თითოეულ მულტიმედიურ ელემენტს გააჩნია თავისი სპეციფიური ფორმატი. HTML 5 ვერსიამდე მულტიმედიური და განსაკუთრებულად ვიდეო-აუდიო და ანიმაციური ფაილების ჩაშენებას ბრაუზერში და შემდგომ მის ასახვას უზრუნველყოფდა FLASH ფლეიერი. დღესდღეობით კი HTML 5 ის მეშვეობით გაცილებით მარტივად შეგვიძლია მივაწოდოთ მომხმარებელს ჩვენს მიერ ჩაშენებული მულტიმედიური ფაილი იქნება ეს ვიდეო, ანიმაცია თუ აუდიო ფაილი.

### მულტიმედიური ელემენტების ფორმატები და სპეციფიკა

**ვიდეო ფაილების ფორმატები და მათი აღწერა**

| ფორმატი | გაფართოება | აღწერა |
|---|---|---|
| MPEG | .mpg<br>.mpeg | MPEG. შემუშავებულია „ Moving Pictures Expert Group“ -ის მიერ პირველი ყველაზე პოპულარული ვიდეო ფორმატის მქონე ფაილი WEB-ზე გამოსაყენებლად. მისი მხარდაჭერა გააჩნია ყველა ბრაუზერს მაგრამ არ გამოიყენება HTML 5 -ში. მის ნაცვლად ვიყენებთ MP4-ის ფორმატის ფაილებს. |
| AVI | .avi | AVI (Audio Video Interleave).  შემუშავებულია კომპანია Microsoft-ის მიერ. კარგი ხარისხით აისახება WINDOWS-ის გარემოში მაგრამ არა ბრაუზერებში |
| WMV | .wmv | WMV (Windows Media Video). შემუშავებულია კომპანია Microsoft-ის მიერ. კარგი ხარისხით აისახება WINDOWS-ის გარემოში მაგრამ არა ბრაუზერებში. |

| | | |
|---|---|---|
| QuickTime | .mov | QuickTime. შემუშავებულია კომპანია Apple-ის მიერ. კარგი ხარისხით აისახება Apple-ის გარემოში მაგრამ არა ბრაუზერებში. |
| RealVideo | .rm<br>.ram | RealVideo შემუშავებულია Real Media-ს მიერ რათა უზრუნველყო ვიდეოს გადმოცემა ნელი ინტერნეტ კავშირის შემთხვევაშიც. დღემდე გამოიყენება ონლაინ ვიდეოს ფაილების გადმოსაცემად და ინტერნეტ ტელევიზიებისთვის. არ გამოიყენება ბრაუზერებისთვის. |
| Flash | .swf<br>.flv | Flash- შემუშავებულია კომპანია Macromedia-ს მიერ და როგორც წესი მოითხოვს დამატებით კომპონენტებს, ეგრეთწოდებულ plug-in-ის რათა ბრაუზერს ჰქონდეს მისი ასახვის საშუალება. |
| Ogg | .ogg | Theora Ogg. შემუშავებულია „Xiph.Org Foundation“-ის მიერ. ამ ფორმატს გააჩნია სრული HTML5-ის მხარდაჭერა. |
| WebM | .webm | WebM. შემუშავებულია შემდეგი კომპანიების მიერ: web giants, Mozilla, Opera, Adobe და Google. ამ ფორმატს გააჩნია სრული HTML5-ის მხარდაჭერა. |
| MPEG-4 ან MP4 | .mp4 | MP4. შემუშავებულია „Moving Pictures Expert Group“ -ის მიერ. ამ ფორმატს გააჩნია სრული HTML5-ის მხარდაჭერა და რეკომენდებულია YouTube-ის მიერ. |

აქვე გვინდა მივანიშნოთ, რომ ზემოთ ჩამოთვლილი ვიდეო ფორმატებიდან მხოლოდ MP4, WebM და Ogg ფორმატის ვიდეო ფაილების მხარდაჭერა გააჩნია HTML5-ის მარკირების ენის სტანდარტს.

### აუდიო ფაილების ფორმატები და მათი აღწერა

| ფორმატი | გაფართოება | აღწერა |
|---|---|---|
| MIDI | .mid<br>.midi | MIDI (Musical Instrument Digital Interface). წარმოადგენს ელექტრონული მუსიკის მთავარ ფორმატს. მას კარგად აღიქვამს ყველა ციფრული მოწყობილობა. მათ შორის კომპიუტერული და მუსიკალური მოწყობილობები. არ გამოიყენება ბრაუზერებში. |
| RealAudio | .rm<br>.ram | RealAudio. შემუშავებულია „ Real Media“ -ის მიერ რათა უზრუნველყო დაბალი ინტერნეტ კავშირის შემთხვევაში მისი გადმოცემა. არ გამოიყენება ბრაუზერებში. |
| WMA | .wma | შემუშავებულია კომპანია Microsoft-ის მიერ. კარგი ხარისხით ჟღერს WINDOWS-ის გარემოში მაგრამ არა ბრაუზერებში. |
| AAC | .aac | AAC (Advanced Audio Coding). შემუშავებულია კომპანია Apple-ის მიერ როგორც „iTunes“ ძირითადი ფორმატი. საუკეთესოდ ჟღერს Apple-ის ფირმის კომპიუტერებზე მაგრამ არა ბრაუზერებში. |
| WAV | .wav | WAV. შემუშავებულია IBM და Microsoftის მიერ. საუკეთესოდ ჟღერს Windows, Macintosh, და Linux-ის ოპერაციულ სისტემებზე. გააჩნია HTML5-ის სრული მხარდაჭერა. |
| Ogg | .ogg | Theora Ogg. შემუშავებულია „Xiph.Org Foundation“-ის მიერ. ამ ფორმატს გააჩნია სრული HTML5-ის მხარდაჭერა. |

| | | |
|---|---|---|
| MP3 | .mp3 | MP3 ფაილები როგორც წესი წარმოადგენენ MPEG ფაილების ფორმატის აუდიო ნაწილს. MP3 ყველაზე პოპულარული და გავრცელებული ფორმატია აუდიო ფაილების გადმოსაცემად. ის გახლავთ კარგად კომპრესირებული ტიპის ფაილები საუკეთესო ხარისხით. ამის გამო შედეგად MP3 ფორმატის ფაილები ზომაში საგრძნობლად პატარა გახლავთ. მისი აღქმის მხარდაჭერა გააჩნია ყველა ბრაუზერს. |
| MP4 | .mp4 | MP4. შემუშავებულია „Moving Pictures Expert Group“ -ის მიერ. ეს ფორმატი რეალურად შემუშავებულია ვიდეო ფაილებისთვის მაგრამ შეიძლება გამოყენებული იქნეს აგრეთვე აუდიო ფაილებისთვისაც. მას გააჩნია სრული HTML5-ის მხარდაჭერა. |

აქვე გვინდა მივანიშნოთ, რომ ზემოთ ჩამოთვლილი ვიდეო ფორმატებიდან მხოლოდ MP3, WAV და Ogg ფორმატის აუდიო ფაილების მხარდაჭერა გააჩნია HTML5-ის მარკირების ენია სტანდარტს.

ვინაიდან ზემოთ აღწერილი ცხრილებიდან შეგვიძლია გავაანალიზოთ თუ რომელი ფორმატის ვიდეო ან აუდიო ფაილი სჯობს რომ გამოვიყენოთ ბრაუზერებში მათი სტანდარტის მიხედვით გამოყენების უზრუნველყოფის მიზნით მივდივართ დასკვნამდე , რომ

სჯობს ვიდეო ფაილების ჩვენს პეოექტში ჩაშენებისთვის შევარჩიოთ შემდეგი ფორმატის ფაილები:

MP4, WebM და Ogg

ხოლო რაც შეეხება აუდიო ფაილებს გამოვიყენოთ შემდეგი ფორმატის ფაილები:

MP3, WAV და Ogg.

## ვებსაიტზე მულტიმედიური ელემენტის შემოტანა

მოგეხსენებათ HTML 5 ის ვებ მარკირების ენის შემოღებამდე ვებსაიტზე ვიდეოს შემოტანის სტანდარტიზებული მეთოდი არ არსებობდა. როგორც ზემოთ უკვე აღვნიშნეთ ვიდეო ფაილის აღსაქმელად ბრაუზერს ესაჭიროებოდა ეგრეთწოდებული Flash Player- ი, ანუ დამატებითი კომპონენტი (Plug-in)-ი. HTML 5 ის სტანდარტის მიხედვით ეს პრობლემა უკვე აღმოფხვრილია და ვიდეო ფაილის ბრაუზერში ინტეგრაცია გაცილებით მარტივია ტაგი <video> -ს მეშვეობით, რომელსაც რა თქმა უნდა გააჩნია საკუთარი თვისებები როგორიც არის ვიდეოს ზომები, ანუ სიმაღლე და სიგანე და ასე შემდეგ.

ქვემოთ წარმოდგენილია პრაქტიკული მაგალითი თუ როგორ უნდა ჩავაშენოთ ჩვენთვის სასურველი ვიდეო ფაილი ბრაუზერში.

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  თქვენს ბრაუზერს არ გააჩნია მულტიმედიური ფორმატის მხარდაჭერა
  </video>
</body>
</html>
```


სურ. 3.35.ტაგი <video>-ს მეშვეობით ვოდეო ფაილის დოკუმენტში შემოტანის ნიმუში

ახლა განვიხილოთ ტაგი <video>- ს გამოყენების სპეციფიკა. რაც შეეხება თვისებებს HEIGHT-ს და WIDTH -ს ვფიქრობთ ეს თვისებები ყველასათვის უკვე ნათელია და ყველამ ვიცით მათი დანიშნულება. გადავიდეთ თვისებაზე CONTROLS ეს თვისება ვიდეო კონსოლს (ფლეიერს) განუსაზღვრავს კონტროლის ღილაკების თვისებებს, როგორიცაა PLAY-ს ღილაკი, ღილაკი PAUSE და ხმის მაკონტროლებელი ღილაკი VOLUME.

აგრეთვე დააკვირდებოდით ტაგში <video> განთავსებულია ტაგი <source>. სწორედ ამ ტაგის და მისი თვისებების მეშვეობით უკავშირდება დოკუმენტი კონკრეტულ ვიდეო ფაილს. ჩვენს შემთხვევაში ფაილის URL-ს მისამართს და მის სახელს წარმოადგენს „movie.mp4“. გარდა ამისა მოცემულ ტაგს გააჩნია თვისება TYPE , რომელიც განსაზღვრავს მედია ფაილის ტიპს.

ქვემოთ მოყვანილ ცხრილში ჩამოთვლილია დღევანდელი სტანდარტის მიხედვით გამოყენებადი ფაილის ფორმატის მედია ტიპები:

| ვიდეო ფაილის ფორმატი | მედია ტიპი |
|---|---|
| MP4 | video/mp4 |
| WebM | video/webm |

| | |
|---|---|
| Ogg | video/ogg |

ზემოთ მოყვანილ მაგალითში შენიშნავდით ტაგები <video></video>- ს შორის განთავსებულ ტექსტს:

**„თქვენს ბრაუზერს არ გააჩნია მულტიმედიური ფორმატის მხარდაჭერა “**

ეს ტექსტი აისახება ბრაუზერებში მხოლოდ იმ შემთხვევაში თუ ბრაუზერს არ გააჩნია ვიდეო ფორმატის მქონე ფაილების მხარდაჭერა. ვინაიდან მომხმარებელი არ შევიყვანოთ შეცდომაში და ინფორმირებული იყოს იმის თაობაზე, რომ თუ ბრაუზერი არ ასახავს ვიდეო ფაილს ეს არ არის საიტის ხარვეზი და იზრუნოს იმაზე, რომ იხილოს ჩვენს მიერ მიწოდებული საიტი შესაბამისი ბრაუზერის მეშვეობით.

ახლა წარმოგიდგენთ შემდეგ მაგალითს

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  თქვენს ბრაუზერს არ გააჩნია მულტიმედიური ფორმატის მხარდაჭერა
  </video>
</body>
</html>
```

სურ. 3.36.ტაგი <video>-ს მეშვეობით ვიდეო ფაილის დოკუმენტში შემოტანის ნიმუში დამატებითი თვისებებით

ზემოთ მოყვანილ მაგალითში გვინდა ხაზი გავუსვათ თვისებას AUTOPLAY-ს, აღნიშნული თვისება უზრუნველყოფს ვიდეო ფორმატის ფაილის ავტომატურად ჩართვას. ასეთ შემთხვევაში ვიდეო კონსოლის ღილაკები რომლებზეც ზემოთ გვქონდა საუბარი PLAY, PAUSE და VOLUME უკვე აღარ არის მომხმარებლისთვის ხელმისაწვდომი. ანუ ვიდეო ჩატვირთვისთანავე გაეშვება ავტომატურ რეჟიმში.

თუ გავაკეთებთ მოკლე რეზიუმეს, აღმოჩნდება, რომ ვებსაიტზე HTML 5 ის სტანდარტის მიხედვით უნდა გამოვიყენოთ სამი ფორმატის ვიდეო ფაილები, ვინაიდან უზრუნველეყოთ მომხმარებლისთვის ვიდეო მასალის მიწოდება ბრაუზერებისთვის აღსაქმელ ფორმატში. ეს ფორმატები გახლავთ:

- MP4
- WebM

- Ogg

ქვემოთ მოყვანილ ცხრილში ასახულია რომელ ბრაუზერს გააჩნია კონკრეტული ვიდეო ფაილების ფორმატის მხარდაჭერა.

| ბრაუზერები | MP4 | WebM | Ogg |
|---|---|---|---|
| Internet Explorer | დიახ | არა | არა |
| Chrome | დიახ | დიახ | დიახ |
| Firefox | დიახ | დიახ | დიახ |
| Safari | დიახ | არა | არა |
| Opera | დიახ | დიახ | დიახ |

HTML დოკუმენტში აუდიო ფაილის შემოსატანად გამოიყენება ტაგი <audio>, რომელსაც ტაგი <video>-მსგავსად გააჩნია თვისება controls . აღნიშნული თვისება უზრუნველყოფს აუდიო კონსოლის კონტროლის ღილაკების არსებობას, როგორებიცაა PLAY, PAUSE და VOLUME.

```
<!DOCTYPE html>
<html>
<body>

<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  თქვენს ბარუზერს არ გააჩნია აუდიო ფაილების აღქმის მხარდაჭერა
</audio>

</body>
</html>
```


სურ. 3.37.ტაგი <audio>-ს მეშვეობით აუდიო ფაილის დოკუმენტში შემოტანის ნიმუში

აღნიშნულ ტაგს ასევე ვიდეოს მსგავსად გააჩნია თვისება AUTOPLAY. ამ თვისების გამოყენების შემთხვევაში კონსოლი საერთოდ არ აისახება ბრაუზერში და აუდიო ფაილი დოკუმენტის ჩატვირთვისთანავე იწყებს აუდიო ფაილის ფონურ რეჟიმში გაშვებას.

აგრეთვე აუდიო ფაილის ტაგსაც, ვიდეოს მსგავსად გააჩნია ტაგი <source>, რომელიც მისი თვისებების მეშვეობით უკავშირდება კონკრეტულ აუდიო ფაილს. ჩვენს შემთხვევაში ფაილის URL-ს მისამართს და მის სახელს წარმოადგენს „horse.ogg“. გარდა ამისა მოცემულ ტაგს გააჩნია თვისება TYPE , რომელიც განსაზღვრავს მედია ფაილის ტიპს.

ქვემოთ მოყვანილ ცხრილში წარმოდგენილია იმ აუდიო ფაილების მედია ტიპები რომლებიც გამოიყენება HTML5-ის სტანდარტის მიხედვით.

| აუდიო ფაილის ფორმატი | მედია ტიპი |
|---|---|
| MP3 | audio/mpeg |
| Ogg | audio/ogg |

| | |
|---|---|
| Wav | audio/wav |

ტაგებს შორის მოთავსებული ტექსტი:

**„თქვენ ბრაუზერს არ გააჩნია აუდიო ფაილების აღქმის მხარდაჭერა“**

აისახება მხოლოდ მაშინ როდესაც ბრაუზერი ვერ აღიქვამს აუდიო ფალის ფორმატს და ვერ გადმოსცემს მას მომხმარებლისთვის.

ქვემოთ მოყვანილია მაგალით სადაც აუდიო კონსოლს მითითებული აქვს თვისება AUTOPLAY

```
<!DOCTYPE html>
<html>
<body>

<audio autoplay>
  <source src="horse.ogg" type="audio/ogg">
  თქვენს ბარუზერს არ გააჩნია აუდიო ფაილების აღქმის მხარდაჭერა
</audio>

</body>
</html>
```

HTML 5 ის სტანდარტის მიხედვით უნდა გამოვიყენოთ სამი ფორმატის აუდიო ფაილები, ვინაიდან უზრუნველეყოთ მომხმარებლისთვის აუდიო მასალის მიწოდება ბრაუზერებისთვის აღსაქმელ ფორმატში. ეს ფორმატები გახლავთ:

- MP3
- Wav
- Ogg

ქვემოთ მოყვანილ ცხრილში ასახულია რომელ ბრაუზერს გააჩნია კონკრეტული ვიდეო ფაილების ფორმატის მხარდაჭერა.

| ბრაუზერები | MP3 | Wav | Ogg |
|---|---|---|---|
| Internet Explorer | დიახ | არა | არა |
| Chrome | დიახ | დიახ | დიახ |
| Firefox | დიახ | დიახ | დიახ |
| Safari | დიახ | დიახ | არა |
| Opera | დიახ | დიახ | დიახ |

მულტიმედიური ელემენტების დოკუმენტში ჩასაშენებლად ასევე აქტუალურად გამოიყენება ტაგი <object>, ამ ტაგის მეშვეობით შეგვიძლია დოკუმენტში ჩავაშენოთ სხვა მულტიმედიურ ელემენტებთან ერთად როგორც FLASH, JAVA APPLET და PDF ფორმატის ფაილები ასევე HTML ფორმატის ფაილები. ქვემოთ მოყვანილია მაგალითები რომლებიც ასახავენ თუ როგორ გამოიყენება აღნიშნული ტაგი პრაქტიკაში.

**FLASH ფორმატის ფაილის შემოტანა დოკუმენტში**

```
<!DOCTYPE html>
<html>
<body>

<object width="400" height="50" data="flashfile.swf"></object>

</body>
</html>
```

**HTML ფორმატის ფაილის შემოტანა დოკუმენტში**

```
<!DOCTYPE html>
<html>
<body>

<object width="100%" height="500px" data="snippet.html"></object>

</body>
</html>
```

სურ. 3.38.ტაგი <object>-ის მეშვეობით HTML ფაილის დოკუმენტში შემოტანის ნიმუში

ასევე ანალოგიური გახლავთ ტაგი <EMBED> რომელსაც უკვე დიდი ხანია ვიყენებთ HTML-ში და მისი მხარდაჭერა HTML-ის მარკირების ენის ძველ ვერსიებსაც კი ჰქონდათ.

**FLASH ფორმატის ფაილის შემოტანა დოკუმენტში ტაგი <embed> ის მეშვეობით**

```
<!DOCTYPE html>
<html>
<body>

<embed width="400" height="50" src="flashfile.swf">

</body>
</html>
```

სურ. 3.39.ტაგი <embed>-ს მეშვეობით ფლეშ ფორმატის ფაილის დოკუმენტში შემოტანის ნიმუში

**HTML ფორმატის ფაილის შემოტანა დოკუმენტში ტაგი <embed> ის მეშვეობით**

```
<!DOCTYPE html>
<html>
<body>

<embed width="100%" height="500px" src="snippet.html">

</body>
</html>
```

სურ. 3.40.ტაგი <embed>-ს მეშვეობით HTML ფაილის დოკუმენტში შემოტანის ნიმუში

**გრაფიკული ფორმატის ფაილის შემოტანა დოკუმენტში ტაგი <embed> ის მეშვეობით**

```
<!DOCTYPE html>
<html>
<body>

<embed src="audi.jpeg">

</body>
</html>
```

სურ. 3.41.ტაგი <embed>-ის მეშვეობით გრაფიკული ფაილის დოკუმენტში შემოტანის ნიმუში

*კითხვები თვით შეფასებისთვის:*

- რამდენი ტიპის ვიდეო ფორმატის ფაილი არსებობს, რომლებიც შეგვიძლია, რომ ჩავაშენოთ ვებ გვერდზე?
- ჩამოთვალეთ თითოეულ მათგანს შორის განსხვავება
- შეიძლება თუ არა, რომ ვებ გვერდზე ჩავაშენოთ HTML ფორმატის ფაილი?
- ჩამოთვალეთ სხვა მულტიმედიური ობიექტები რომლებიც შეიძლება, რომ გამოვიყენოთ ვებ გვერდის გასაფორმებლად
- რა განსხვავებაა ამ მულტიმედიურ ობიექტებს შორის?
- რომელი ფორმატის აუდიო ფაილი უფრო ოპტიმალურია ვებ სივრცეში გამოსაყენებლად?

## 7.3. ვებსაიტზე ფორმის ელემენტების გაფართოებული ტიპების გამოყენება

### HTML 5 ის მარკირების ენაში დამატებული ფორმის ელემენტები და მათი ტიპები

**მიმდინარე პარაგრაფის თემატიკა**

- <input> ელემენტის ტიპები NUMBER, DATE, RANGE, COLOR, EMAIL, URL
- ელემენტი list-ის მიმოხილვა

- ტაგი <datalist> -ის მიმოხილვა

HTML 5 ის ვებ მარკირების ენაში გვეძლევა საშუალება გამოვიყენოთ ფორმის ელემენტების გაფართოებული საშუალებები. ჩვენ შევეცდებით მოცემულ პარაგრაფში გადმოგცეთ ყველა მათგანის გამოყენების წესი და სპეციფიკა. მართალია ქვემოთ ჩამოთვლილი გაფართოებული შესაძლებლობების ფორმის ელემენტები არ სარგებლობენ ყველა წამყვანი ბრაუზერების მიერ მხარდაჭერით მაგრამ გარწმუნებთ ამა თუ იმ შემთხვევაში ამ ელემენტების გამოყენება ძალიან უმარტივებს საქმეს ვებ გვერდის შემქმნელებს.

დავიწყოთ ყველასათვის კარგად ნაცნობი და ყველაზე ხშირად გამოყენებადი ფორმის ელემენტით <input>. განვიხილოთ თუ რა გაფართოებული საშუალებები დაემატა ამ ელემენტს.

<input type="number">

აღნიშნული ელემენტის მეშვეობით შეგვიძლია შევქმნათ რიცხვითი ტიპის ველი და დავუწესოთ მას მინიმალური და მაქსიმალური ზღვარი.

```
<!DOCTYPE html>
<html>
<body>

  <input type="number" name="quantity" min="1" max="5">

</body>
</html>
```


სურ. 3.42.ტაგი number-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

შედეგად მივიღებთ ასეთი ტიპის input ელემენტს, რომელშიც შესაძლებელი იქნება რიცხვითი ტიპის მნიშვნელობის მითითება 1- დან 5 ის ჩათვლით. როგორც დააკვირდით მნიშვნელობის ქვედა და ზედა ზღვარს განსაზღვრავენ თვისებები: MIN და MAX.

ქვემოთ მოყვანილია ცხრილი სადაც ჩამოთვლილია ელემენტი number-ის თვისებები:

| თვისება | თვისების აღწერა |
|---|---|
| Disabled | უთითებს, რომ ელემენტი იყოს გაუქმებული (არა რედაქტირებადი) |
| Max | ელემენტის მაქსიმალური დასაშვები მნიშვნელობა |
| Maxlength | განსაზღვრავს ველის შიგთავსის მაქსიმალურ სიგრძეს |
| Min | ელემენტის მინიმალური დასაშვები მნიშვნელობა |
| Readonly | უთითებს, რომ ველი იყოს მხოლოდ წაკითხვადი, ანუ მისი რედაქტირება შეუძლებელია |
| Required | მინიშნება, რომ ველი აუცილებლად შესავსებია |
| Size | განსაზღვრავს ველის ზომას სიგანეში სიმბოლოების რაოდენობის მიხედვით |
| Step | განსაზღვრავს თითოეული ბიჯის ინტერვალს მზარდობის ან კლებადობის მიხედვით |

| | |
|---|---|
| Value | უთითებს პირველად მნიშვნელობას მოცემული ტიპის ელემენტს |

ქვემოთ მოყვანილია მაგალითი, რომლის მინიმალური მნიშვნელობაც გახლავთ 0, მაქსიმალური დასაშვები მნიშვნელობა 100, პირველადი მნიშვნელობა 30 ხოლო მნიშვნელობის ცვალებადობის ბიჯი 10 ერთეული:

```
<!DOCTYPE html>
<html>
<body>

<input type="number" name="points" min="0" max="100" step="10" value="30">

</body>
</html>
```

სურ. 3.43.ტაგი number-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში დამატებითი თვისებებით

განვიხილოთ შემდეგი ელემენტი:

<input type=”date”>

მოცემული ელემენტი გამოიყენება ველში სადაც მონაცემი უნდა შევიტანოთ თარიღის ფორმატში. აღნიშნული ელემენტის მხარდაჭერა არ გააჩნიათ ბრაუზერებს: Internet Explorer-ს და Firefox-ს. მისი გამოყენების სინტაქსი შემდეგნაირია:

```
<!DOCTYPE html>
<html>
<body>

  დაბადების თარიღი:
  <input type="date" name="bday">

</body>
</html>
```

ხოლო ბრაუზერებში რომლებიც აღიქვამენ ამ ელემენტს აღნიშნული ელემენტი გამოიყურება შემდეგნაირად:


სურ. 3.44.ტაგი date-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

მოცემულ ელემენტს გააჩნია ორი დამახასიათებელი თვისება, ესენი გახლავთ: min და max, აღნიშნული თვისებებით შეგვიძლია ელემენტს დავუწესოთ ზღვარი მინიმალურსა და მაქსიმალურ მნიშვნელობებს შორის.

```
<!DOCTYPE html>
<html>
<body>

მიუთითეთ თარიღი 1980-01-01- მდე:
<input type="date" name="bday" max="1979-12-31">
მიუთითეთ თარიღი 2000-01-01-ს შემდეგ:
<input type="date" name="bday" min="2000-01-02">

</body>
</html>
```

სურ. 3.45.ტაგი number-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში დამატებითი თვისებებით

ზემოთ მოყვანილი კოდის მაგალითის მიხედვით, პირველ ელემენტში თვისება max-ის მეშვეობით დადებული გვაქვს შეზღუდვა, ანუ მომხმარებელი მითითებული თარიღის შემდგომ კალენდარულ რიცხვს ვერ მიუთითებს. ანალოგიური გვაქვს გაკეთებული მეორე ელემენტს მხოლოდ საპირისპირო თვისებით min, სადაც ვზღუდავთ მომხმარებელს მიუთითოს თარიღი 2000 წლის 2 იანვრამდე.

შემდეგი ელემენტი გახლავთ ელემენტი RANGE, რომელსაც ვიყენებთ სლაიდერის ფორმით გარკვეული მნიშვნელობების ასარჩევად კონკრეტულ რიცხვითი მნიშვნელობების დიაპაზონს შორის. მისი გამოყენების სინტაქსი შემდეგნაირია:

```
<!DOCTYPE html>
<html>
<body>

  რიცხვითი მნიშვნელობები:
  <input type="range" name="points" >

</body>
</html>
```

მოცემული კოდის შედეგი კი ქვემოთ არის მოყვანილი:

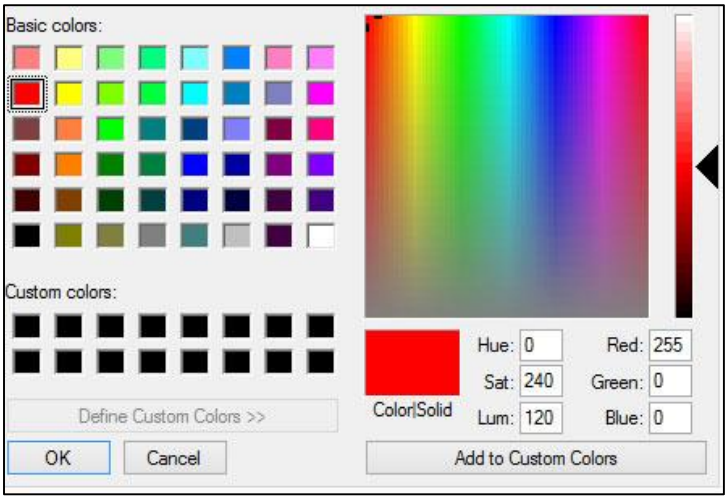

სურ. 3.46.ტაგი range-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

აღნიშნულ ელემენტს შეგვიძლია დავუწესოთ რიცხვითი მნიშვნელობების მინიმალური და მაქსიმალური დასაშვები ზღვარი თვისებებით min და max

```
<!DOCTYPE html>
<html>
<body>

  რიცხვითი მნიშვნელობები:
  <input type="range" name="points" min="0" max="10">

</body>
</html>
```

სურ. 3.47.ტაგი range-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში დამატებითი თვისებებით

შემდეგი ტიპის ელემენტი გახლავთ ელემენტი COLOR, რომელიც საშუალებას გვაძლევს ჩვენთვის კარგად ნაცნობი ფერების პალიტრის დიალოგური ფანჯრიდან ავირჩიოთ გარკვეული ფერი. აღნიშნული ელემენტი მიიღებს ჩვენს მიერ არჩეული ფერის მნიშვნელობას რიცხვითი სახით. კონკრეტულად კი თექვსმეტობითი ათვლის სისტემით.




```
<!DOCTYPE html>
<html>
<body>

  აირჩიეთ სასურველი ფერი:
  <input type="color" name="favcolor">

</body>
</html>
```

აღნიშნულ ელემენტს გააჩნია თვისება value, როდესაც ამ თვისებას არ ვიყენებთ მოცემულ ელემენტთან მაშინ მისი მნიშვნელობა გაჩუმების პრინციპით არის #000000, ანუ შავი ფერის თექვსმეტობითი ათვლის სისტემით წარმოდგენა.

აირჩიეთ სასურველი ფერი:

სურ. 3.48.ტაგი color-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

ხოლო, თუ მივმართავთ მის თვისებას value-ს მაში საშუალება გვეძლევა თავიდანვე მივუთითოთ მას საწყისი ფერი. მაგალითად:

```
<!DOCTYPE html>
<html>
<body>

  აირჩიეთ სასურველი ფერი:
  <input type="color" name="favcolor" value="#ff0000">

</body>
</html>
```

ზემოთ მოყვანილი კოდის შედეგი ბრაუზერში:

აირჩიეთ სასურველი ფერი:

სურ. 3.49.ტაგი color-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

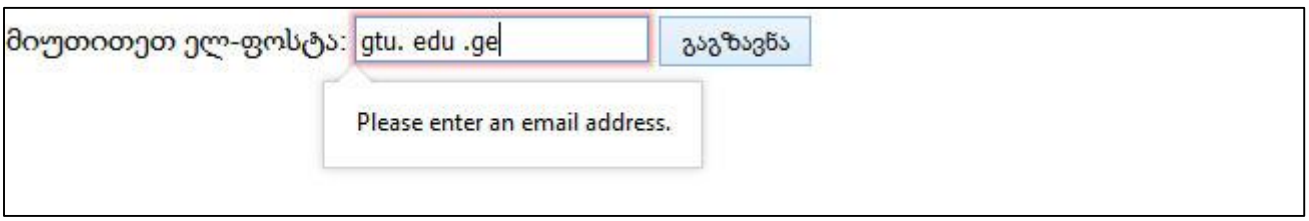
მოყვანილ მაგალითში თვალსაჩინოდ ჩანს, რომ აღნიშნული ელემენტის საწყისი მნიშვნელობა ანუ თვისება value განსაზღვრულია წითელი ფერით #ff0000.

შემდეგი ელემენტი რომელიც უნდა განვიხილოთ გახლავთ პრაქტიკაში გამოსაყენებლად ძალიან სასარგებლო ელემენტი ეს არის EMAIL. მოგეხსენებათ ვების სტანდარტით ელფოსტის მისამართში აუცილებლად უნდა იყოს სიმბოლოები @ (ძაღლუკა), აუცილებლად სიმბოლოების ნაკრებში უნდა ერიოს წერტილის სიმბოლო და ამ წერტილის შემდეგ გარკვეული სიმბოლოების ნაკრები. სწორედ ამ ფორმატის დაცვას ემსახურება ხსენებული ელემენტი. საკმარისია დაირღვეს ელ-ფოსტის სტანდარტიზებული მისამართის ფორმატი, რომ ის უმალ გამოგვიტანს შეტყობინებას ამის შესახებ. ხაზგასმით გვინდა აღვნიშნოთ, რომ ამ ფორმის ელემენტის გადამოწმება სისწორეზე ხდება მხოლოდ მას შემდეგ როცა ფორმისთვის გამოვიყენებთ ჩვენთვის უკვე კარგად ცნობილ ღილაკს SUBMIT-ს.

```
<!DOCTYPE html>
<html>
<body>

<form action="action_page.php">
  მიუთითეთ ელ-ფოსტა:
  <input type="email" name="email">
  <input type="submit" value="გაგზავნა">
</form>

</body>
</html>
```

სურ. 3.49.ტაგი email-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

ზემოთ მოცემულ მაგალითზე კარგად ჩანს, რომ გვაქვს ფორმა, რომელიც უნდა გადაიგზავნოს სერვერზე კონკრეტულად რომელიღაც PHP მოდულთან, გვაქვს ელემენტი EMAIL და ასევე ფორმისთვის აუცილებელი ელემენტი SUBMIT-ი. სწორედ ამ საბმითის რეალიზების შემდგომ ამოწმებს ბრაუზერი სწორად იქნა თუ არა შევსებული ელ-ფოსტის ველი. ქვემოთ მოყვანილ მაგალითში ელემენტი email შეგნებულად შევავსეთ შეცდომით და მივიღეთ შემდეგნაირი შედეგი:

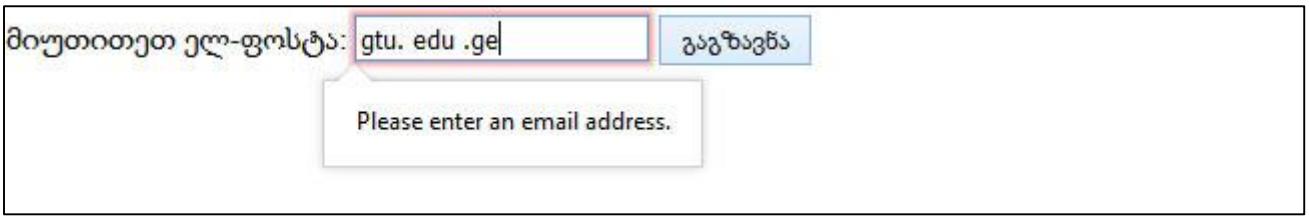



სურ. 3.50. ტაგი email-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

თვალნათლივ ჩანს, რომ ფორის მონაცემები სერვერზე არ გაიგზავნა და ბრაუზერმა გამოგვიტანა შეტყობინება არაკორექტული ელ-ფოსტის ფორმატთან დაკავშირებით.

ელემენტი EMAIL -ის მსგავსად ანალოგიური ფუნქცია აკისრია ჩვენს შემდგომ განსახილველ ელემენტს, მხოლოდ იმ განსხვავებით , რომ აღნიშნული ელემენტი ემსახურება URL მისამართის ფორმატის გადამოწმებას.

```
<!DOCTYPE html>
<html>
<body>

<form action="action_page.php">
  მიუთითეთ ელ-ფოსტა:
  <input type="email" name="email">
  <input type="submit" value="გაგზავნა">
</form>

</body>
</html>
```



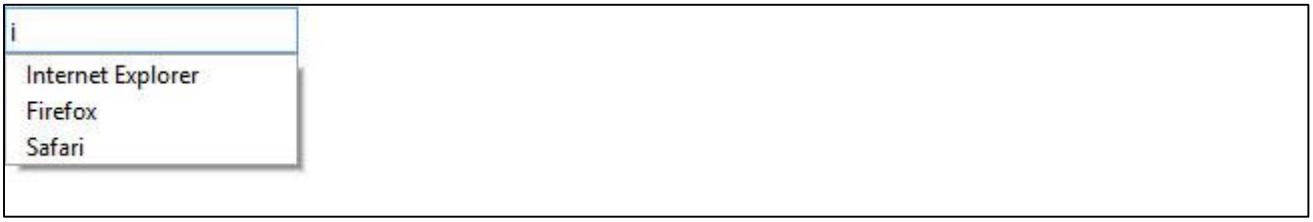

სურ. 3.51. ტაგი email-ის ტიპის <input>-ის პრაქტიკული გამოყენების ნიმუში

აქაც ანალოგიურად წინა მაგალითისა თვალნათლივ ჩანს, რომ ფორმაში არსებული ელემენტი URL მოწმდება ბრაუზერის მიერ ჩანაწერის ფორმატის სიზუსტეზე და უზუსტობის შემთხვევაში გამოაქვს შეტყობინება და ამავდროულად ფორმის ელემენტის მნიშვნელობა არ გადაეცემა სერვერზე არსებულ PHP მოდულს დასამუშავებლად.

HTML 5-ის მარკირების ენაში გვაქვს კიდევ ასეთი საინტერესო ელემენტი LIST, რომელსაც როგორც წესი სტრუქტურულად მოყვება ტაგი **< datalist >.** ამ ელემენტის და აღნიშნული ტაგის საშუალებით შეგვიძლია მივაღწიოთ იმ ეფექტს , როდესაც მომხმარებელი ფორმის ელემენტში ახორციელებს მონაცემის შეტანას და ფორმის ელემენტი მას სთავაზობს ჩამოსაშლელი მენიუს საშუალებით გარკვეულ წინაწარ ჩვენს მიერ განსაზღვრულ მნიშვნელობებს.

```
<!DOCTYPE html>
<html>
<body>

  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>

</body>
</html>
```

შედეგად მივიღებთ <input>-ის ტიპის ელემენტს, რომელშიც გარკვეული მონაცემის შეტანისთანავე ჩამოიშლება წინასწარ ჩვენს მიერ გაწერილი მნიშვნელობები. დააკვირდით იმ გარემოებას, რომ აუცილებელია list ის ტიპის ელემენტს რაც მითითებული აქვს თვისებაში list , იგივე უნდა ჰქონდეს მითითებული ტაგ <datalist>-ს თვისებაში id




სურ. 3.52.ტაგი datalis-ის პრაქტიკული გამოყენების ნიმუში

HTML 5 ის ვებ მარკირების ენას კიდევ აქვს დამატებული მთელი რიგი ელემენტებისა და ტაგებისა რომლებიც ჩვენი განხილვის თემად არ იქცა, იმ მოსაზრების გამო რომ მათ უმრავლესობას ბრაუზერებისგან მოცემულ მომენტში არ გააჩნიათ მხარდაჭერა და მათი გამოყენება არა რეკომენდებულია.

ესენი გახლავთ:

- datetime
- datetime-local
- month
- number
- search
- tel
- time
- week

*კითხვები თვით შეფასებისთვის:*

- რომელი ტიპის ფორმის ელემენტები დაემატა HTML5-ს?
- აქვს თუ არა მხარდაჭერა ყველა თანამედროვე ბრაუზერს ამ ტიპის ელემენტის ასახვისთვის?
- რეკომენდებულია თუ არა ახალი დამატებული ფორმის ელემენტების გამოყენება ?
- რა თვისებები გააჩნია თითოეულ ფორმის ელემენტს?

## 7.4. ბლოკის ვიზუალიზაცია და ანიმაციის გამოყენება

**მიმდინარე პარაგრაფის თემატიკა**

- ოვალურ კუთხეებიანი ბლოკების შექმნა CSS3-ის მეშვეობით
- ბლოკზე ფერების მინიჭება RGBA ფორმატით CSS3-ის მეშვეობით
- ჩრდილის ეფექტის დადება ბლოკზე და ტექსტზე CSS3-ის მეშვეობით
- ანიმაცია CSS3-ის მეშვეობით

მომდევნო პარაგრაფში განვიხილოთ CSS3-ის გაფართოებული შესაძლებლობები. გვინდა ხაზგასმით აღვნიშნოთ, რომ ქვემოთ მოყვანილი მაგალითები არ არის უზრუნველყოფილი ყველა ბრაზაურის მხარდაჭერით. ასე, რომ თითოეული მათგანის გამოყენებისას გულდასმით უნდა გავტესტოთ თითოეული ეფექტი სხვადასხვა ბრაუზერში, რათა უზრუნველყოფილნი ვიყოთ, რომ ჩვენი ვებ გვერდი ყველა ბრაუზერში ერთნაირად აისახებოდეს.

### ოვალურ კუთხეებიანი ბლოკების შექმნა CSS3-ის მეშვეობით

CSS3 -ის მეშვეობით შესაძლებლობა გვეძლევა შევქმნათ ბლოკები რომლებსაც გააჩნიათ ოვალური კუთხეები. ასეთი ბლოკების შექმნის შესაძლებლობა თავიდან გვაცილებს ვებ გვერდზე ზედმეტი გრაფიკული გამოსახულების გამოყენების აუცილებლობას. ასეთი ოვალურ კუთხეებიანი ბლოკის შესაქმნელად CSS3-ში გამოიყენება თვისება **border-radius** , მისი მეშვეობით შეგვიძლია მივუთითოთ ბლოკის დონის ელემენტს თუ რა რადიუსის კუთხე უნდა ჰქონდეს მას.

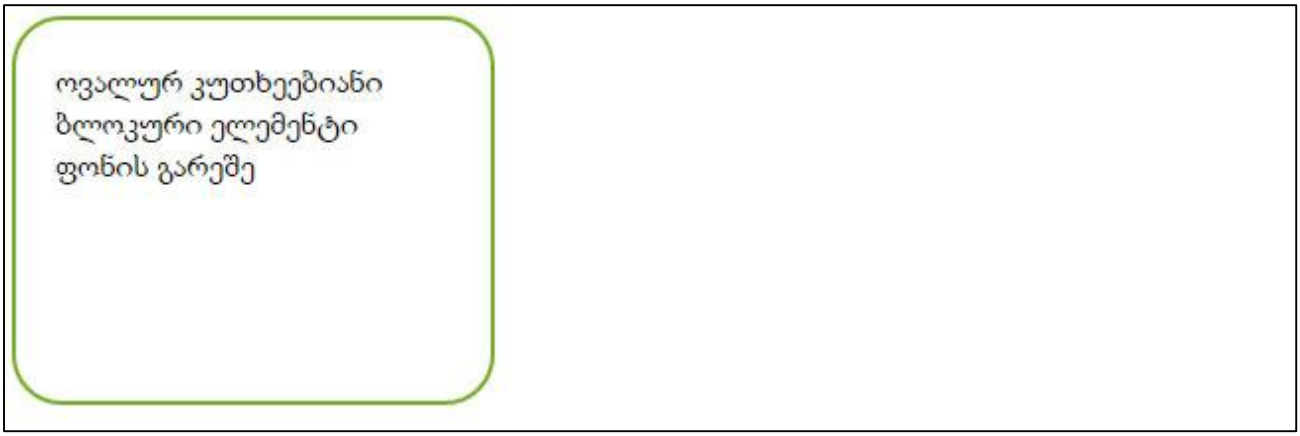
```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners1 {
    border-radius: 25px;
    background: #73AD21;
    padding: 20px;
    width: 200px;
    height: 150px;
}
</style>
</head>
<body>

<div id="rcorners1">ოვალურ კუთხეებიანი ბლოკი ფონის ფერით</div>

</body>
</html>
```


სურ. 3.53. CSS3- ის ოვალურ კუთხიანი ბლოკის გამოყენების ნიმუში

მოყვანილ მაგალითში თვისება border-radius განსაზღვრავს ბლოკის კუთხის მომრგვალების კოეფიციენტს.

შემდეგი მაგალითი თითქმის იდენტურია წინამორბედისა, მხოლოდ ამ შემთხვევაში ბლოკური ტიპის ელემენტს არ გააჩნია ფონის ფერი და წარმოჩენილია უბრალოდ ჩარჩოს სახით.

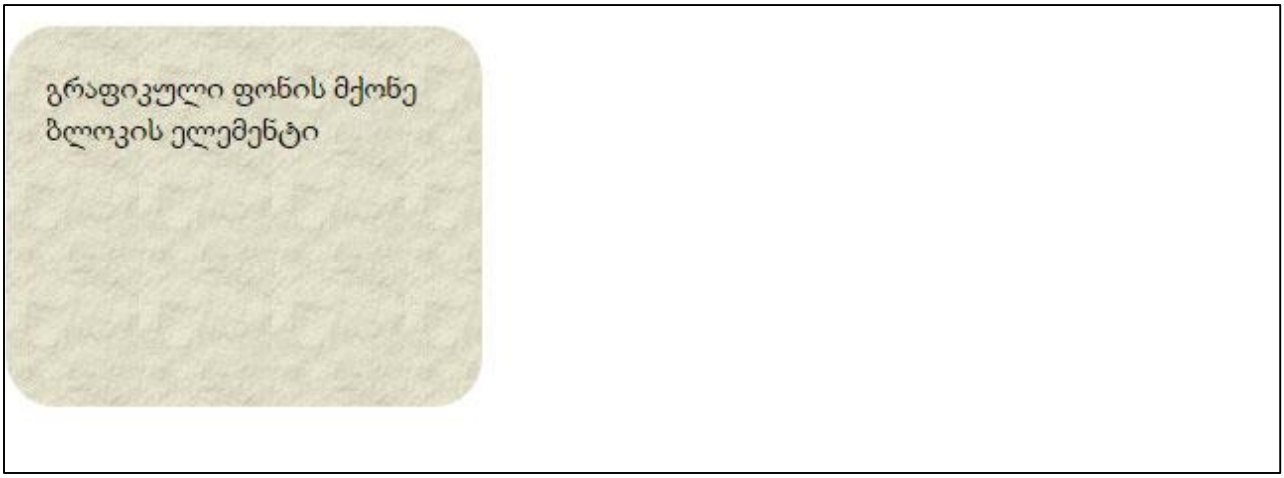
```
<!DOCTYPE html>
<html>
<head>
<style>

#rcorners1 {
    border-radius: 25px;
    border: 2px solid #73AD21;
    padding: 20px;
    width: 200px;
    height: 150px;
}

</style>
</head>
<body>

<div id="rcorners1">ოვალურ კუთხეებიანი ბლოკური ელემენტი ფონის გარეშე</div>

</body>
</html>
```




სურ. 3.54. CSS3- ის ოვალურ კუთხიანი და კონტურიანი ბლოკის გამოყენების ნიმუში

და ბოლოს ვნახოთ თუ როგორ აისახება გრაფიკული ფონის მქონე ბლოკური ტიპის ელემენტი ოვალური კუთხეებით.

```
<!DOCTYPE html>
<html>
<head>
<style>

#rcorners1 {
    border-radius: 25px;
    background: url(paper.gif);
    background-position: left top;
    background-repeat: repeat;
    padding: 20px;
    width: 200px;
    height: 150px;
}
</style>
</head>
<body>

<p id="rcorners1">გრაფიკული ფონის მქონე ბლოკის ელემენტი</p>

</body>
</html>
```




სურ. 3.55. CSS3- ის ოვალურ კუთხიანი ბლოკის გრაფიკული ფონით გამოყენების ნიმუში

ზემოთ განხილულ მაგალითებში ჩვენ ვიხილეთ თუ როგორ შეიძლება მოვამრგვალოთ ბლოკის ტიპის ელემენტის კუთხეები ოთხივე მხრიდან. თუმცა CSS3-ის მეშვეობით შეგვიძლია ვმართოთ ბლოკის თითოეული კუთხის მომრგვალების რადიუსი ერთმანეთისაგან დამოუკიდებლად. ასეთ შემთხვევაში საჭიროა დავიცვათ გარკვეული დადგენილი წესები რომელთა მიხედვითაც გვეძლევა საშუალება განვსაზღვროთ თითოეული კუთხის რადიუსი.

თუ ჩვენ თვისება border-radius-ს მივანიჭებთ:

- 4 მნიშვნელობას, მაშინ პირველი მნიშვნელობა იქნება მარცხენა ზედა კუთხის რადიუსი, მეორე მარჯვენა ზედა კუთხის , მესამე მარჯვენა ქვედა კუთხის და მეოთხე მარცხენა ქვედა კუთხის.
- 3 მნიშვნელობის შემთხვევაში, პირველი იქნება მარცხენა ზედა კუთხის რადიუსი, მეორე მარჯვენა ზედა და მარჯვენა ქვედა კუთხის რადიუსი, ხოლო მესამე მარცხენა ქვედა კუთხის რადიუსი.

- 2 მნიშვნელობის შემთხვევაში, პირველი მნიშვნელობა იქნება მარცხენა ზედა და მარჯვენა ქვედა კუთხეების რადიუსი(ანუ დიაგონალზე) და მეორე მნიშვნელობა იქნება - მარჯვენა ზედა კუთხის და მარცხენა ქვედა კუთხის რადიუსი(დიაგონალზე).
- ერთი მნიშვნელობის მინიჭების შემთხვევაში მითითებული რადიუსი გავრცელდება ოთხივე კუთხეზე.

```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners4 {
    border-radius: 15px 50px 30px 5px;
    background: #73AD21;
    padding: 20px;
    width: 200px;
    height: 150px;
}

</style>

</head>
<body>

<p id="rcorners4"></p>

</body>
</html>
```

სურ. 3.56. CSS3- ის არასტანდარტული ოვალურ კუთხიანი ბლოკის გამოყენების ნიმუში

*კითხვები თვით შეფასებისთვის:*

- შეიძლება თუ არა ბლოკის დონის ტაგს ჰქონდეს ოვალური კუთხეები?
- შეგვიძლია თუ არა ვმართოთ ბლოკის დონის ტაგის ცალკეული კუთხის მომრგვალების რადიუსი?

## ბლოკზე ფერების მინიჭება RGBA ფორმატით CSS3-ის მეშვეობით

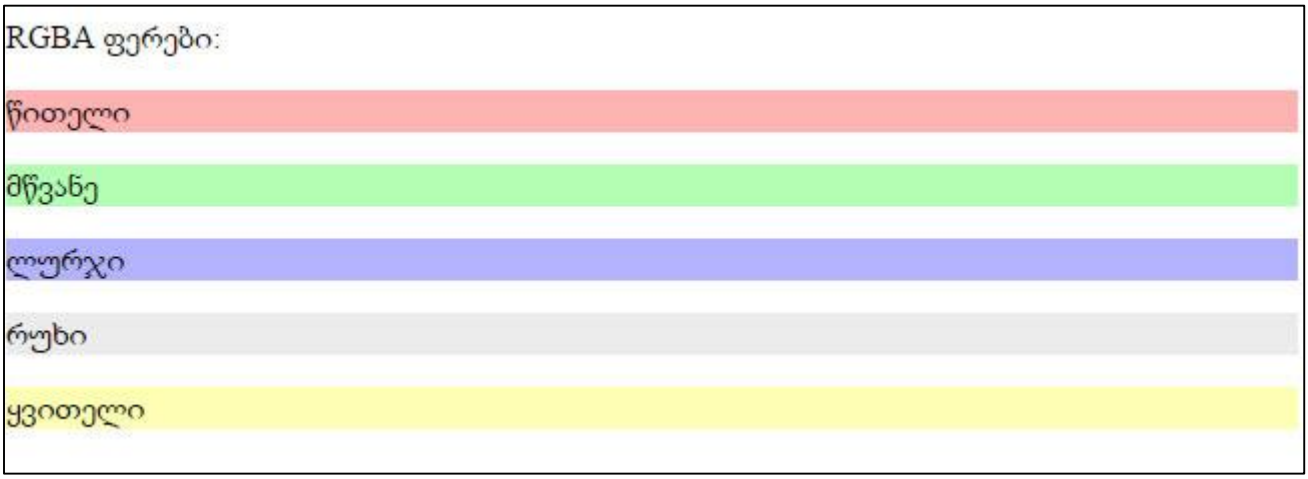
RGBA- ფერები, CSS3-ში წარმოადგენენ ჩვენთვის კარგად ნაცნობი RGB (Red Green Blues) გაფართოებას ვინაიდან მას დამატებული აქვს ალფა კანალი ანუ ფერის გამჭვირვალეობის ხარისხი. გამჭვირვალეობის ხარისხი იცვლება 0.0-იდან 1.0-მდე. ანუ ალფა კანალის მნიშვნელობა 1.0 არის აბსოლუტურად გაუმჭვირვალე ხოლო 0.0 კი აბსოლუტურად გამჭვირვალე.

მისი სინტაქსი შემდეგნაირია:

#p1 {background-color:rgba(255,0,0,0.3);}

პირველი მნიშვნელობა, ჩვენს შემთხვევაში 255 წარმოადგენს წითელ ფერს, მეორე წარმოადგენს მწვანეს და მესამე ლურჯს, ხოლო მეოთხე მნიშვნელობა 0.3 წარმოადგენს გამჭვირვალეობის ხარისხს.

```
<!DOCTYPE html>
<html>
<head>
<style>

#p1 {background-color:rgba(255,0,0,0.3);}
#p2 {background-color:rgba(0,255,0,0.3);}
#p3 {background-color:rgba(0,0,255,0.3);}
#p4 {background-color:rgba(192,192,192,0.3);}
#p5 {background-color:rgba(255,255,0,0.3);}

</style>
</head>
<body>

<p>RGBA ფერები:</p>
<p id="p1">წითელი</p>
<p id="p2">მწვანე</p>
<p id="p3">ლურჯი</p>
<p id="p4">რუხი</p>
<p id="p5">ყვითელი</p>


</body>
</html>
```




სურ. 3.57. CSS3- ის მეშვეობით RGBA ფერების გამოყენების ნიმუში

*კითხვები თვით შეფასებისთვის:*

- რას წარმოადგენს RGBA ფერები?
- რა ფორმატით უნდა მივუთითოთ RGBA ფერები?

### ჩრდილის ეფექტის დადება ბლოკზე და ტექსტზე CSS3-ის მეშვეობით

CSS3 ის მეშვეობით შეგვიძლია დოკუმენტის ელემენტებს, კერძოდ ბლოკის ტიპის ელემენტებს და ტექსტებს დავადოთ ჩრდილის ეფექტი. ამ თვისებებს ვმართავთ შემდეგი CSS3-ის მითითებებით.

- text-shaodow
- box-shadow

პირველ ჯერზე განვიხილოთ ყველაზე მარტივი მაგალითი ტექსტზე ჩრდილის ეფექტის მინიჭებისა:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-shadow: 2px 2px;
}
</style>
</head>
<body>

<h1>ტექსტი ჩრდილის ეფექტით!</h1>

</body>
</html>
```


სურ. 3.58. CSS3- ის მეშვეობით ტექსტზე ჩრდილის ეფექტის გამოყენების ნიმუში

ანუ აღნიშნული ეფექტის გამოყენება CSS3 -ში საკმაოდ მარტივია, პირველი მნიშვნელობა განკუთვნილია ჰორიზონტალური ჩრდილის ზომის მისათითებლად, ხოლო მეორე კი, ვერტიკალურის.

აგრეთვე არსებულ თვისებას თუ დავუმატებთ მესამე მნიშვნელობას ანუ ჩრდილის ფერს, მივიღებთ შემდეგ შედეგს:

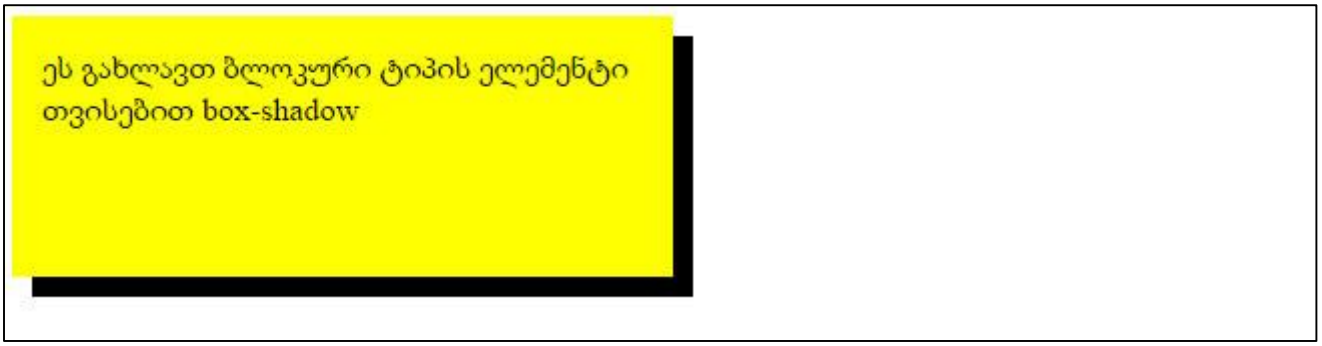
```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    text-shadow: 2px 2px red;
}
</style>
</head>
<body>

<h1>ტექსტი ჩრდილის ეფექტით, სადაც ჩრდილს ფერი აქვს მინიჭებული!</h1>
</body>
</html>
```


სურ. 3.59. CSS3- ის მეშვეობით ტექსტზე ჩრდილის და ჩრდილის ფერის ეფექტის გამოყენების ნიმუში

ზემოთ მოყვანილ მაგალითში ნათლად ჩანს, რომ ტექსტის ჩრდილის ფერი გახლავთ წითელი ფერის.

ახლა რაც შეეხება ბლოკური დანარჩენი ელემენტებს და მათზე ჩრდილის დადების ეფექტს. პრინციპი ამ შემთხვევაში უცვლელი რჩება, უბრალოდ არა ტექსტური ელემენტებისთვის ვიყენებთ თვისებას box-shaodow

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 300px;
    height: 100px;
    padding: 15px;
    background-color: yellow;
    box-shadow: 10px 10px;
}
</style>
</head>
<body>

<div>ეს გახლავთ ბლოკური ტიპის ელემენტი თვისებით box-shadow</div>

</body>
</html>
```

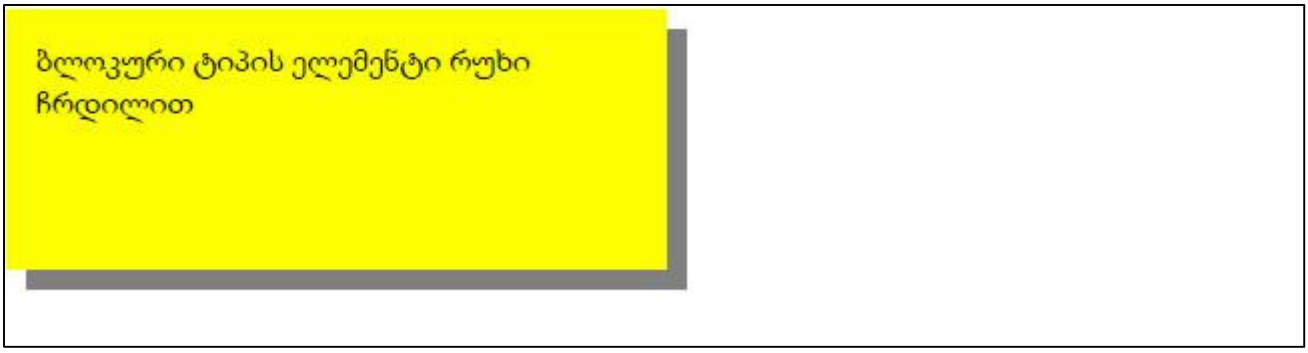



სურ. 3.60. CSS3- ის მეშვეობით ბლოკური ტიპის ელემენტზე ჩრდილის ეფექტის გამოყენების ნიმუში

ანუ, როგორც უკვე აღვნიშნეთ ვუთითებთ ჰორიზონტალური და ვერტიკალური ჩრდილის ზომებს, შენიშნავდით ალბათ, გაჩუმების პრინციპით ჩრდილის ფერი გახლავთ შავი.

ანალოგიურად ტესტის ჩრდილის ეფექტისა, ასევე შეგვიძლია შევცვალოთ box-shadow[-ს ჩრდილის ფერიც.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 300px;
    height: 100px;
    padding: 15px;
    background-color: yellow;
    box-shadow: 10px 10px grey;
}
</style>
</head>
<body>

<div>ბლოკური ტიპის ელემენტი რუხი ჩრდილით</div>

</body>
</html>
```




სურ. 3.61. CSS3- ის მეშვეობით ბლოკური ტიპის ელემენტზე ჩრდილის ეფექტის გამოყენების ნიმუში

*კითხვები თვით შეფასებისთვის:*

- შესაძლებელია თუ არა შევუცვალოთ ბლოკური ტიპის ელემენტს ჩრდილის ფერი?

- შესაძლებელია თუ არა შევუცვალოთ ტექსტური ტიპის ელემენტს ჩრდილის ფერი და ზომა?

### ანიმაცია CSS3-ის მეშვეობით

CSS3-ის მეშვეობით ვებ გვერდზე შეგვიძლია შევქმნათ ანიმაცია, ყოველგვარი JavaScript-ისა და Flash-ის დახმარების გარეშე. ანიმაციის შექმნის მთავარი პრინციპი მდგომარეობს იმაში, რომ უნდა წინასწარ განვსაზღვროთ საგასაღებო კადრები მოცემულ ანიმაციაში. ანუ თუ რა თვისებები უნდა შეეცვალოს ობიექტს, მისი საწყისი მნიშვნელობები და საბოლოო.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 100px;
    height: 100px;
    background-color: red;
    -webkit-animation-name: example; /* Chrome, Safari, Opera */
    -webkit-animation-duration: 4s; /* Chrome, Safari, Opera */
    animation-name: example;
    animation-duration: 4s;
}

/* Chrome, Safari, Opera */
@-webkit-keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

სურ. 3.62. CSS3- ის მეშვეობით ანიმაციის  გამოყენების ნიმუში

ზემოთ მოყვანილ კოდში განსაზღვრულია, ანიმაციის ხანგრძლივობა 4 წამი და მითითებულია, რომ ბლოკური ტიპის ელემენტის ფონის ფეი უნდა შეიცვალოს წითლიდან ყვითელზე. კიდევ ერთხელ გვინდა ხაზგასმით ავღნიშნოთ, რომ CSS3-ის ანიმაციის შემთხვევაში მისი თვისებების ცვლილებაში შეზღუდულნი არ ვართ, ანუ ანიმაციის ობიექტად შეიძლება ვიქონიოთ რომელიმე ელემენტის ჩარჩოს ფერი ან ზომა, თუნდაც მისი პოზიცია.

## 7.5. საიტის ინტერაქტივისა და ეფექტების გამოყენება - java script

დღევანდელი ბაზრის ანალიტიკა ცხადყოფს ვებ ტექნოლოგიების სწრაფ განვითარებას, რაც თავისთავად მიბმულია ტექნოლოგიურ განვითარებასთან. ამ ყოველივესთან პროპორციულია ვებ სტანდარტი, რომელიც კონკრეტული დროის მოთხოვნების მიხედვით იცვლება.

სტატიკური ვებ გვერდების ერა დიდი ხანია დასრულდა. დღეს მსოფლიოს დიდი ნაწილი ორიენტირებულია დინამიური საიტების, სხვადასხვა ეფექტებითა და ინტერაქციით გაფორმებული საიტების შექმნაზე.

ვებსაიტზე ინტერაქტივისა და ეფექტების შექმნის საშუალებას იძლევა Javascript-ი. დეველოპერების უმეტესობა იშვიათად გამოიყენებს დიდი მოდულის შესაქმნელად სტანდარტული Javascript კოდის ნულიდან წერის მეთოდს. ეს ყოველივე დროის დიდ კარგვად მიიჩნევა.

არსებობს Javascript_ის უამრავი ბიბლიოთეკა, რომელიც ამარტივებს ბევრად მუშა პროცესს, რადგანაც ორიენტირებულნი არიან კონკრეტული მიმართულებების გასამარტივებლად. რა თქმა უნდა ყველა ბიბლიოთეკის ცოდნა წარმოუდგენელია, უბრალოდ საჭიროა გქონდეთ წარმოდგენა მათ შესაძლებლობებზე, იცოდეთ მოძიება, კოდის სტრუქტურის შესაბამისად ინტეგრირება html დოკუმენტში, მათი სტილიზება და შემცველობის ცვალებადობა.

არსებული მოდულში განხილული იქნება ბიბლიოთეკის მუშაობის პრინციპები, მათი მოძიება / ინტეგრაცია. მაგალითები შესაძლებლობას მოგცემთ უკეთ გაიგოთ ბიბლიოთეკების არსი და დანიშნულება.

პირველ რიგში დეტალურად უნდა მოხდეს მიღებული დავალების ანალიზი, განისაზღვროს ცალკეული დეტალი. შემდეგ მოხდეს დავალების შესაბამისი სხვადასხვა ბიბლიოთეკის მოძიება და განხილვა. მოძიებული ბიბლიოთეკების განხილვის საფუძველზე შეირჩეს ოპტიმალური ვარიანტი და მოხდეს მისი რედაქტირება, რათა მივიღოთ დავალების იდენტური Javascript კოდი.

## 7.6. ბიბლიოთეკის კოდის სტრუქტურის გაცნობა

**მიმდინარე პარაგრაფის თემატიკა**

- დავალების დეტალების ანალიზი
- თანამედროვე ბიბლიოთეკების მოძიება/გადმოწერა
- ბიბლიოთეკის კოდის სტრუქტურის აღწერა

ნებისმიერი საქმის წარმატებით შესრულებისათვის მისი სწორი აღქმა და დაგეგმარებაა საჭირო. ვებ საიტის ფორმირებისას აუცილებელია საწყის ეტაპზე მოხდეს მიღებული დავალების სწორი ანალიზი. ამის შემდგომ შეირჩეს მისი შექმნისათვის ხელსაყრელი ტექნოლოგიები.

თუ ამ დებულებით ვიხელმძღვანელებთ ვებ გვერდზე ეფექტებისა და ინტერაქციის განთავსების პროცესს იდენტური მიდგომა სჭირდება. საჭიროა ზედმიწევნით ზუსტად იქნას აღქმული დავალების სირთულე და მისი გადაჭრის გზები, მხოლოდ შემდგომი ნაბიჯია Javascript ბიბლიოთეკის მოძიება კონკრეტული დავალების შესასრულებლად. ამ შემთხვევაში კი აშკარად დაგჭირდებათ „სასიცოცხლო მნიშვნელობის“ ზოგადი განათლება ამ მიმართულებით.

და რატომ ზოგადი განათლება?
რა თქმა უნდა ფუნდამენტურ ცოდნის წინააღმდეგი არავინაა, მაგრამ მისი შეძენა ვებ დეველოპინგში მრავალწლიან თავდაუზოგავ შრომას და დიდ პრაქტიკას მოითხოვს. ასევე არსებობს ერთი „მაგრამ“. შესაძლოა კონკრეტულ ტექნოლოგიაში საკმაოდ წარმატებულნი იყოთ, მაგრამ სასურველია აუწყოთ ფეხი სიახლეებს.  აუცილებლია ზოგადი ცოდნა გარკვეული ტექნოლოგიის არსებობის, მისი შესაძლებლობის შესახებ, არსი თუ სად შეიძლება მისი გამოყენება, რა კონკრეტული საქმის გაადვილებას ემსახურება ეს ტექნოლოგია. სწორედ ეს მოიაზრებოდა ზოგად განათლებაში.

ტექნოლოგია, რომელიც შეიძლება ერთობ უსარგებლო გეგონოთ, შესაძლოა საიტის შემუშავების პროცესში რაღაც ეტაპზე დიდი დახმარება გაგიწიოთ. მაგრამ დამერწმუნეთ თუნდაც უსარგებლო ინფორმაციის ფლობა თავიდან აგარიდებთ ბევრი გამოუსადეგარი ტექნოლოგიის გამოყენებისაგან.

Javascript ბიბლიოთეკების რაოდენობა საკმაოდ ვრცელია. მათზე ინფორმაციის ყველაზე მარტივი გზა მიმდინარე ეტაპზე პოპულარული ბიბლიოთეკების სტატისტიკის მონახულებაა, რომელიც არსებულ დროში მოთხოვნად რესურსზე მოგცემთ სრულ წარმოდგენას. ამ ყოველივეს გაგიადვილებთ თქვენი თვითგანვითარების დაგეგმავს და დაგეხმარებათ ორიენტირის განსაზღვრაში.

რა თქმა უნდა არსებული რეიტინგების საძიებო სისტემაში მიგნება დიდ სირთულეს არ წარმოადგენს, მაგრამ არსებობს ერთი მეტად საინტერესო პროექტი bower.  ოფიციალური ვებ გვერდია http://bower.io

Bower –twitter_ის პროდუქტია, რომელიც წარმოადგენს პაკეტების მართვის მენეჯერს. ის უზრუნველყოფს საიტებზე ბიბლიოთეკების მოძიებას, ჩამოწერა, განარქივებას, პროექტის ფაილში კოდის ინტეგრაციას, დანამატების ( plugin ) მოძიებას, ინსტალაციას, მათ განახლებებზე ზრუნვას და ა.შ.

სხვადასხვა პროგრამირების ენას თავისი მენეჯერები გააჩნიათ ბიბლიოთეკების მოძიება/დაყენებისთვის. მაგ.: gem წარმოადგენს Ruby_ის მენეჯერს, pip– Python და ა.შ.

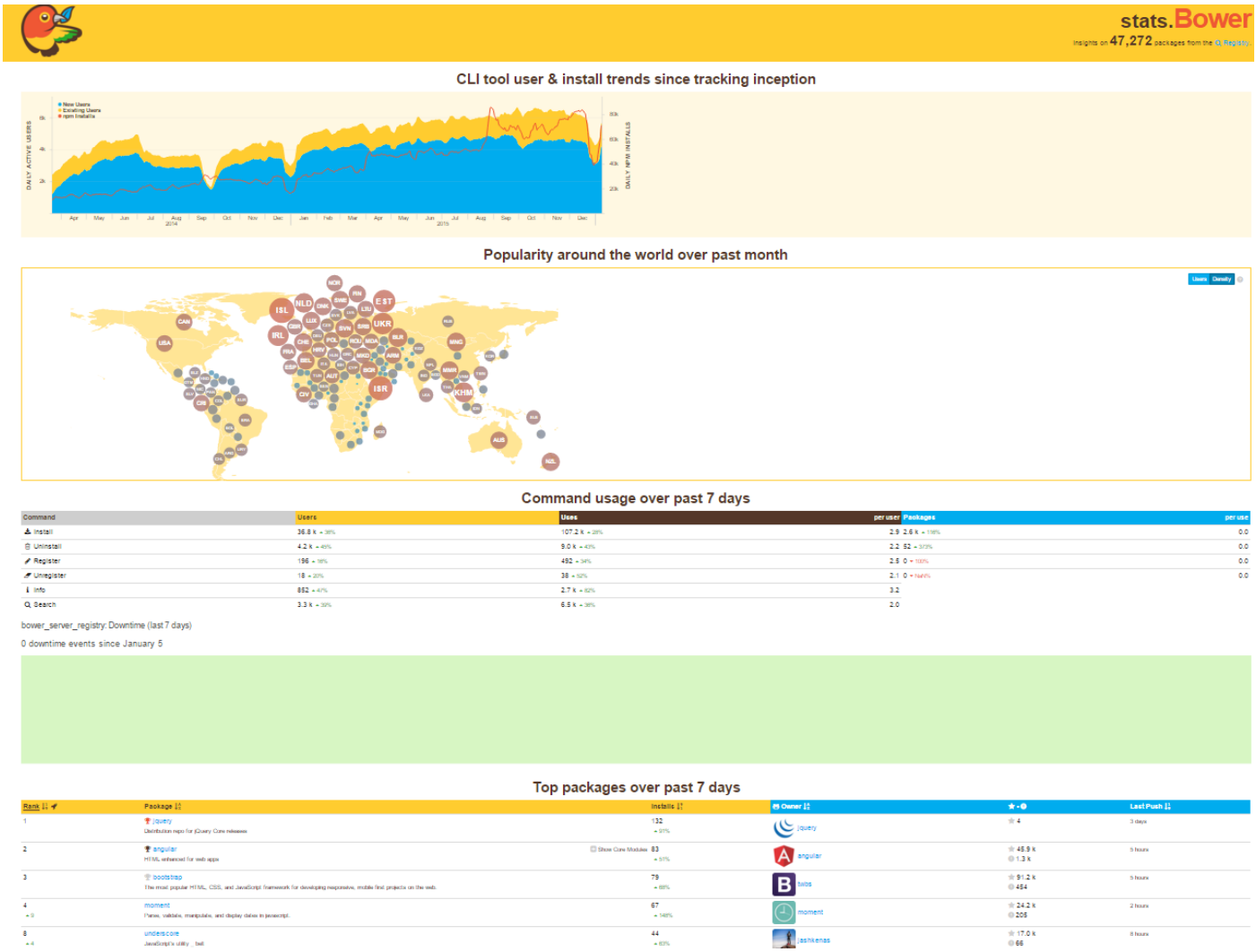
BowerJavascript_ის სტანდარტულ მენეჯერად არ მოიაზრება, მაგრამ  ის საკმაოდ დიდი პოპულარობით სარგებლობს - ის აერთიანებს დაახლოებით 11 ათას პაკეტს.

მისი დეტალური ანალიზი - დაყენება / შემდგომი მოხმარება არსებულ კურსში ახსნილი არ იქნება, მაგრამ გირჩევთ გაამახვილოთ ყურადღება არსებულ პროდუქტზე.

Bower_ის ოფიციალურ ვებ გვერდზე ჩანართი stats( http://bower.io/stats/ ) უზრუნველყოფს თქვენთვის სტატისტიკური მონაცემების მოწოდებას, თუ რომელი ბიბლიოთეკებიდან Framework_ია არსებულ დროში ყველაზე მოთხოვნადი და საძიებო სისტემების გვერდის ავლით იღებთ სანდო ინფორმაციას. (იხ.სურ. 4.1)

რით განსხვავდება ბიბლიოთეკა და Framework ერთმანეთისაგან?

ბიბლიოთეკა წარმოადგენს პროგრამულ მოდულს დაწერილს გარკვეულ პროგრამირების ენაზე, ხოლო Framework ბიბლიოთეკებისა და ინსტრუმენტების ერთობლიობაა.




4.1 სტატისტიკა javascript_ის ბიბლიოთეკების რეიტინგის შესახებ

უკანასკნელი 7 დღის მონაცემები რეალურად წლების მანძილზე ნაგროვები სტატისტიკური მონაცემებია. როგორც მოცემულობიდან ჩანს მოწინავე პოზიციებს შემდეგი ბიბლიოთეკები იკავებენ:

- jQuery;
- Moment;
- Underscore;

გამოტოვებული მონაცემები წარმოადგენენ Framework_ებს და მათი განხილვა ამ ეტაპზე არამიზანშეწონილია.

რეიტინგში მოცემულია 100 ელემენტიანი ჩამონათვალი და რა თქმა უნდა არ წარმოადგენს სრულ სიას. შესაძლოა რამდენიმე მათგანი ერთმანეთის მსგავსად მუშაობდეს, მაგრამ უმეტეს შემთხვევებში მათი ფუნქციონალი განსხვავებულია.

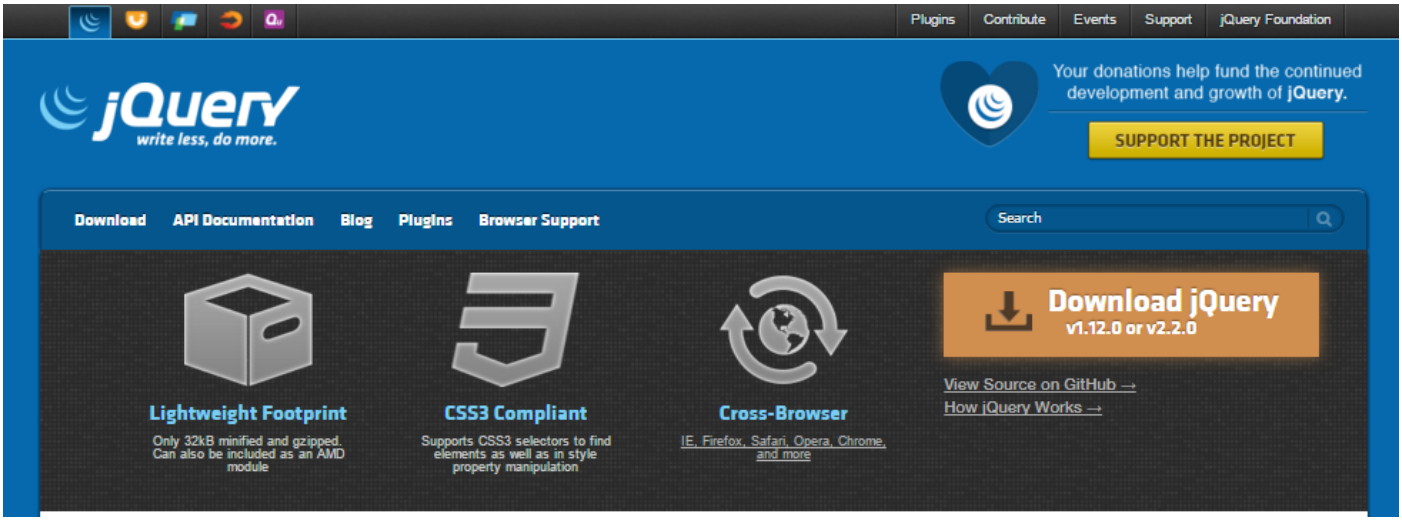
ბიბლიოთეკების საერთო სტილში შესაძლოა მათი დოკუმენტში ინტეგრაცია მივიჩნიოთ. თუ bower მენეჯერი არ იქნა გამოყენებული, მაშინ უმჯობესია ბიბლიოთეკების მოძიება/გამოყენება უშუალოდ მათი ოფიციალური საიტებიდან. გამომდინარე იქიდან, რომ ორგანიზაციები პერიოდულად ახდენენ ბიბლიოთეკის კოდის/ვერსიის განახლებას, საკუთარ საიტზე ყოველთვის უახლეს პროდუქტს გვთავაზობენ ( იგულისმხება უკვე გატესტილი და წარმოებაში ჩაშვებული ნაწარმი ).

განვიხილოთ გზა, როცა ბიბლიოთეკის განთავსება უნდა მოხდეს თქვენივე სერვერზე, ე.ი. საჭიროება იქნება ბიბლიოთეკის ჩამოწერისა. გლობალური მისამართით მისი დოკუმენტში ინტეგრირება შემდეგ ქვეთავში იქნება ახსნილი:

### jQuery

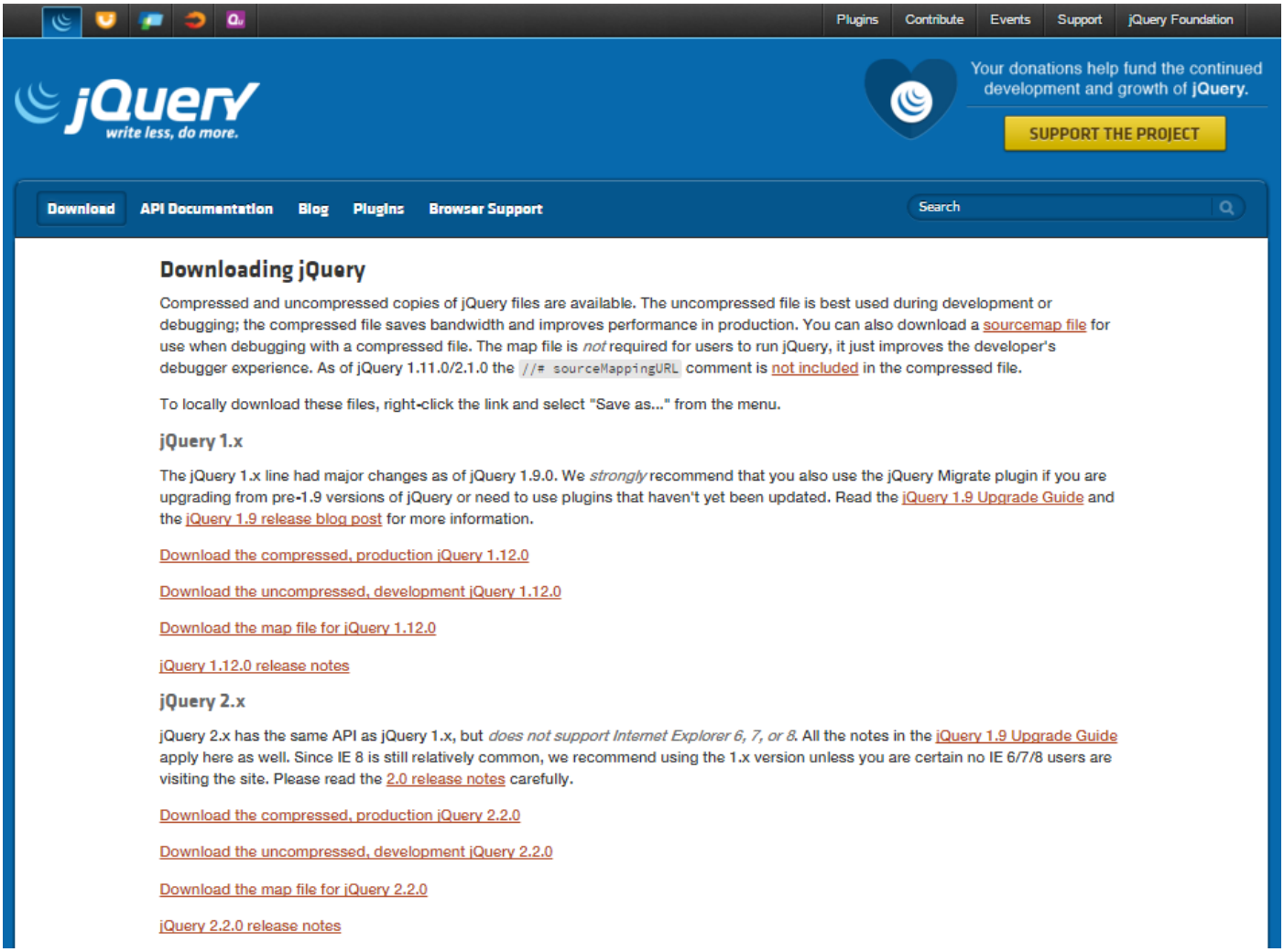
არსებული ბიბლიოთეკა რეიტინგის სათავეშია მოქცეული. ის ყველაზე ფართოდ გამოყენებადია მსოფლიოში და ძირითადად ორიენტირებულია ვებ გვერდზე არსებული ინფორმაციის ეფექტებისა და ანიმაციის შემუშავებაზე. მისი ფუნქციების გამოყენებით მარტივადაა შესაძლებელი html დოკუმენტის ჩონჩხის ფორმირება-ელემენტების შექმნა, წაშლა, კონკრეტული ბლოკების ხილვადობის რეგულირება, მომხმარებლებთან ურთიერთობის მარტივი შესაძლებლობები ( cookie ).

ბიბლიოთეკა jQuery_ის ოფიციალურ ვებ გვერდზე შესაძლებელია მისი მოძიება ჩამოწერა - http://jquery.com/ (იხ.სურ.4.2)




4.2 ბიბლიოთეკა Jquery ოფიციალური ვებ გვერდი

Download jQuery ბმულის მეშვეობით შესაძლებელია პროდუქტის ჩამოსაწერი მისამართის აქტივაცია. (იხ.სურ.4.3) ამჟამად ხელმისაწვდომია ორი ვერსია: 1.x და 2.x. გარდა იმისა რომ 2.x ვერსია ცხადია უფრო ახალი და დახვეწილია, მასში ასევე შეჩერებულია Internet Explorer 6,7 და 8 ვერსიების მხარდაჭერა. ასე რომ თუ თქვენთვის ჯერ კიდევ აქტუალურია ეს მოძველებული ( ფაქტობრივად ხმარებიდან ამოღებული ) ბრაუზერები, ჯობია ისევ 1.x-ზე შეჩერდეთ.




4.3 jQuery ბიბლიოთეკის ჩამოსაწერი ოფიციალური მისამართი

ბიბლიოთეკა დევს ჩვეულებრივი და მინიმიზებული, ანუ შეკუმშული სახით (შესაბამისად uncompressed და compressed). ჩვეულებრივი ვარიანტის გამოყენება მოსახერხებელია დეველოპმენტის პროცესში, თუ სპეციალისტი ერევა უშუალოდ კოდში და საჭიროებს მის შეცვლას. ვიზუალიც შესაბამისად გაცილებით ლამაზი სანახავია, რაც შეეხება კომპრესირებულ ვერსია ის მეტად მობილურია, ზომაში საკმაოდ მცირეა, რაც უზრუნველყოფს ბიბლიოთეკის შედარებით მეტი სისწრაფით ჩატვირთვას, ამიტომ მასზე დიდი მოთხოვნაა. ძირითად შემთხვევებში სწორედაც, რომ ბიბლიოთეკის მინიმიზებული ვერსია გამოიყენება.

## Moment js

დროის მონაცემების დამუშავების პროცესი, როგორც დამწყები ასევე გამოცდილი პროგრამისტისათვის Javascript_ში მარტივ საქმეს არ წარმოადგენს. შესაბამისად არსებობს ბიბლიოთეკა **Moment js,** რომელიც დეველოპერებს საგრძნობლად უმსუბუქებს არსებული ტიპის მონაცემებთან მუშაობას.

**Moment js** ეს არის ბიბლიოთეკა **ღია კოდით (** open source), რომელიც გამოიყენება დროის ობიექტებთან სამუშაოდ მათი ფორმატირებისა და დაჭრისთვის ( პარსირება ). იგი საშუალებას აძლევს პროგრამისტს აირიდოს პირდაპირი შემხებლობა Javascript Date ობიექტებზე. ეს კი ნამდვილი შვებაა ☺.

**Moment js** იძლევა Javascript-Date ჩაშენებული ფუნქციების გაფაროების შესაძლებლობებს. მაგ.: კალენდარული დრო, მრავალ ენოვანი მხარდაჭერა დროის მხარდაჭერა და ა.შ.

არსებული ბიბლიოთეკის მოძიება და ჩამოწერა შესაძლებელია მისივე ოფიციალურ ვებ გვერდზე http://momentjs.com/ (იხ.სურ.4.4)


4.4 moment js მოძიება ჩამოწერა

ჩამოსაწერი ვერსიებიდან ერთი სუფთა moment.js_ია ხოლო moment+locates.js ფაილში დამატებული რეგიონული ენები. თუ მსგავსი საჭიროება არ დგას უმჯობესია გამოყენებულ იქნას პირველი ვერსია ზომაში შედარებით მინიმიზებულია.

ორივე ფაილი წარმოადგენს Javascript გაფართოების ფაილს. მათი ჩამოწერისას საიტის გადამისამართება ხდება სხვა ფანჯარაში კოდის სახით (იხ.სურ.4.5). **ctrl+s** კლავიშების კომბინაციით შესაძლებელია არსებული კოდის შენახვა ლოკალურ მისამართზე მომავალი გამოყენებისათვის.

```
//! moment.js
//! version : 2.11.1
//! authors : Tim Wood, Iskren Chernev, Moment.js contributors
//! license : MIT
//! momentjs.com

;(function (global, factory) {
    typeof exports === 'object' && typeof module !== 'undefined' ? module.exports = factory() :
    typeof define === 'function' && define.amd ? define(factory) :
    global.moment = factory()
}(this, function () { 'use strict';

    var hookCallback;

    function utils_hooks__hooks () {
        return hookCallback.apply(null, arguments);
    }
```

### 4.5 moment js ფაილის შემცველობა

### Underscore

Javascript სამყაროში ერთ-ერთი ფართოდ გამოყენებადი ბიბლიოთეკა underscore_ია. მისი მეშვეობით შესაძლებელია კოლექციებთან, ცვლადებთან , ობიექტებთან, ფუნქციებთან, მასივებთან და ა.შ გაფართოებული და გამარტივებული ფუნქციებით მუშაობა.

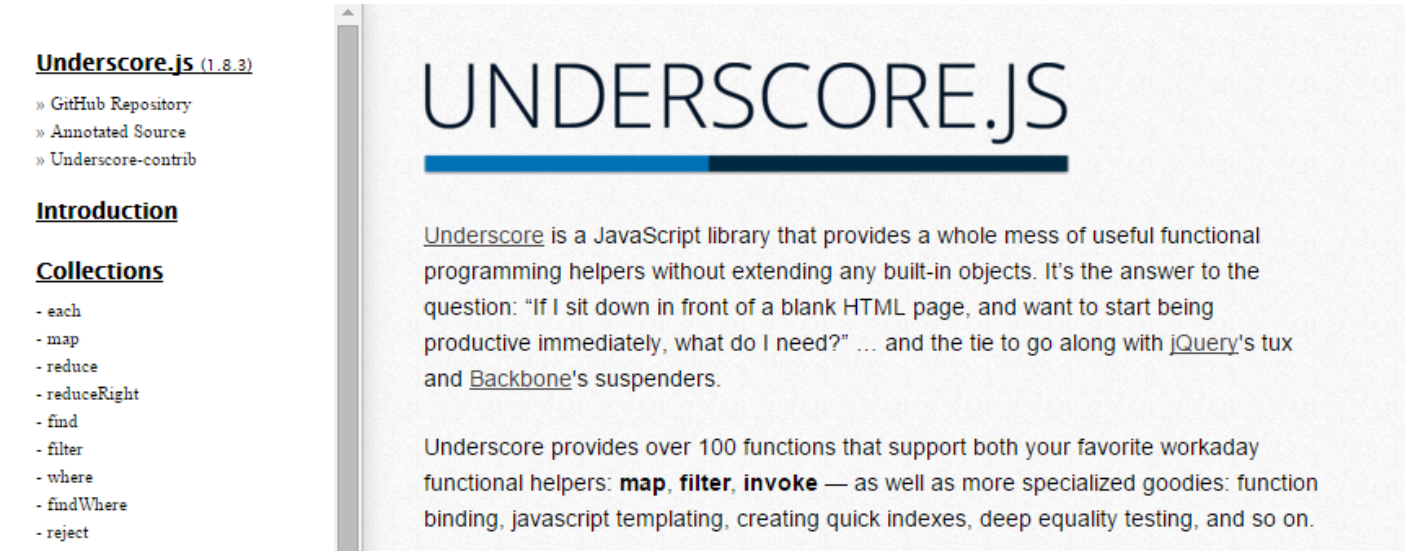
მითითებული ბიბლიოთეკის ოფიციალური მისამართია http://underscorejs.org/ . (იხ.სურ.4.6) არსებულ მისამართზე შესაძლებელია უახლესი ვერსიის ჩამოწერა. მიმდინარე ეტაპზე 1.8.3 ვერსია არეკომენდირებული საიტის მიერ.

როგორც წინა შემთხვევებში არსებული ბიბლიოთეკაც წარმოდგენილია ორი სახით:

- Development Version(1.8.3)- სტანდარტული ვერსია
- Production Version (1.8.3) - მინიმიზებული ვერსია

მათი შერჩევა და ჩამოწერა თქვენს ნებასურვილზეა დამოკიდებული. ჩამოწერა იმავე სტილით ხდება, როგორიც აღწერილია moment js ბიბლიოთეკის შემთხვევაში.

მაგ.: შეადარეთ 1_დან 7_მდერიცხვების გამოტანა ეკრანზე - ჩანაწერი სტანდარტული Javascript_ის და




4.6 ბიბლიოთეკა underscore js ოფიციალური ვებ გვერდი

underscorejs ბიბლიოთეკის გამოყენებით;

```
// სტანდარტული  js  ------ -გზა I

alert(1);
alert(2);
alert(3);
alert(4);
alert(5);
alert(6);
alert(7);

// სტანდარტული  js  ------ -გზა II

for(var i =1; i <=7; i++){
alert(i);
}



// ბიბლიოთეკაunderscore js
_.each([1,2,3, 4, 5, 6, 7], alert);
```

როგორც თემის დასაწყისში აღვნიშნეთ არსებული ბიბლიოთეკები უმარტივებენ დეველოპერს კოდთან ურთიერთობას, ამიტომ მათი ინტენსიური გამოყენება ზედმიწევნით წაგადგებათ საქმიანობაში. მიუხედავად რიგი დადებით თვისებებისა სხვადასხვა ბიბლიოთეკების, მოცემულ შემთხვევაში ორიენტირებულნი ვიქნებით jQuery ბიბლიოთეკაზე და შესაბამისად მაგალითების განხილვა ამ ბიბლიოთეკასთან კავშირზე იქნება დამყარებული.

რატომ მაინც და მაინც jQuery?

jQuery ორიენტირებულია უფრო ელემენტების ვიზუალიზაციაზე, მათ გაფორმებაზე, html ტეგებზე, სტრუქტურაზე. გამომდინარე აქედან უმჯობესია საიტის ეფექტებით სრულყოფა მოვახდინოთ სწორედ არსებული ბიბლიოთეკის გამოყენებით, უფრო კონკრეტულად რომ ვთქვათ გარჩეული იქნება jQuery_ის გამოყენებით შექმნილი დამატებითი მოდულები - plugin_ები.

ბიბლიოთეკის მიბმა დოკუმენტთან არ ნიშნავს იმ შედეგის ავტომატურ მიღებას რაც სასურველია. რიგ შემთხვევაში დეველოპერები თავად ცდილობენ კოდის დაშენებას არსებული ბიბლიოთეკის ბრძანებების მეშვეობით, მაგრამ სიმარტივისთვის უმეტესობა უკვე არსებულ პროდუქტს იყენებს.

ხშირია კონკრეტული მოდულის უშუალოდ ბიბლიოთეკად მოხსენიება, მაგრამ დოკუმენტალურად ეს ასე არ არის მაგ.: jQueryDropDownMenu ან jQuerySlideShow წარმოადგენენ **jQuery ბიბლიოთეკის** დამატებებს (plugin).

სხვადასხვა ბიბლიოთეკების დანამატების მოძიება ყველაზე მარტივად საძიებო სისტემების მეშვეობითაა შესაძლებელი. მაგალითად jQuery ბიბლიოთეკის ოფიციალურ გვერდზე არსებობს ცალკე ჩანართი**plugins**http://plugins.jquery.com/ ,სადაც ასევე შესაძლებელია მოცემული ბიბლიოთეკის დანამატების მიგნება / ჩამოწერა.

საიტების ნაწილზე ბიბლიოთეკის დანამატები წარმოდგენილია არქივირებული სახით, რაც საკმაოდ კომფორტულია. გამომდინარე იქიდან, რომ ყველა საჭირო ფაილია ერთადაა მოქცეული ეს საკმაოდ ამარტივებს დეველოპერის საქმეს. შესაძლოა ვებ გვერდზე სასურველი plugin მოცემული იყოს კოდის სტრუქტურის სახით. ეს არის შემთხვევა, როდესაც კოდის ინტეგრირებისათვის დაგჭირდებათ ან ის გლობალური მისამართები რომლებიც გამოყენებულია საიტზე, ან ინდივიდუალურად მათზე გასვლა / ჩამოწერა და სწორი ინტეგრირება.

როგორც აღვნიშნეთ მზა პროდუქტი ( plugin ) ბევრად ამარტივებს საქმიანობას. მასში მინიმალური ჩარევით შესაძლებელია იმ ეფექტის მიღება, რომლის შექმნას შესაძლოა საკმაოდ დიდი დროითი რესურსის დახარჯვა დასჭირდეს. დეველოპერს ძირითად შემთხვევაში მაინც სტილურ ფაილებთან ურთიერთობა უწევს, რათა გადააწყოს არსებული მოდული მისთვის შესაფერის ჩარჩოზე, შეუცვალოს ფონები, ფერები, ზომები. სტილური ფაილი ერთ ერთმნიშვნელოვანი შემადგენელი ნაწილია ბიბლიოთეკის მოდულის ( plugin )კოდის სტრუქტურისა. გარდა სტილური ფაილისა შესაძლოა მოცემულ მოდულს მოსდევდეს იმ სურათების ერთობლიობა, რომელიც საიტზე შემცველობის გამოსახულებად (img) ან ფონებად (background-image)გამოიყენებოდეს. ზოგადად ვახსენეთ თანდართული სკრიპტები.

შესაძლებელია სკრიპტები თავმოყრილი იყოს ცალკე Javascript ფაილად ( **js** გაფართოებით ). ასევე არ არის გამორიცხული კოდი ან მისი რაიმე ნაწილი უშუალოდ html დოკუმენტის კონკრეტულ გვერდზე იყოს განსათავსებელი. ძირითადად ასეთი ტიპის სკრიპტის დანამატები, შეიცავს პარამეტრების კონფიგურაციის შემცველ კოდს და საკმაოდ მოსახერხებელია სხვადასხვა ეფექტების სამართავად (მიმართულების, სისწრაფის, რაიმე ბლოკის ხილვადობა / დამალვის და ა.შ ).

ჩვენს მიერ აღწერილი თითოეული დეტალი აზრს დაკარგავდა იმ html კოდის გარეშე, რომელიც აუცილებდა უნდა ერთვოდეს თან ძირითად სტრუქტურას. გამომდინარე Jacascript_ის სპეციფიკიდან ის მუშაობს HTML დოკუმენტთან ( ტეგებთან ), შესაბამისად ეფექტების სრულყოფილი მუშაობისათვის აუცილებელია ტეგების განლაგების მემკვიდრეობის დაცვა, ასევე მათთვის საჭირო სელექტორების მისადაგება (id, class).

კოდის სტრუქტურის უკეთ გასაგებად გაეცანით მოცემულ დავალებას და მის გადაჭრის გზებს.

საწყის ეტაპზე დავგეგმოთ ამოცანა, შევეცადოთ მოვიძიოთ მიმსგავსებული სკრიპტი, მოვახდინოთ მისი ჩამოწერა / ინტეგრირება დოკუმენტში და მორგება საწყის ამოცანაზე.

**დავალება I**

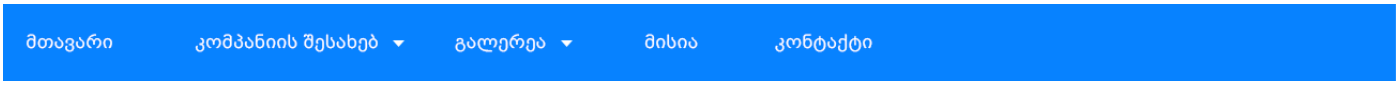
შეეცადეთ ბიბლიოთეკის მეშვეობით განათავსოთ საიტზე ჩამოსაშლელი მენიუ. ნაბიჯები მოცემულია სურათ 4.7 - 4.8 - 4.9_ზე

დავალების ანალიზის შემდეგად მისი წარმატებით შესრულებისათვის საჭიროა რამდენიმე ასპექტის გათვალისწინება:

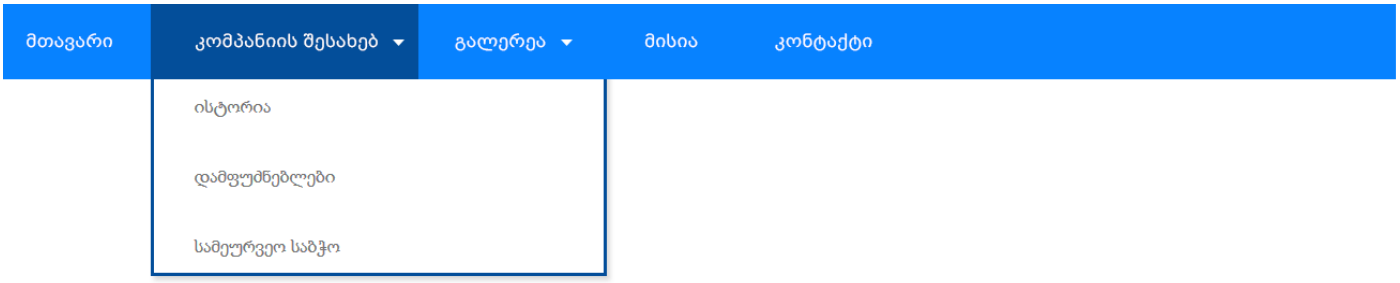
- შერჩეულ იქნას ბიბლიოთეკა;
- მოძიებულ იქნას მსგავსი სახის სკრიპტი;
- ჩამოიწეროს და მოერგოს კოდი უკვე არსებულ html დოკუმენტს;
- მოხდეს ჩარევა ფაილების კოდში, რათა მოერგოს მომხმარებლისთვის სასურველ დიზაინსა;
- შიგთავსი შეივსოს მომხმარებლის სასურველი ინფორმაციით

- მთავარი
- კომპანიის შესახებ
  - ისტორია
  - დამფუძნებლები
  - სამეურვეო საბჭო
- გალერეა
  - ფოტო
  - ვიდეო
- მისია
- კონტაქტი

საწყის ეტაპზე არსებული html კოდის შესაბამისი ვიზუალი ბრაუზერში შემდეგი სახით გამოისახება (იხ.სურ.4.7). დავალების შესაბამისად საჭიროა მისი მოდერნიზაცია და მიღება სურათ 4.8_ზე ნაჩვენები შედეგისა, სადაც პუნქტ „**კომპანიის შესახებ**“ ხელის მიტანისას ვერტიკალურად ჩამოიშლება მენიუს პუნქტები: *ისტორია, დამფუძნებლები, სამეურვეო საბჭო,* ხოლო პუნქტ „**გალერეა** “_ის შემთხვევაში გამოისახება: *ფოტო და ვიდეო* (იხ.სურ 4.9)




4.8 - ჰორიზონტალურად განლაგებული პუნქტები




4.9 - მაუსის კურსორის მიტანისას მიღებული შედეგი

## დავალება II

შეეცადეთ ბიბლიოთეკის მეშვეობით განათავსოთ საიტზე ინტერაქტიული ბლოკი - სლაიდები, რომელიც უზრუნველყოფილი იქნება სანავიგაციო გადამრთველებით და ბულეტებით. ( ვიზუალურად კარგად აღსაქმელად ბულეტები აღებულია პირობითად შავ ფერში ) დავალების შესაბამისი მოცემულობა გამოსახულია სურათ 4.10 – 4.11.

როგორც წინა დავალების განხილვისას გავამახვილეთ ყურადღება მოცემულობის წარმატებით შესრულებისათვის საჭიროა იმავე პუნქტების ზედმიწევნით ზუსტად შესრულება:

- შერჩეულ იქნას ბიბლიოთეკა;
- მოძიებულ იქნას მსგავსი სახის სკრიპტი;
- ჩამოიწეროს და მოერგოს კოდი უკვე არსებულ html დოკუმენტს;
- მოხდეს ჩარევა ფაილების კოდში, რათა მოერგოს მომხმარებლისთვის სასურველ დიზაინსა;
- შიგთავსი შეივსოს მომხმარებლის სასურველი ინფორმაციით


4.10 სტატიკური ბლოკი


4.11 სლაიდერი სანავიგაციო ატრიბუტებით

**დავალება I** წარმატებით განსახორციელებლად მივყვეთ გეგმას, რომელიც თქვენს მიერვე იქნა გაწერილი ორიოდ წუთის წინ და განვახორციელოთ ეტაპობრივად.

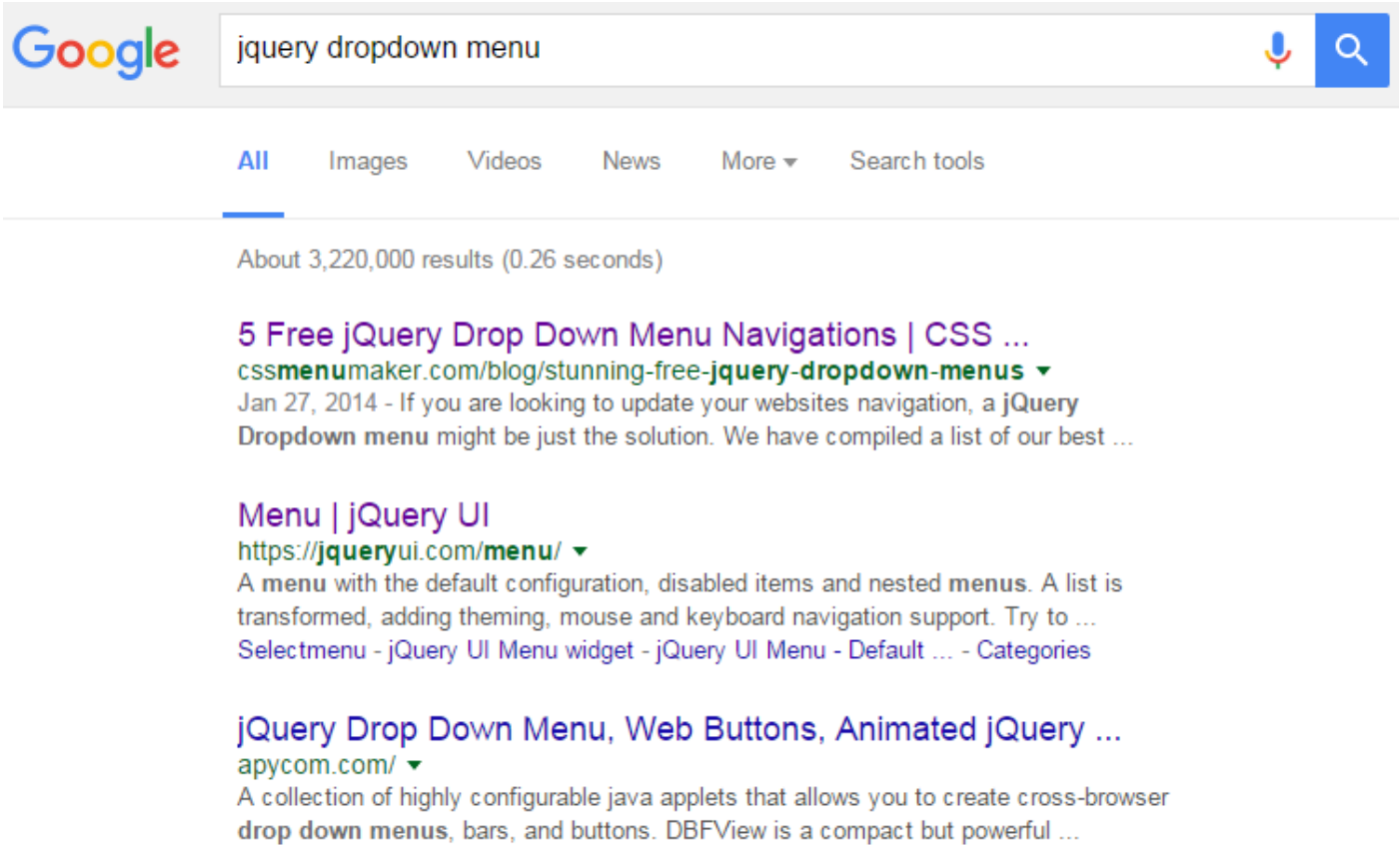
1. შერჩეულ იქნას ბიბლიოთეკა.

გამომდინარე წინა თქმისაგან გამოვიყენებთ ბიბლიოთეკა jQuery_ის.

2. მოძიებულ იქნას მსგავსი სახის სკრიპტი;

დავალებაში მოცემულია ჩამოსაშლელი მენიუ, გამოსაყენებელი ბიბლიოთეკაა jQuery, დარჩა მიმსგავსებული სკრიპტის მოძიება.

საძიებო სისტემაში ვცდილობთ jQuery drop down menu _ით მითითებული კოდის მოძებნას. ( იხ. სურ. 4.12 ). სხვადასხვა ეტაპზე საწყის პოზიციებზე რა თქმა უნდა სხვადასხვა საიტები განლაგდება და ამაში არაფერია გასაკვირი. არც ისაა დოგმად რომ აუცილებლად ყველაზე რეიტინგული საიტებიდან უნდა მოხდეს თქვენთვის სასურველი კოდის ჩამოწერა. მთავარია მიაგნოთ მსგავს სკრიპტს რათა მცირე დროში მიიღოთ მაქსიმალური შედეგი.




4.12 ჩამოსაშლელი მენიუს მოძიება

პირველი ბმულის http://cssmenumaker.com/blog/stunning-free-jquery-dropdown-menus მონახულების შემდგომ ჩემს მიერ შერჩეული იქნა მიმსგავსებული ვარიანტი. სასურველი მენიუს დეტალური გვერდზე წარმოდგენილია პროდუქტის Demo - საჩვენებელი ვერსია რაც საკმაოდ მიმარტივებს არჩევანის სისწორეში დარწმუნებას - http://cssmenumaker.com/menu/cherry-responsive-menu. ამასთან ფაილების

ერთობლიობა მოცემული .ZIP პორმატით , რაც ავტომატურად გარანტია იმისა, რომ კოდები მოქცეული იქნება ერთ სივრცეში და გაბნეულად მათი შეგროვების საჭიროება არ იქნება. (იხ.სურ.4.13)

**source.zip** _ ფაილის ჩამოწერით დასრულდება მოძიება-ჩამოწერის პროცედურა.

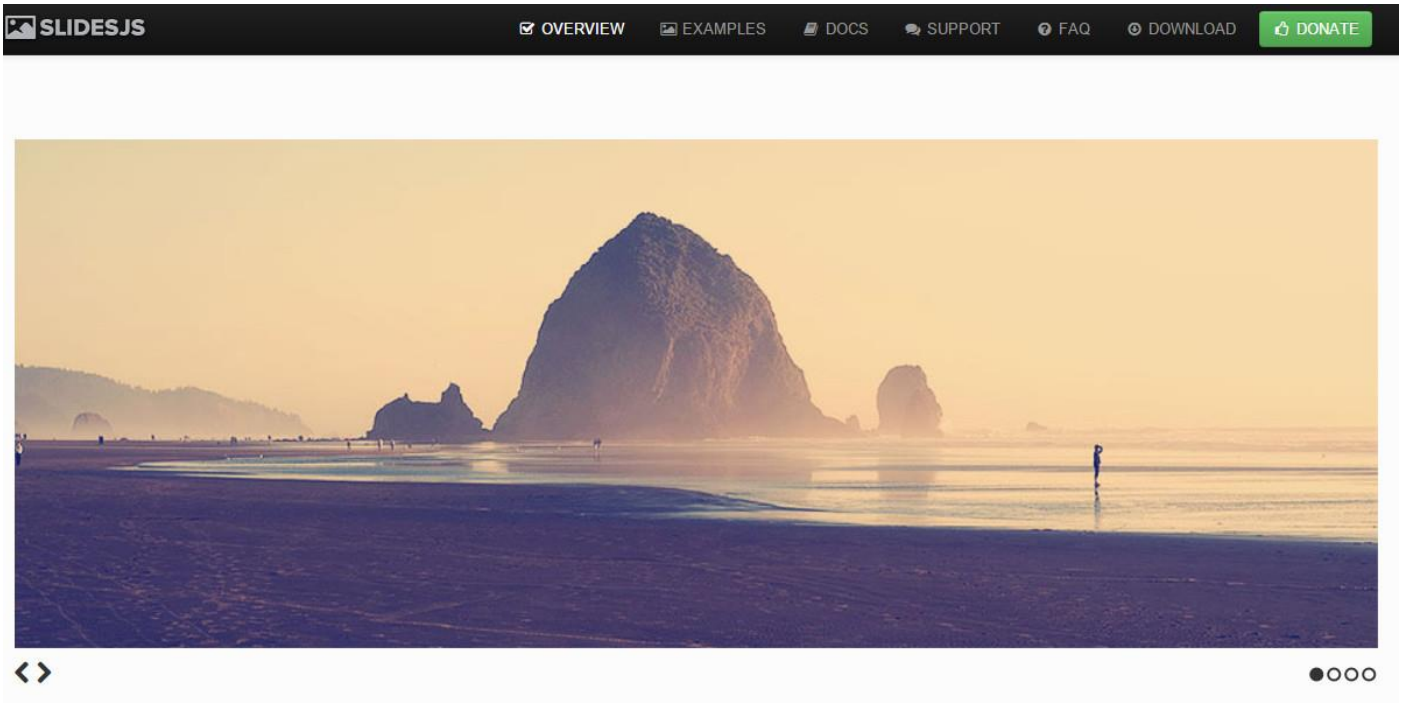
მოცემული მენიუს საიტზე ინტეგრირებას, პარამეტრებისა და შემცველობის ცვლილებას შემდეგ ქვეთავებში გავეცნობით.


4.13 შერჩეული jQeuru drop down menu ჩამოწერა

**დავალება II** _ის წარმატებით შესრულებისათვის საჭიროა იმ გზის განმეორება, რაც დეტალურადაა ახსნილი **დავალება I**_ის შემთხვევაში.

საძიებო სისტემის დახმარებით შესაძლებელია  jQuery slideshow _დანამატის (plugin) მოძიება. მისი ფაილური სტრუქტურის ჩამოწერა.

მოცემულ შემთხვევაში ჩემს მიერ არჩევანი შეჩერდა **SLIDESJS** სლაიდერზე, რომელიც მიმსგავსებულია ჩვენთვის სასურველ ეფექტის მქონე სკრიპტთან. ( ვიზუალი იხილეთ სურათ 4.14 ) მისი მოძიება შესაძლებელია შემდეგ მისამართზე http://www.slidesjs.com/ . ჩამოწერას რაც შეეხება ასევე უმარტივესად ხორციელდება მითითებულ მისამართზე გამოტანილი DOWNLOAD ღილაკის საშუალებით.




4.14 SLIDESJS - მზა სლაიდერი

## 7.7. მოძიებული ბიბლიოთეკის ინტეგრაცია ვებგვერდთან

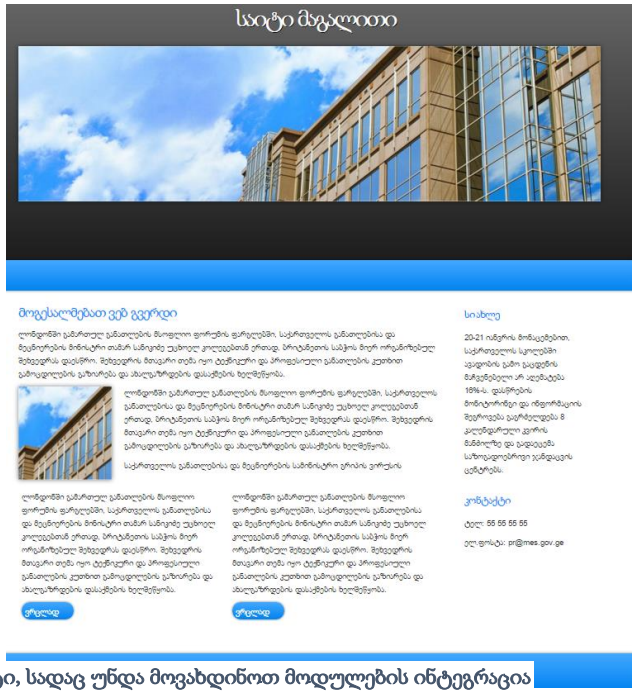
**მიმდინარე პარაგრაფის თემატიკა**

- ბიბლიოთეკის დამაკავშირებელი მისამართები
- ბიბლიოთეკასთან თანდართული ფაილები
- html ელემენტების მზა  ჩონჩხი
- სტილური ფაილი

ბიბლიოთეკის სასურველი მზა მოდულის მოძიებით საკმაოდ დიდი ნაბიჯია გადადგმული, მაგრამ მნიშვნელოვანია მისი სრულფასოვანი ინტეგრაცია მოცემულ ვებ დოკუმენტში.

ძირითადად კოდის განთავსების მეთოდები  ერთგვაროვანია ( ვისაუბრეთ ზედა ქვეთავში ), მაგრამ ფაილური სტრუქტურიდან გამომდინარე მაინც განსხვავებული მიდგომა სჭირდება დამაკავშირებელ მისამართებს მისამართებს.

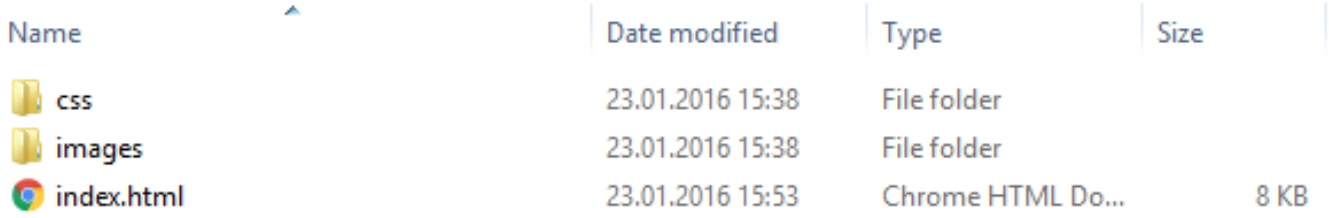
მოცემული **დავალება I** _ ის მაგალითზე მოვახდინოთ კოდის ინტეგრაცია უკვე გამზადებულ ვებ გვერდზე. (იხ.სურ. 4.15 ). მისი საწყისი ფაილური სისტემა მოცემულია სურათ 4.16



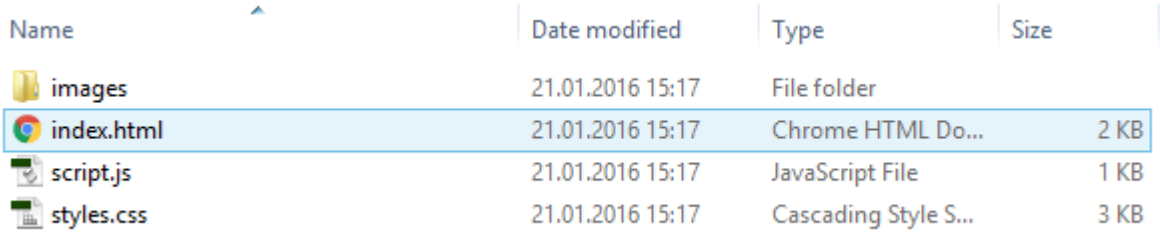

4.15 საიტი, სადაც უნდა მოვახდინოთ მოდულების ინტეგრაცია

საიტზე დატოვებულია ლურჯი არეალი, სადაც იდეაში უნდა გამოისახოს ჩამოსაშლელი მენიუ(სურ 4.9). საწყის ეტაპზე საჭიროა მოხდეს თქვენს მიერ უკვე ჩამოწერილი jQuery Drop Down Menu_ს.ZIP ფაილის ამოარქივება.

პროცესის დასრულების შემდეგ მიიღებთ პროდუქტს შემდეგი ფაილური სტრუქტურით: საქაღალდე **images,** ფაილები **index.html, script.js, styles.css.** (იხ.სურ. 4.17), სადაც images საქაღალდე შეიცავს საჭირო სურათებს (ძირითადად ფონებს მენიუს შემთხვევაში),index.html იმ ტეგთა მემკვიდრეობას და სელექტორებს, რომელიც საჭიროა ჩონჩხისათვის, script.js დამატებით სკრიპტების ერთობლიობა ეფექტებისათვის, styles.css მენიუს სტილებს.




4.16 საიტის საწყისი ფაილური სტრუქტურა




4.17 jQuery Drop Down Menu ფაილური სტრუქტურა

jQuery Drop Down Menu_ს **index.html** კოდის ვიზუალიზაციით შესაძლებელია არსებული სკრიპტის მუშაობისათვის საჭირო მისამართების და უშუალოდ ტეგთა წყობის გაცნობა, ამიტომ მოახდინეთ ფაილის რომელიმე რედაქტორში გახსნა. შედეგად მიიღებთ სურათ 4.18 გამოსახულ მოცემულობას.

```
<!doctype html>
<html lang=''>
<head>
   <meta charset='utf-8'>
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="styles.css">
   <script src="http://code.jquery.com/jquery-latest.min.js" type="text/javascript"></script>
   <script src="script.js"></script>
   <title>CSS MenuMaker</title>
</head>
<body>

<div id='cssmenu'>
<ul>
   <li class='active'><a href='#'><span>Home</span></a></li>
   <li class='has-sub'><a href='#'><span>Products</span></a>
      <ul>
         <li><a href='#'><span>Product 1</span></a></li>
         <li><a href='#'><span>Product 2</span></a></li>
         <li class='last'><a href='#'><span>Product 3</span></a></li>
      </ul>
   </li>
   <li class='has-sub'><a href='#'><span>About</span></a>
      <ul>
         <li><a href='#'><span>Company</span></a></li>
         <li class='last'><a href='#'><span>Contact</span></a></li>
      </ul>
   </li>
   <li class='last'><a href='#'><span>Contact</span></a></li>
</ul>
</div>

</body>
<html>
```

### 4.18 jQuery Drop Down Menu_ს index.html ფაილის შემცველობა

თქვენი ყურადღების გამახვილებას მოვითხოვ სურათ 4.18 ზე მოცემულ რამდენიმე სტრიქონზე. ესენია:

- **სტრიქონი 7** - სტილების შეტანა;
- **სტრიქონი 8** - ბიბლიოთეკის მიბმა;
- **სტრიქონი 9**-თანდართული სკრიპტის ფაილი ჩამოშლის ეფექტის მისაცემად;
- **სტრიქონი 14** - იდენტიფიკატორი id = 'classmenu'მენიუს შემომსაზღვრელ (მშობელ) ბლოკზე.
- **სტრიქონი 16**- იდენტიფიკატორი class = 'active'გამოიყენება გარკვეული ეფექტისათვის. არსებულ მაგალითზე ისარი იცვლის ფერს ( თეთრდება ). მისი უგულველყოფით არაფერი დაშავდება.
- **სტრიქონი 17**-იდენთიფიკატორიclass = 'has-sub'განსაზღვრავს, რომ არსებულ მენიუს ველს გააჩნია ქვეკატეგორიები (ჩამოსაშლელი მენიუ), მისი არ გამოყენება არავითარ შემთხვევაში არ შეიძლება

შევეცადოთ გავითვალისწინოთ გამოყოფილი სტრიქონები და მოვახდინოთ მოდულის საიტზე ჩაშენება. თავდაპირველად მოვაწყოთ საიტის ფაილური სტრუქტურა სასურველ ნებაზე. მაგალითისათვის უკვე არსებულ images მოვახდინოთ მოდულისათვის საჭირო სურათების გადატანა ( რა თქმა უნდა თუ ამის საჭიროებაა). გამომდინარე საქაღალდეების სახელების ერთგვაროვნებისა არანაირი ცვლილება არ იქნება საჭირო მათ აღსაქმელად (თუ წვდომა არ აქვთ css ფაილიდან).

შესაძლოა საიტზე გარდა ჩამოსაშლელი მენიუს სკრიპტისა ჩასაშენებელი იყოს სხვა მოდულები, ამიტომ უკეთ გასარკვევად შესაძლებელია გადაერქვას script.js და styles.css სახელები და მეტად მივამსგავსოთ კონკრეტულ მოდულს, ასევე დავახარისხოთ საქაღალდეების მიხედვით. მაგ.: საიტის ფაილურ

სტრუქტურაში ჩავაშენებ js საქაღალდეს სადაც განთავსდება Javascript_ის ფაილები, ხოლო უკვე არსებულ css ფაილში გადავიტან styles.css. ამასთანავე არ არის აზრს მოკლებული ლოგიკური სახელების მინიჭება styles.css ჩანაცვლდეს ვთქვათ menu_style.css და menu_script.js. შედეგად შეიცვლება საიტის ფაილური სტრუქტურა(იხ.სურ.4.19), რაც მომავალში გამოიწვევს დამაკავშირებელი მისამართების ცვლილებასაც.


4.19 საიტის ახალი ფაილური სტრუქტურა ჩამოსაშლელი მენიუს ფაილების დანამატით

ფაილების ზედმიწევნით ზუსტად განლაგების დასრულების შემდეგ საჭიროა საიტის კოდის სტრუქტურაში ჩარევა და ფაილებთან დამაკავშირებელი მისამართების ჩაშენება. ყურადღება გაამახვილეთ სურათ 4.18 _ის **7, 8, 9** სტრიქონებზე და მოახდინეთ მათი გადაკოპირება საიტის <head> ... </head> გამხსნელ და დამხურავ ბრძანებებს შორის. ასევე სავალდებულოა სამისამართე ტეგების ატრიბუტებში (css_ის შემთხვევაში href='…' , ხოლო Javascript_ის შემთხვევაში src='…' ) დაიტანოთ ცვლიბებლი (იხ. სურ. 4.20-4.22). საიტის კოდის ცვლის ვარიანტი წარმოდგენილია ნაბიჯებად 4.20 – 4.22 სურათებზე.

```
4  <head>
5    <title>ARaynorDesign Template</title>
6    <meta name="description" content="free website template" />
7    <meta name="keywords" content="enter your keywords here" />
8    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
9    <meta http-equiv="X-UA-Compatible" content="IE=9" />
10   <link rel="stylesheet" type="text/css" href="css/style.css" />
11 </head>
```

4.20 საიტის პირვანდელი კოდი

სურათ 4.20 წარმოდგენილია საიტის კოდის პირვანდელი სახე, გამოყოფილია head ტეგი მენიუს ფაილების დამაკავშირებელი მისამართების გასაწერად.

სურათ 4.21 ნაჩვენებია 12 – 14 სტრიქონებზე ახალი კოდის იმპორტირება. ლურჯი ხაზგასმით აღნიშნულია მისამართები, რომლებიც ავტომატურ რეჟიმში გადმოყვა დამაკავშირებელ ბმულებს. მოვახდინოთ მისი სასურველით ჩანაცვლება

```
4  <head>
5    <title>ARaynorDesign Template</title>
6    <meta name="description" content="free website template" />
7    <meta name="keywords" content="enter your keywords here" />
8    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
9    <meta http-equiv="X-UA-Compatible" content="IE=9" />
10   <link rel="stylesheet" type="text/css" href="css/style.css" />
11
12    <link rel="stylesheet" href="styles.css">
13    <script src="http://code.jquery.com/jquery-latest.min.js" type="text/javascript"></script>
14    <script src="script.js"></script>
15
16 </head>
```

4.21 მენიუს ფაილების მისამართების უცვლელი სახით გადმოტანა

ჩვენს მიერ ფაილების სხვადასხვა საქაღალდეებში დახარისხებამ და სახელის ცვლილებამ მოითხოვა არსებული მისამართის კორექტირება. სურათ 4.22_ზე მოცემულია კორექტირებული მისამართები, მორგებული მხოლოდ ჩვენ შემთხვევაზე ( არ გეგონოთ ის უნივერსალურია და ნებისმიერ სკრიპტზე

იმუშავებს ), როგორც ატყობთ **13** სტრიქონმა არ განიცადა ცვლილება, გამომდინარე იქიდან, რომ იქ მითითებული იყო ბიბლიოთეკის გლობალური მისამართი. რადგანაც არ მოვახდინეთ ბიბლიოთეკის ჩამოტვირთვა და მისი ლოკალურად ქცევა არსებული მისამართი დარჩა უცვლელ მდგომარეობაში.

```
12  <link rel="stylesheet" href="css/menu_style.css">
13  <script src="http://code.jquery.com/jquery-latest.min.js" type="text/javascript"></script>
14  <script src="js/menu_script.js"></script>
```

4.22 კორექტირებული ( ჩვენ რეალობას მორგებული ) მისამართები

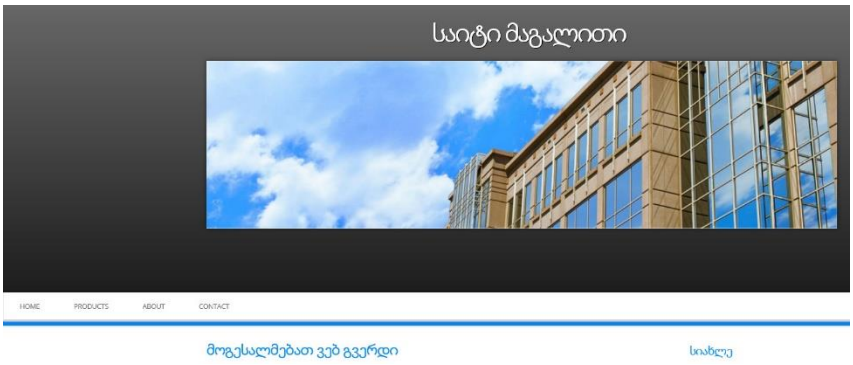
ფაილების დაკავშირებამ წარმატებით ჩაიარა ახლა კონცენტრირება უშუალოდ მენიუს html კოდის ჩაშენებაზე გადაიტანეთ. ამისათვის საჭიროა მენიუს ბლოკის კოპირება სასურველ ადგილას. შედეგად კოდი მიიღებს 4.23 სურათის მოცემულობას.

```
34  <div id="menu_container">
35    <div id='cssmenu'>
36  <ul>
37    <li class='active'><a href='#'><span>Home</span></a></li>
38    <li class='has-sub'><a href='#'><span>Products</span></a>
39      <ul>
40        <li><a href='#'><span>Product 1</span></a></li>
41        <li><a href='#'><span>Product 2</span></a></li>
42        <li class='last'><a href='#'><span>Product 3</span></a></li>
43      </ul>
44    </li>
45    <li class='has-sub'><a href='#'><span>About</span></a>
46      <ul>
47        <li><a href='#'><span>Company</span></a></li>
48        <li class='last'><a href='#'><span>Contact</span></a></li>
49      </ul>
50    </li>
51    <li class='last'><a href='#'><span>Contact</span></a></li>
52  </ul>
53  </div>
54  </div>
```

4.23 მენიუს html ჩონჩხის კოპირება შესაბამის ბლოკში

როგორც სურათიდან ჩანს წინასწარ გამოყოფილ არეალში მოვახდინეთ იმ ტეგთა წყობის ჩაშენება, რომელიც jQuery მოდულის index.html ფაილის შემცველობაშია.

სურათ 4.24_ზე ვიზუალიზებულია საიტზე მიღებული შედეგი. საბოლოო სრულყოფისათვის გაეცანით მომდევნო ქვეთავს, სადაც იქნება ახსნილი კოდთან ურთიერთობა



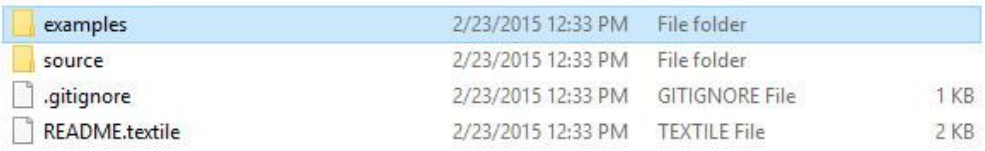

4.24 საიტზე ასახული მენიუს ვიზუალი

**დავალება** I_ის სკრიპტის ინტეგრაციის პროცესზე დაკვირვებით საკმაო

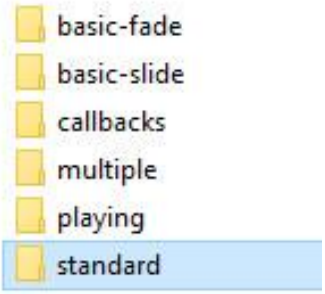
წარმოდგენა შეგექმნებოდათ ზოგადად ბიბლიოთეკის მოდულის მიბმაზე უშუალოდ თქვენთვის სასურველ დოკუმენტთან. ახსნის პროცესში რამდენიმე დეტალზე იყო ყურადღება გამახვილები. ესენია : დამაკავშირებელი მისამართები და ის იდენტიფიკატორები, რომელებიც ეწერება უშუალოდ html ტეგებს, რათა მოხდეს სკრიპტის ორიენტირება უშუალოდ კონკრეტულ ელემენტებზე.

**დავალება II**_ის სკრიპტების ინტეგრირების პროცესის განხილვისას ბევრი რამე უკვე ცნობილი იქნება თქვენთვის და დეტალურად იმ ასპექტებს, რომელიც დავალება I_ის შემთხვევაში დეტალურადაა განხილული, აღარ ჩაუღრმავდებით. ყურადღება გამახვილდება იმ დამატებით სკრიპტის ფრაგმენტებზე რომელთაც აქამდე არ შეხებიხართ.

**SLIDESJS** ფაილის ჩამოწერისა და ამოარქივების შემდეგ გამოისახება სურათ 4.25_ზე მოცემული ფაილური სტრუქტურა. არსებულ შემთხვევაში **examples** საქაღალდე პრიორიტეტულია . მასში პლაგინის მუშაობის საჩვენებელი ვარიანტი ( ვარიანტები ) არის თავმოყრილი. (იხ სურ. 4.26)



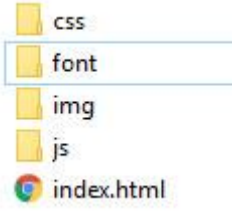

4.25 SLIDESJS ძირითადი საქაღალდე




4.26 examples ფაილის შიგთავსი

ხშირ შემთხევაში სლაიდერის ჩამოწერისას არსებული სლაიდის სხვადასხვა ინტერპრეტაციით გამოყენებაა შესაძლებელი, მაგ.: არსებული სლაიდერი ნავიგაციით, მის გარეშე, დამხმარე ბულეტებით და ა.შ ... მოცემულ შემთხვევაშიც **examples** საქაღალდეში რამდენიმე სტილით გაფორმებული სლაიდია თავმოყრილი. სურვილისებრ შეგიძლიათ მოინახულოთ თითოეული. მაგალითის დონეზე გარჩეული იქნება **playing** საქაღალდეში მოთავსებული სკრიპტი, რომლის ფაილური სტრუქტურა მოცემულია სურათ 4.27_ზე.

სურათ 4.27_დან კარგად ჩანს თქვენთვის უკვე კარგად ნაცნობი ფაილთა კრებული. ამჯერადაც იმ ქმედებათა ერთობლიობას მივმართავთ, რაც დავალება I_ის შემთხვევაში განვახორციელეთ. საჭიროა სასურველი ფაილის გადატანა უკვე არსებულ საიტის სტრუქტურაში.




4.27 standard შიგთავსი

თუ index.php ფაილის შიგთავსში head ტეგს დავაკვირდებით ვნახავთ, რომ არსებულ დოკუმენტში ჩართულია fontAwesome – აიკონების ნაკრები. (იხ.სურ. 4.28 – 24 სტრიქონი ). შესაძლოა კონკრეტული ბულეტების შეტანა მოცემული ბიბლიოთეკის მიხედვით ხორციელდებოდეს. შესაძლებელია მისი

უგულველყოფა და სტანდარტული ფონებით ან სურათებით ჩანაცვლება, მაგრამ უმჯობესია თუ ტექნოლოგიებს აუწყობთ ფეხს და გაარჩევთ ( დამოუკიდებლად ) fontAwesome აიკონებს. შესაბამისად ნება თქვენზეა თუ გსურთ დატოვეთ არსებული მისამართი, მას დასჭირდება იმ ფონტების ერთობლიობა, რომელიც არსებულ შემთხვევაში განთავსებულია font საქაღალდეში და მოგიწევთ მისი გადატანაც მთავარ საიტზე.

```
23  <link rel="stylesheet" href="css/example.css">
24  <link rel="stylesheet" href="css/font-awesome.min.css">
```

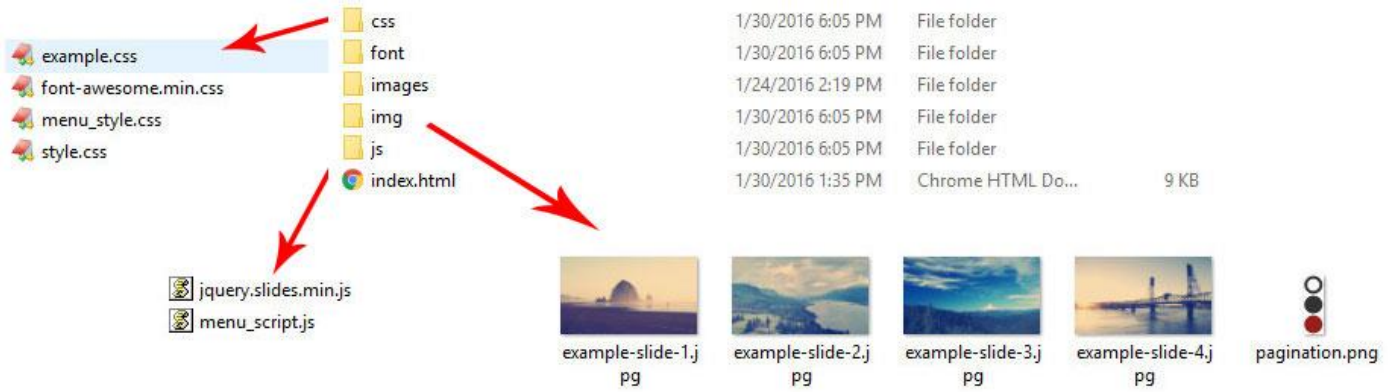
4.28 fontawesome_ის იკონების ფაილის მიბმა

css საქაღალდეში განთავსებული ფაილებიდან font-awesome.min.css სწორედაც, რომ ნახსენები ბიბლიოთეკის ფაილია და მისი უგულველყოფა ფატალური შედეგით არ დასრულდება.

ასევე ფაილი **img** შეიცავს იმ სურათების ერთობლიობას რომელიც უნდა განთავსდეს სლაიდერში. თუ გსურთ მითითებული საქაღალდე უცვლელად გადაიტანეთ , ან თქვენთვის სასურველი შექმენით და იქ განათავსეთ სლაიდის სურათები. დასაშვებია უკვე არსებული images საქაღალდის გამოყენება.

js - საქაღალდეში განთავსებულია სკრიპტი, რომელიც უზრუნველყოფს სლაიდის მუშაობს. მისი გადატანა უმნიშვნელოვანესია.

შედეგად თქვენი საიტის ფაილური სტრუქტურა გაიზრდება და მიიღებს შემდეგ სახეს (იხ.სურ. 4.29)




4.29 საიტის სტრუქტურაში ახალი ფაილების დამატება

ფაილების საჭირო ადგილას გადატანით ერთ ერთი მნიშვნელოვანი ეტაპი გადალახულია. ახლა საჭიროა ზუსტი დამაკავშირებელი მისამართების მითითება. ეს ყოველივე გადაულახავ პრობლემას არ წარმოადგენს, რადგანაც ფაქტობრივად მათი კოპირება გიხდებათ ჩამოწერილი ფაილის index.html_დან თქვენს დოკუმენტში ( მოცემულ შემთხვევაში ასევე index.html _ში ).

მზა მოდულის index.html ფაილი შემდეგნაირად გამოიყურება ( იხ.სურ 4.30 ) (კოდი სიდიდის გამო გაწყვეტილია, წარმოდგენილია მხოლოდ მნიშვნელოვანი ნაწილები)

```
<!doctype html>
<html>
<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width">
 <link rel="stylesheet" href="css/example.css">
 <link rel="stylesheet" href="css/font-awesome.min.css">
 <style>
  body {
   -webkit-font-smoothing: antialiased;
   font: normal 15px/1.5 "Helvetica Neue", Helvetica, Arial, sans-serif;
   color: #232525;
   padding-top:70px;
  }
```

……..

```
<body>
 <div class="container">
  <div id="slides">
   <img src="img/example-slide-1.jpg" alt="Photo by: Missy S Link: http://www.flickr.com/photos/listenmissy/5087404401/">
   <img src="img/example-slide-2.jpg" alt="Photo by: Daniel Parks Link: http://www.flickr.com/photos/parksdh/5227623068/">
   <img src="img/example-slide-3.jpg" alt="Photo by: Mike Ranweiler Link: http://www.flickr.com/photos/27874907@N04/4833059991/">
   <img src="img/example-slide-4.jpg" alt="Photo by: Stuart SeegerLink: http://www.flickr.com/photos/stuseeger/97577796/">
  </div>
 </div>
 <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
 <script src="js/jquery.slides.min.js"></script>
 <script>
  $(function() {
   $('#slides').slidesjs({
    width: 940,
    height: 528,
    play: {
     active: true,
     auto: true,
     interval: 4000,
     swap: true
    }
   });
  });
 </script>
</body>
</html>
```

4.30 მზა მოდული index.html ფაილის შემცველობა

სურათ 4.30 მოცემულობიდან მნიშვნელოვანი სტრიქონები გამოყოფილია მონიშვნის ხაზით. მაგ.:

- **სტრიქონი 6** - უზრუნველყოფს სტილური ფაილის ჩართვას;
- **სტრიქონი 7** – font-awesome;
- **სტრიქონი 8** - მიუხედავად იმისა რომ სტილები გატანილია გარე ფაილში, როგორც ატყობთ გამოყენებულია შიდა სტილებიც. გამომდინარე იქიდან, რომ slideshow მოდულის ჩამოწერისას

ფაილი უზრუნველყოფილი იყო რამდენიმე სტილის სლაიდერით. გარე სტილური ფაილი ყველასთვის ერთი გახლდათ, შესაბამისად მათი გარჩევადობისათვის ინდივიდუალურად გაწერილია სტილი შიდა სტილებად. შესაბამისად შესაძლებელია მოცემული სტილი ამოიჭრას მითითებული სტრიქონიდან და გადაიტანოთ თქვენთვის სასურველ გარე სტილურ ფაილში, თუნდაც example.css _ში (სტრიქონი 6);

განსხვავებით **დავალება I**_საგან მოცემულ შემთხვევაში სკრიპტები ჩართულია არა დოკუმენტის head ბლოკში არამედ body ტეგში. ეს მეთოდიც გამოყენებადია, როცა სკრიპტი არ საჭიროებს კონკრეტულ გვერდზე გამოყენებას.

ყურადღება გაამახვილეთ შემთხვევაზე როდესაც ერთდროულად საიტზე რამდენიმე სკრიპტია განსათავსებელი. ყოველი მოდულის ჩამოწერისას მას საკუთარი ბიბლიოთეკის მისმართი მოყვება (**სტრიქონი 131**).რამდენიმე ბმულის ერთდროული გამოყენება არ შეიძლება თუ საქმე ერთ კონკრეტულ ბიბლიოთეკას ეხება. მაგ. საიტზე გამოყენებულია jQuery Drop Down Menu, ასევე ვიყენებთ jQuery slideshow, შესაბამისად ბიბლიოთეკა გამოიყენება jQuery. მასთან დამაკავშირებელი მისამართი 1_ზე მეტი არ არის საჭირო. უმჯობესია დატოვოთ ბიბლიოთეკის უახლესი ვერსია, ამიტომაა რეკომენდირებული ბიბლიოთეკის ოფიციალური საიტიდან ჩართვა სადაც განიცდის ავტომატურ განახლებებს.

jQuery Drop Down Menu მოდულის ჩართისას უკვე მოვახდინეთ ინტეგრირება არსებული მისამართის და 131 სტრიქონზე მოცემული კოდის შეტანა აღარაა საჭირო.

**132 სტრიქონზე** მითითებულია ლოკალური სკრიპტის მისმართი, მისი კოპირება ჩვენ დოკუმენტში აუცილებელია, ისვე როგროც **133-146 სტრიქონებს** შორის მოქცეული სკრიპტისა. ის უზრუნველყოფს სლაიდ შოუს სკრიპტის პარამეტრების მართვას - კონფიგურირებას.

კოდის სრულფასოვანი მუშაობისათვის დარჩა მხოლოდ იმ html ტეგების კოპირება რომელიც საიტზე სურათების გამოტანას უზრუნველყოფს **სტრიქონები 123 და124 . დავალება I**__ის მსგავსად ყურადღება გაამახვილეთ მოცემულ სელექტორებზე class = 'container' და id = 'slides'.

```
<div class = 'container'>
    <div id = 'slides'>


    </div>
</div>
```

მითითებულ სტრიქონებზე ყურადღების გამახვილება საშუალებას მოგცემთ უშეცდომოთ ასახოთ სლაიდერი თქვენთვის სასურველ ადგილას და მოამზადოთ მომავალი ცვლილებებისათვის.

## 7.8. მზა კოდში ცვლილებების შეტანა

**მიმდინარე პარაგრაფის თემატიკა**

- ტექსტის ცვლილება
- სურათის პარამეტრები
- ბმულის მისამართები
- ელემენტების ზომის პარამეტრები
- ეფექტების სიჩქარე
- ეფექტის ტრაექტორია

მოცემულ ქვეთავში აღწერილი იქნება დავალების პარამეტრების ცვლილება.

რა თქმა უნდა მოძიებული ბიბლიოთეკის plugin საკმაოდ დიდ როლს თამაშობს დეველოპერისათვის საქმის გაადვილების თვალსაზრისით, მაგრამ არ არსებობს იდეალური და უნივერსალური plugin, რომელიც ერთი - ერთში მოერგება სასურველ საიტს. ყველა დანამატი საჭიროებს რაღაცა დოზით ჩარევას სრულყოფისათვის( საუბარია საიტის პარამეტრებთან მიმართებაში ).

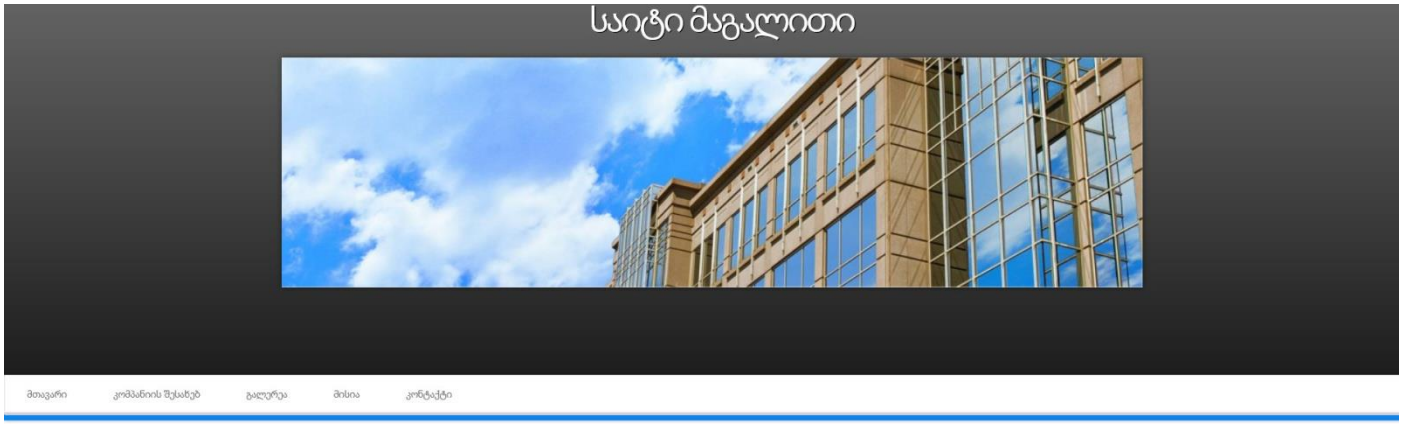
მიუხედავად იმასა, რომ jQuery Drop Down Menu იდეალურად მოერგო საიტის პროპორციებს, სამუშაო ჯერ არ დასრულებულა.

უპირველეს ყოვლისა ჩასანაცვლებელია მენიუს შემცველობა იმ კონკრეტული პუნქტებით, რომელსაც მოითხოვს დამკვეთი (მომხმარებელი).

```
<div id="menu_container">
 <div id='cssmenu'>
  <ul>
   <li class='active'><a href='#'><span>მთავარი</span></a></li>
   <li class='has-sub'><a href='#'><span>კომპანიის შესახებ</span></a>
    <ul>
     <li><a href='#'><span>ისტორია </span></a></li>
     <li><a href='#'><span>დამფუძნებლები</span></a></li>
     <li class='last'><a href='#'><span>სამეურვეო საბჭო</span></a></li>
    </ul>
   </li>
   <li class='has-sub'><a href='#'><span>გალერეა</span></a>
    <ul>
     <li><a href='#'><span>ფოტო</span></a></li>
     <li class='last'><a href='#'><span>ვიდეო</span></a></li>
    </ul>
   </li>
   <li class='last'><a href='#'><span>მისია</span></a></li>
   <li class='last'><a href='#'><span>კონტაქტი</span></a></li>
  </ul>
</div>
</div>
```

როგორც სურათ 4.18 საუბრისას გავამახვილეთ არსებობს აუცილებლობა მთავარ და ქვე მენიუებს შესატყვისი სელექტორები ჰქონდეთ გაწერილი (id = 'classmenu', class = 'has-sub' ) . შესაბამისად თქვენს მიერ მოწყობილ ჩადგმულ მენიუზე დაურთეთ თან არსებული სელექტორები, მთავარი ბლოკი -> id = 'classmenu', იმ კონკრეტულ li ტეგებს რომელებისაც გააჩნიათ ქვე მენიუ class = 'has-sub'. (იხ. სურ . 4.31)

მიღებული შედეგი სასურველთან მიახლოებულია. ეკრანზე გამოისახება მენიუ ჩამოშლა აკეცვის ფუნქციით. საჭიროა მხოლოდ სტილების რედაქტირება და იდეალური შედეგიც არ დააყოვნებს (იხ.სურ. 4.32)




4.32 საიტზე მიღებული შედეგი

მენიუს სრულყოფილებამდე მცირეოდენი გვაშორებს. jQuery Drop Down Menu _ს შემთხვევაში ძირითადი შემხებლობა მხოლოდ სტილურ ფაილთან მოგვიწევს გამომდინარე იქიდან, რომ სკრიპტთან მიმართებაში არც არანაირ პარამეტრის ცვალებადობას აზრი არა აქვს- მენიუ იშლება ზევიდან ქვევით და არსებულს სკრიპტს სხვა თვისების მიღება არ შეუძლია. ( მიმდინარე მოდული არ ითვალისწინებს სკრიპტის მთლიანი კოდის ცვლილებას, პარამეტრები კი არ გააჩნია მენიუთა უმეტესობას ).

ჩვენს მიერ შერჩეული მოდული jQuery Drop Down Menu , არც ისე დიდი რაოდენობის სტილების შეიცავს სულ რაღაც 131 სტრიქონია ( ☺ ).შესაძლებელია ეს რაოდენობა ბევრს გრანდიოზულად ეჩვენოს, მაგრამ არც ისე რთულადაა საქმე გამომდინარე იქიდან, რომ არსებული კოდი ორიენტირებული ფაქტობრივად 2 ძირითად სელექტორზე ესენია id = 'classmenu', class = 'has-sub'. აქ კი ნამდვილად გაგიმართლათ, რადგანაც ყველაფრის შეცვლა უმარტივეს პროცედურებთანაა დაკავშირებული. იმ შემთხვევაში თუ მაინც წააწყდით გარკვეულ სირთულეებს გამოიყენეთ სტილების ცვლილებისათვის ყველაზე მოხერხებული ბრაუზერების დამატებითი ხელსაწყო Inspect Element ( მასთან შეხება მოდულის წინაპირობის გავლისას მოგიწევდათ ), რათა დაადგინოთ კონკრეტული ელემენტის მიმმართველი და იმ სტილის მდებარეობა, რომელიც მასზეა მისადაგებული.

ჩამოვთვალოთ ის განსხვავებები რაც თვალში საცემია და რისი აღმოფხვრაც მოგვიწევს სტილის ფაილში:

1. მენიუ განთავსებულია საიტის მარცხენა კიდეში;
2. ასევე მისი ფონი არ არის შესაბამისი;
3. შესაცვლელია მენიუს ველების ფონტის ზომა და ფერი;
4. ასევე მაუსის კურსორის მიტანისას სხვა ფერის ფონი გამოისახება;

გავყვეთ ეტაპობრივად თითოეულ პუნქტს.

მენიუს მარცხენა კიდიდან ცენტრისაკენ წასაძრავად საჭიროა მენიუს ძირითად ბლოკს გაეწეროს სიგანე და გაცენტრების ბრძანება. უმჯობესია მიგნებულ იქნეს კონკრეტული სტრიქონი და ჩანაცვლდეს ზომები, დასაშვებია ასევე უკანასკნელ ხაზზე გაიწეროს ბრძანებები (რაც გამოიწვევს ჩანაცვლებას), მაგრამ ეს არც ისე კარგ ტონად ითვლება, რადგანაც საქმე გვაქვს კოდის დუბლირებასთან. (იხ.სურ.4.33)

```
2  #cssmenu {
3    font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, Sans-Serif;
4    font-size: 10px;
5    line-height: 15px;
6    text-transform: uppercase;
7    text-align: left;
8    width: 930px;
9    margin: auto;
10 }
```

4.13 მენიუს გაცენტრება

როგორც სურათ 4.33_დან ჩანს ორი ბრძანების ჩამატება **8-9 სტრიქონებზე** გამოიწვია მენიუს გაცენტრება.

```
12 #cssmenu > ul {
13   width: auto;
14   list-style-type: none;
15   padding: 0;
16   margin: 0;
17   font-size: 16px;
18   /* background: #ffffff;
19   border: 1px solid #ece6e8;
20   border-bottom: 3px solid #d9ced2;
21   -webkit-border-radius: 2px;
22   -moz-border-radius: 2px;
23   -o-border-radius: 2px;
24   border-radius: 2px;*/
25 }
```

4.34 მენიუს ფონის და წარწერის ზომის ცვლილება

ახლა გადავინაცვლოთ მენიუს ფერის და წარწერის ზომის ცვლაზე.

როგორც სურათ 4.34_ზე არის მოცემული font-size ბრძანებით მითითებულ მენიუს ტეგზე შესაძლებელია მისი შემცველობის ზომის ცვლილება (სტრიქონი 17), ასევე დამატებითი ბრძანებები, რომლებიც უზრუნველყოფდნენ ფონის და ჩარჩოს მინიჭებას მიმდინარე საიტის სტილში არ არის გათვალისწინებული და შეგიძლიათ წაშალოთ (სურათზე დაკომენტარებულია )

გამომდინარე იქიდან, რომ მენიუს ველები დალინკულია ფერი კი <a> ტეგზეა მინიჭებული მის შესაცვლელად უნდა ვიმოქმედოთ უშუალოდ ბმული ბრძანებაზე. ჩრდილის მოსახსნელად და ფერის მხოლოდ შვილობილ ლინკებზე მოსახსნელად ვიყენებთ უკვე ნაცად css კოდს, რომელიც აუცილებლად განთავსებული უნდა ქვედა სტრიქონზე ,რათა გადაეწეროს ანალოგიურ ბრძანებას.(იხ.სურ.4.35) მიიღება ფერშეცვლილი მენიუ

```
70  #cssmenu > ul li a {
71    display: block;
72    padding: 12px 24px 11px 24px;
73    text-decoration: none;
74    color: #747474;
75    text-shadow: 0px 1px 0px #fff;
76  }
77  #cssmenu > ul > li > a {
78    color: #fff;
79    text-shadow: 0px;
80  }
```

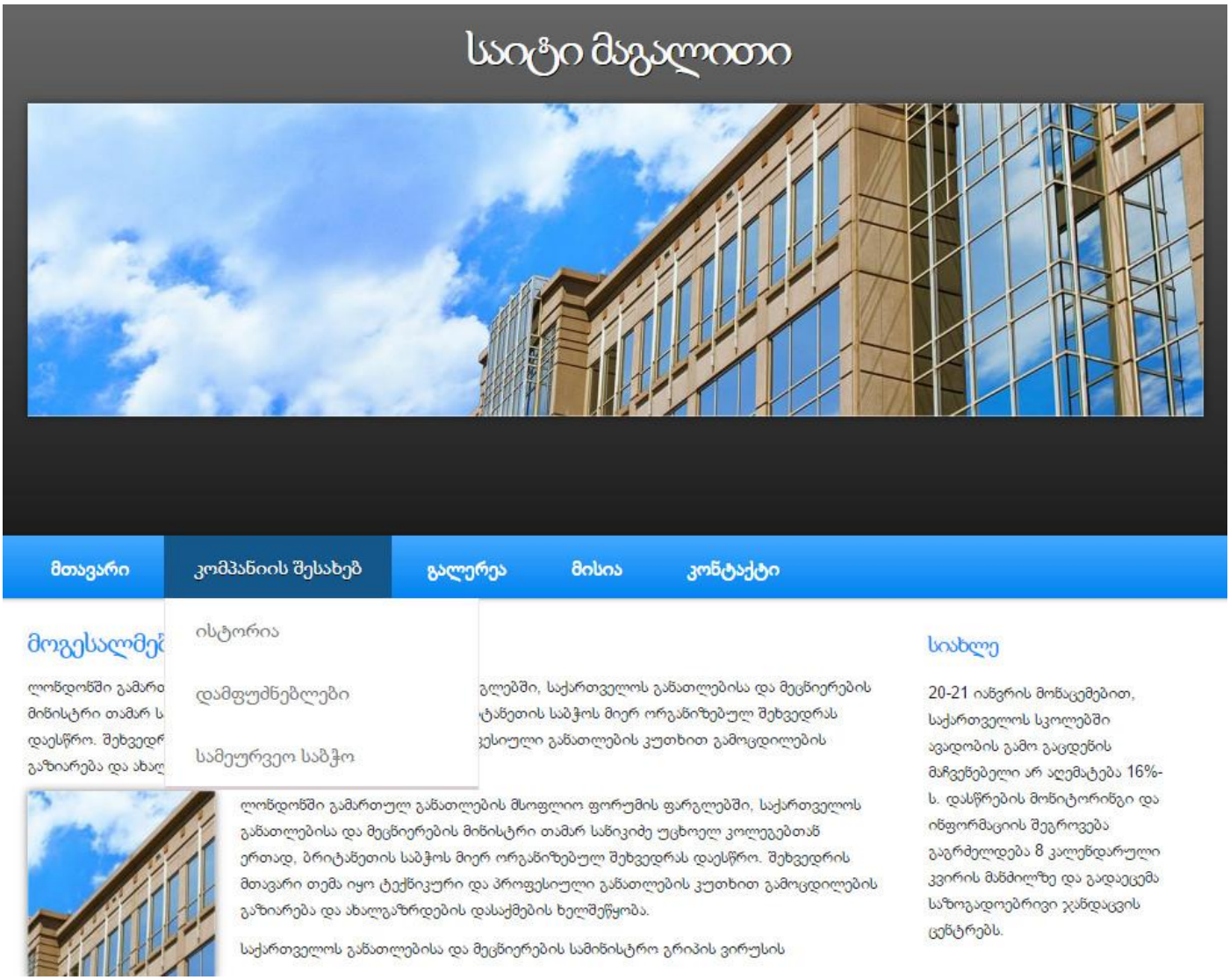
4.35 წარწერის ფერის ცვლილება

```
65  #cssmenu > ul li.has-sub > a.active,
66  #cssmenu > ul li.has-sub > a:hover {
67    /*background: #d80041 url('images/caret.png') no-repeat;*/
68    background: #13578b;
69    background-position: 90% 195%;
70  }
```

4.36ფონის ცვლილება მაუსის კურსორის მიტანისას

მაუსის კურსორის მიტანისას ფონის ცვლილებისათვის განახორციელეთ სურათ 4.36_ზე მოცემული კოდის კოპირება თქვენს დოკუმენტში.

საბოლოო შედეგი ნაჩვენებია სურათ 4.37_ზე




4.37 ჩამოსაშლელი მენიუს დასრულებული იერსახე

**დავალება II**_ის ვიზუალის ცვლილებისათვის საჭიროა სტილურ ფაილში (example.css) ჩარევა. მოცემულ შემთხვევაში სტილურ ფაილზე ყურადღების გამახვილება არამიზანშეწონილი იქნება, გამომდინარე იქიდან, რომ დავალება I_ის გარჩევისას დეტალურად შევეხეთ სტილებზე ზემოქმედებას.

უმჯობესი იქნება თქვენი რესურსის მობილიზება მოვახდინოთ განსხვავებული ტიპის სკრიპტზე, რომელიც პირველ შემთხვევაში არ ფიგურირებდა.

```
123  <div class="container">
124   <div id="slides">
125     <img src="img/example-slide-1.jpg">
126     <img src="img/example-slide-2.jpg">
127     <img src="img/example-slide-3.jpg">
128     <img src="img/example-slide-4.jpg">
129   </div>
130  </div>
```

4.38 სურათების ჩანაცვლება

საწყის ეტაპზე html დოკუმენტში გადავხედოთ იმ ბლოკს, სადაც ხდება უშუალოდ სურათების შემოტანის პროცესი, რათა მოვახდინოთ img ტეგში მისამართების ცვლილება თქვენთვის საჭირო გამოსახულების ასასახად. (იხ.სურ.4.38) როგორც სურათიდან ჩანს მთავარი ყურადღება სელექტორებს class = 'container' და id = 'slides' ექცევა. სურათები სტანდარტულად არის განთავსებული.

სასურველი სურათების განთავსების შემდეგ დროა გადახვიდეთ უშუალოდ სლაიდერის პარამეტრების ცვლილებაზე.

განსხვავებით წინა დავალების დროს გამოყენებული სკრიპტისაგან, მოცემულ მოდულს მოყვება დამატებითი სკრიპტი (იხ.სურ.4.39) უშუალოდ დოკუმენტის ტანში ჩადგმული , რომლის მეშვეობათაც შესაძლებელია სლაიდერის დამატებითი პარამეტრების კონფიგურირება.

```
133  <script>
134   $(function() {
135    $('#slides').slidesjs({
136      width: 940,
137      height: 528,
138      play: {
139        active: true,
140        auto: true,
141        interval: 1000,
142        swap: true
143      }
144    });
145   });
146  </script>
```

4.39 პარამეტრების კონფუგურაციის სკრიპტი

თუ სკრიპტის ყველანაირი დეტალი გაკმაყოფილებთ და საიტზე მოსარგებად მხოლოდ ზომების ცვლილებას საჭიროებს სტილებში ჩარევისაგან თავის არიდებით შესაძლებელია მოთხოვნის დაკმაყოფილება. **სტრიქონ 135-136** გამოტანილია ზომების პარამეტრები, რომელთა ცვლილება გამოწვევს სლაიდერის ზომების
ცვლილებას. შესაბამისად **width**: - სიგანე, **height**:- სიმაღლე.

გარდა ზომის ცვლილებისა შესაძლებელია უშუალოდ სკრიპტის სიჩქარის ცვლილება. (სტრიქონი 141). **interval** – 1000 უზრუნველყოფს ანიმაციის მოქმედებაში მოყვანას ყოველ ერთ წამში. მისი ცვლილება ანიმაციის დაჩქარება / შენელებას უზრუნველყოფს ( 4000 – 4 წამი, 1 წამი უდრის 1000 მილი წამს ).

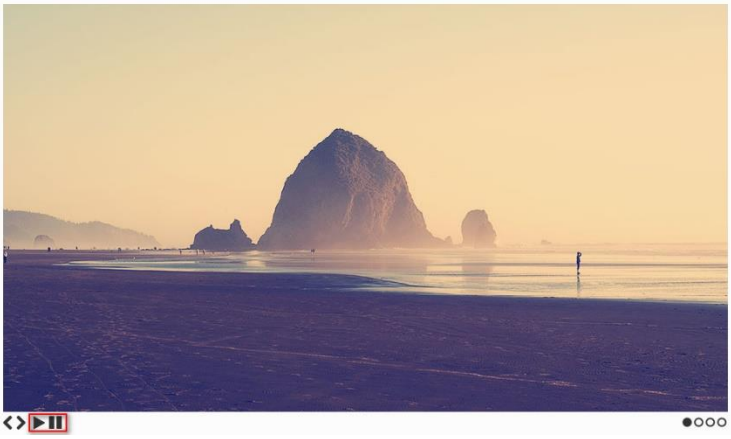
პროგრამირებაში საკვანძო სიტყვა **true** ჭეშმარიტების, დადებითი სიდიდის აღმნიშვნელია შესაბამისად სადაც იგია გაწერილი თანხმობას დასტურს აღნიშნავს. მაგ.: **auto: true** სლაიდის ანიმაციის ავტომატურ დაწყებას უზრუნველყოფს. ის ჩატვირთვის თანავე იწებს მოქმედებას. თუ დეველოპერს სურს სლაიდი მხოლოდ ღილაკზე დაჭერისას გაეშვას მაშინ მოცემული თვისებას უნდა მიანიჭოს მნიშვნელობა **false** --- **auto: false;**

4.40 play / pause ღილაკის ვიზუალიზაცია

**active**: **true** ---- პასუხისმგებელია სამართავი ღილაკების pause / play - გამოჩენაზე (იხ.სურ4.40)

**active**: **false** --- მოცემულ ღილაკი უხილავი გახდება მომხმარებლისთვის / აღარ აისახება ვებ გვერდზე.

**swap**: **true** - უზრუნველყოფს ღილაკის ერთსახოვნად გამოჩენას. ერთ ღილაკშია pause / play გაერთიანებული და ჩანს მონაცვლეობით, მისთვის **false** მნიშვნელობის მინიჭებით გამოისახება სურათ 4.41_ის მოცემულობა, გამოჩნდება ორივე ღილაკი ერთდროულად



4.41 play/pause ღილაკის ერთდროული ვიზუალიზაცია

გაითვალისწინეთ მოცემული სკრიპტის ობიექტის თვისებები: **width, height, interval, active, swap** პირობითია და შესაძლებელია სხვადასხვა დეველოპერის ან ორგანიზაციის მიერ სხვადასხვანაირად იყოს დეკლარირებული ( მაგ. ანიმაციის სიჩქარე **interval**_ის მაგივრად **speed** და ა.შ ) და მოცემული ახსნა არ იყოს თავსებადი სხვადასხვა შემთხვევებთან, მაგრამ აზრობრივად და შინაარსობრივად სკრიპტების უმეტესობა ერთ კანონზომიერებას ემორჩილებიან.

*კითხვები თვითშეფასებისათვის*

1. რას წარმოადგენს ბიბლიოთეკები და განმარტეთ მათი როლი თანამედროვეობაში;
2. დაასახელეთ ყველაზე მოთხოვნადი javascript ბიბლიოთეკა;
3. რა ფაილთა ერთობლიობისაგან შედგება jquery კონკრეტული ეფექტების მოდულები;
4. რა წესების დაცვაა საჭირო ბიბლიოტეკის მოდულის არასებულ საიტზე ჩასაშენებლად;
5. რა მნიშვნელობა აქვს html ტეგების ჩონჩხისა და იდენთიფიკატორები ზედმიწევნით სწორად დაცვას?

# 8. საიტის ინტერაქტივისა და ეფექტების შემუშავება - JavaScript

Javascript მსოფლიოს ყურადღების ცენტრში 1995 წლიდან მოექცა. მისი მეშვეობით შესაძლებელი იყო ვებ გვერდზე პროგრამული ნაწილის განთავსება. საწყის ეტაპზე იგი აღიქმებოდა მხოლოდ ბრაუზერ Netscape Navigator_ში.

Javascript ენაზე მოთხოვნის სწრაფმა ზრდამ გამოიწვია უმოკლეს დროში მისი ინტეგრაცია წამყვან ბრაუზერებში. ამ პროგრამირების ენამ საშუალება მისცა დეველოპერებს შეექმნათ ვებ საიტებისთვის სხვადასხვა პროგრამული დანამატები (მაგ. რუკები), მაგრამ ძირითადად იგი გამოიყენებოდა ვებ გვერდზე ინტერაქციისა და სხვადასხვა ეფექტების შესამუშავებლად.

გასათვალისწინებელია, რომ Javascript პროგრამირების ენას არავითარი საერთო ( წარმოების თვალსაზრისით ) არ გააჩნია JAVA პროგრამირებასთან. ხშირ შემთხვევაში იგი აღიქმება ხან Java პროგრამირების შვილობილ ენად, ხანაც უშუალოდ java_დ. დაუშვებელია Javascript_ის java_დ მოხსენიება. ეს ორი ერთმანეთისაგან განსხვავებული პროგრამირების ენებია. უნდა აღინიშნოს, რომ მათ შეუძლიათ თანადროული მუშაობა.

მას შემდეგ, რაც არსებული პროგრამირების ენის მოხმარება გასცდა Netscape Navigator და გახდა უფრო ფართოდ გამოსაყენებელი ორგანიზაცია ECMA_ს მიერ შემუშავებულ იქნა სტანდარტი ECMAScript - დოკუმენტი, რომელშიც აღწერილი იყო ენის მუშაობის პრინციპები.

საწყის პერიოდში, როდესაც ECMAScript სტანდარტი იწყებდა განვითარებას მსოფლიოში დომინირებდა პროგრამირების ენის Java. Java იმდენად პოპულარული გახლდათ - მიიჩნეოდა, რომ მომავალში ყველა პროგრამირების ენა ჩაიყლაპებოდა ამ გიგანტის მიერ. სწორედ არსებული სიტუაციიდან გამომდინარე

ორგანიზაცია ECMA_მ განახორციელა გამართლებული მარკეტინგული სვლა და პროდუქტს უწოდა Javascript_ი ( თავდაპირველი სახელწოდება არსებული **LiveScript** ). ამ პერიოდიდან დოკუმენტაცია ECMAScript მოიხსენება Javascript_ად. არსებულ ენას ECMAScript_ის სტანდარტი გააჩნია დღესაც უბრალოდ ფართო მასებისთვის ის რჩება Javascript ენად.

პროგრამების შესასრულებლად არა აქვს მნიშვნელობა, თუ რომელ პროგრამირების ენაზე ისინი დაწერილი, გამოიყენება 2 მეთოდი : **„კომპილაცია“** და **„ინტერპრეტაცია“**.

- **კომპილაცია** - ეს არის შემთხვევა როცა პროგრამის კოდს იღებას სპეციალური პროგრამა **„კომპილატორი“** , გარდაქმნის სხვა ენად, როგორც წესი მანქანურ ენად (ნულებისა და ერთების ენად 0-1) . ეს კომპილირებული კოდი დამოუკიდებლად Source (საწყისი) კოდისგან ეშვება პროგრამის სახით.

- **ინტერპრეტაცია** - ეს არის შემთხვევა, როცა პროგრამის კოდს იღებას ინსტრუმენტი, რომელიც მოიხსენება **„ინტერპრეტატორად“** და ასრულებს ისე როგორი სახითად მიუვიდა სკრიპტი. Javascript სწორედ ასეთი ტიპის პროგრამირების ენაა. ეს მიდგომა გამოიყენება ბრაუზერებში javascript_ის კოდი მიღებისას.

თანამედროვე ინტერპრეტატორები გარდაქმნისა და უშუალოდ გაშვების პროცესშიც ახდენენ javascript კოდის ოპტიმიზაციას, ამიტომ Javascript მუშაობს ძალიან სწრაფად.

დღეს ფაქტობრივად ყველა ბრაუზერში ჩაშენებულია Javascript ინტერპრეტატორი, ამიტომ მისი შესაძლებელია მისი აქტიური გამოყენება ვებ გვერდებზე. **რა თქმა უნდა შესაძლებელია Javascript გამოყენება ბრაუზერის გარეშეც.**

## რა შეუძლია Javascript_ს ?

Javascript_ის შესაძლებლობები დამოკიდებულია გარემოზე რომელშიც უწევს ფუნქციონირება. მაგ.: ბრაუზერში მას შეუძლია ყველაფერი რაც შეეხება ვებ გვერდის მანიპულაციას, მომხმარებელს, ურთიერთობას სერვერთან გარკვეული დოზით:

- ახალი HTML ტეგების შექმნა, არსებულის წაშლა, ტეგებისათვის სტილების გაწერა, მათი დამალვა/გამოჩენა;
- მომხმარებლების მოქმედებებზე რეაგირება, მაუსისა და კლავიატურის ხდომილებებზე (event) კონკრეტული ფუნქციონალის მინიჭება;
- სერვერზე მოთხოვნების გაგზავნა, ინფორმაციის ჩატვირთვა ვებ გვერდის განახლების გარეშე (ajax ტექნოლოგია);
- Cookie_თან ურთიერთობა;

ეს მხოლოდ ნაწილია იმ შესაძლებლობისა, რომელსაც ფლობს პროგრამირების ენა Javascript.

მიმდინარე მოდულის ფარგლებში ჩვენ შევეხებით javascript პროგრამირებას, რომელიც ორიენტირებულია მომხმარებლის მხარეზე, შესაბამისად მისი მოქმედების გარემო იქნება ბრაუზერი.

## 8.1. მარტივი ამოცანის გადაწყვეტა JavaScript ენის ძირითადი ელემენტების გამოყენებით

**მიმდინარე პარაგრაფის თემატიკა**

- დეკლარაციების გამოცხადება
- არითმეტიკული, ლოგიკური, ბინარული, სტრიქონული და პირობითი ოპერატორები
- შედეგის გამოტანის ოპერატორები

როგორც შესავალში აღვნიშნეთ javascript ფართო შესაძლებლობების მქონე პროგრამირების ენაა. ჩვენს შემთხვევაში ორიენტირებულნი ვიქნებით javascript_ის გამოყენებაზე ვებ გვერდში, შესაბამისად სამუშაო გარემო html დოკუმენტთან უშუალო კავშირში იქნება.

javascript კოდის არეალის გამოყოფა html დოკუმენტის სხვადასხვა ადგილასაა შესაძლებელი. გამომდინარე დავალების ინტერესებიდან ის შეიძლება გამოძახებულ იქნას:

- <head> JS </head> ტეგში;
- <body> JS </body> ტეგში;
- შესაძლებელია პროგრამული კოდის სხვა დოკუმენტში ფორმირება და შემდგომში მისი იმპორტორება **html** დოკუმენტში;

თითოეულ ამ ხერხს დაწვრილებით გზადაგზა გავარჩევთ. განვიხილავთ თუ რა დატვირთვის მატარებელია ზემოთ ჩამოთვლილი გზებით სკრიპტის ასახვა. მაგრამ საწყის ეტაპზე შევეცადოთ მოვამზადოთ Javascript არეალი <body> ტეგში. ამისათვის საჭიროა გავხსნათ და დავხუროთ <script> … </script> ტეგი. არსებულ ტეგში განთავსებული ინფორმაცია წარმოადგენს პროგრამულ კოდს.

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
</head>
```

```
<body>

 <script>
 .....................................
  </script>

</body>
</html>
```

მიმდინარე ეტაპზე დასაშვებია <script> ტეგის უატრიბუტოდ გამოცხადება.

შესაძლებელია <script type="text/javascript"> გამოყნებაც, რომელიც  არც თუ ისე შორეულ წარსულში აქტიურად გამოიყენებოდა. დეველოპერთა დიდი ნაწილი არსებულ ჩანაწერს დღესაც ინტენსიურად იყენებს.

შესაძლოა რომელიმე საიტთან შეხებისას <script language="JAVASCRIPT"> გახსნის შემდეგნაირი ჩანაწერი აღმოაჩინოთ. ის საკმაოდ ძველმოდურია და ფაქტობრივად აღარსად გამოიყენება. თუ მეორე ჩანაწერზე აღვნიშნეთ რომ ჯერ კიდევ აქტიურად მოიხმარება, ამ ტიპის ჩანაწერი არც თუ ისე დადებითად დაგახასიათებთ - ის არავალიდურია.

შესაძლოა Javascript_ით გაფორმებულ დოკუმენტში თვალი მოკრათ მსგავსი ტიპის ჩანაწერს

```
<script type="text/javascript"><!--
   ...
//--></script>
```

სადაც <script> გამხსნელ - დამხურავ ტეგებს შორის მოთავსებულია html კომენტარები. ეს მეთოდი გამოყენებოდა იმისათვის, რომ  თუ ბრაუზერს არ ექნებოდა Javascript_ის მხარდაჭერა მასში არსებული კოდი არ გამოისახებოდა ეკრანზე ის ავტომატურად გადაიქცეოდა კომენტარად.

შესაბამისად დღევანდელ გარემოში არსებულ ყველა ბრაუზერს აქვს Javascript_ის მხარდაჭერა და მიმდინარე კოდის გამოყენება აღარაა აქტუალური.

javascript_ში ინფორმაციის გამოტანის რამდენიმე მეთოდი არსებობს. მაგ.:

- alert('შეტყობინება 1') - ინფორმაციის გამოტანა დიალოგური ფანჯარის მეშვეობით;
- document.write('შეტყობინება 2') - ინფორმაციის გამოტანა ვებ გვერდის შემცველობაში;
- console.log('შეტყობინება 3')  - ინფორმაციის გამოტანა კონსოლში;

სანამ უშუალოდ სათითაო მეთოდს გავარჩევთ ზოგადად ავხსნათ მეთოდის არსი. მეთოდი იგივე ფუნქციაა. მისი მოხსენიება ორივენაირად შეიძლება. ჩემს მიერ გამოყენებული იქნება ორივე მიმართება და იმედია ეს არ გამოიწვევს რაიმე გაურკვევლობას.

პროგრამირებაში ამომავალი წერტილი **ობიექტია**. ნებისმიერი მოქმედება დაკავშირებულია კონკრეტულ ობიექტთან. ობიექტებზე ვრცლად შემდეგ ქვეთავებში ვისაუბრებთ ახლა მხოლოდ მცირეოდენი რაკურსით შემოვიფარგლოთ, რათა გაგიადვილდეთ ფუნქციის/მეთოდის არსის გაგება და არსებული ჩანაწერების დაზეპირება გაუაზრებლად არ მოახდინოთ.

მაგ.: გვაქვს ობიექტი ტელეფონი. მას გააჩნია **თვისებები** და **მეთოდები**. თვისებებში შეიძლება ვიგულისხმოთ წონა, ფერი, მეხსიერების ზომა და ა.შ. აი მეთოდები წარმოადგენს მის შესაძლებლობებს. ვთქვათ ტელეფონის მეშვეობით შესაძლებელია ფოტოსურათის გადაღება, შეტყობინების გაგზავნა.

რომ შევაჯამოთ ეს არც თუ ისე ვრცელი აბზაცი მივიღებთ შემდეგს:

- ობიექტის **თვისების** გასაგებად თქვენ სთხოვთ ობიექტს გამოგიტანოს/გიჩვენოს უკვე არსებული თვისება; მაგ.: გამოსახოს ეკრანზე ტელეფონის წონა.
- ობიექტის **მეთოდის** გამოყენებით შენ ზემოქმედებ ამ კონკრეტულ ობიექტზე და ახდენ რაიმე ტიპის მანიპულაციას; მაგ.: გაგზავნოს შეტყობინება თქვენს მიერ აკრეფილი ტექსტით.

შესაძლოა ეს ყოველივე ამ ეტაპზე სრულად გასაგები არ იყოს, ასევე არ წარმოადგენდეს იდეალური ახსნა ობიექტის თვისებასა და მეთოდს შორის განსხვავებისა, მაგრამ იქება მცირეოდენი ბიძგი მომავალში ყველაფრის უკეთ აღსაქმელად.

**თვისებასა** და **მეთოდს** გააჩნია ერთი ფუნდამენტალური პრინციპი ორივე მიეკუთვნება კონკრეტულ ობიექტს. კუთვნილება გამოისახება - წერტილით **„ . “** ;

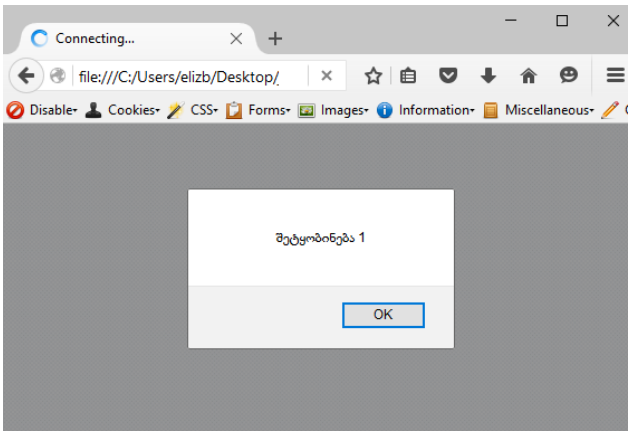
ასევე ყველა მეთოდი გამოისახება ფრჩხილებით-**( )**; ამ ეტაპზე არ ვეხებით თვისებებს და მათ გამოსახვას.

ჩვენს მიერ ჩამოთვლილ გამოტანის ოპერატორებზე ყურადღების გამახვილება კიდევ ერთხელ შეიძლება - **document.write**(), **console.log**() **,** **alert**() **;** როგორც თეორიული მასალიდან გავიგეთ პირობა:

- „ყველა მეთოდი აისახება ფრჩხილებით“ - ჭეშმარიტია.
- „მეთოდი მიეკუთვნება ობიექტს და გამოიყოფა წერტილით“ - პირველ ორ შემთხვევაში პირნათლად სრულდება. ანუ:
    - **document** - წარმოადგენს **ობიექტს** და შესაბამისად **write**() მისი ერთ-ერთი რიგითი მეთოდია - რომელიც პასუხისმგებელია ინფორმაციის გამოტანაზე ვებ გვერდის ტანში;
    - **console** -ასევე წარმოადგენს **ობიექტს** და მისი ერთ-ერთი მეთოდია **log**()- ინფორმაციის გამოტანა კონსოლში;
    - რაც შეეხება **alert()** მეთოდს ჩანაწერში არ ფიგურირებს ობიექტი და მასზე მიმაგრება წერტილის მეშვეობით. alert() მეთოდის სრულყოფილი ჩანაწერი შემდეგნაირია **window.alert();** ობიექტ window_ს მომავალში დაწვრილებით შევეხებით ამ ეტაპზე არსებული ჩანაწერით წარმოგიდგინეთ, რომ alert() არაფრით განსხვავდება ზემოხსენებული მეთოდებისაგან და ისიც ობიექტის მეთოდია უბრალოდ მასი შემოკლებული ჩანაწერი დასაშვებია ( ამ დეტალსაც შემდგომში გავარჩევთ );

Javascript რეგისტრზე დამოკიდებული ენაა შესაბამისად **Alert(), ALERT()** და ა.შ არსებული მეთოდების სხვადასხვაგვარი ინტერპრეტაციით დაწერა არ გამოიტანს სასურველ შედეგს. არსებული მეთოდები გამოიყენეთ მხოლოდ ასოთა იმ რეგისტრით, როგორიც მოცემულია მაგალითად.

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>

 <script>
          alert('შეტყობინება 1');
  </script>

</body>
</html>
```




5.1 alert() დიალოგური ფანჯარა

დავუბრუნდეთ ჩვენს მიერ უკვე აღწერილ მეთოდ alert() _ ს და ვნახოთ მისი გამოძახებით მიღებული შედეგი. ( იხ.სურ.5.1 )

გასათვალისწინებელია, რომ დიალოგური ფანჯარა alert() თანამედროვე ბრაუზერში განსხვავებული დიზაინით და მდებარეობით გამოიხატება.

ყურადღება გაამახვილეთ alert() მეთოდში ჩაწერილ სიტყვაზე - მას ფუნქციის **არგუმენტი** ეწოდება. მიმდინარე ეტაპზე მისი დამახსოვრება დიდ მნიშვნელობას არ წარმოადგენს, რეალურ არსს ფუნქციების განხილვისას გაეცნობით.

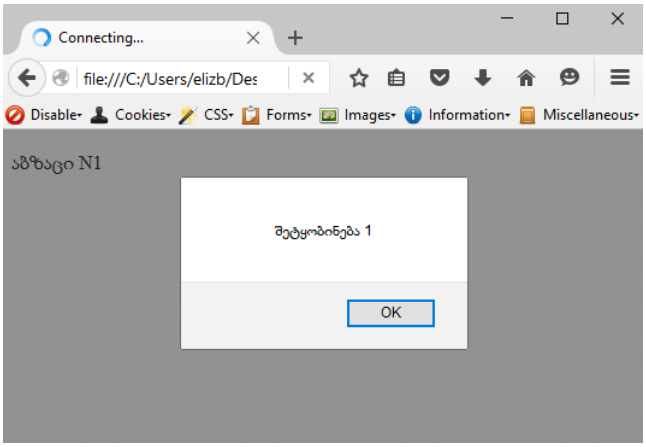
როგორც ატყობთ ტექსტი ბრჭყალებშია მოთავსებული. დასაშვებია როგორც ერთმაგი ასევე ორმაგი ბრჭყალების გამოყენება. მაგ.: 'შეტყობინება 1' და "შეტყობინება 1" ორივე მართებული ჩანაწერია. თუ მხოლოდ რიცხვითი მნიშვნელობა გამოსატანი დასაშვებია ბრჭყალების გარეშე მითითება. მაგ: **alert( 150 );**

alert() წარმოადგენს მეთოდს, რომელიც იწვევს html კოდის ჩატვირთვის შეყოვნებას.

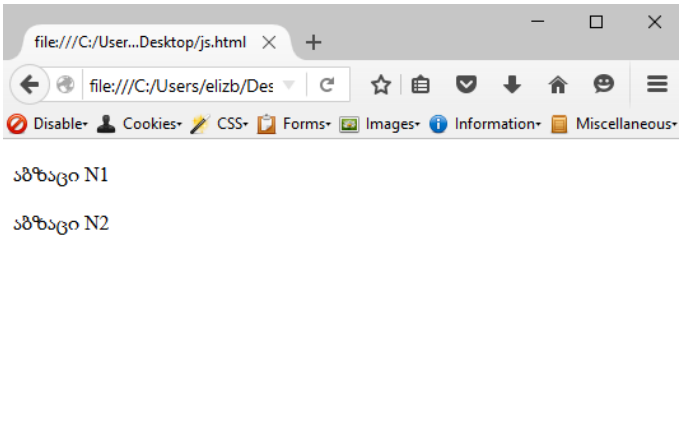
```
<!DOCTYPE HTML>
```

```
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <p> აბზაცი N1 </p>
 <script>
          alert('შეტყობინება 1');
  </script>
<p>აბზაცი N2</p>
</body>
</html>
```

კოდის მართლზომიერი მუშაობის შედეგად მონიტორზე გამოისახე alert() დიალოგური ფანჯრის მეშვეობით 'შეტყობინება 1'. თუ უკან ფონს დაუკვირდებით ბრაუზერის <body> ტეგში გამოსახულია წარწერა **„აბზაცი N1“** (იხ.სურ. 5.2), შესაბამისად **„აბზაცი N2“** არსად ფიგურირებას. ასევე ბრაუზერის ფანჯრის ზედა title ველში ჩატვირთვის ნიშანი აქტიურია, რაც მიუთითებს რომ ვებ გვერდის ჩატვირთვა არ დასრულებულა. მხოლოდ alert() ფანჯრის **„OK“** ღილაკით დადასტურება გამოიწვევს დანარჩენი კოდის ჩატვირთვას შემდგომ alert() ფუნქციამდე, თუ იგი კიდევ სადმე იარსებებს მიმდინარე ვებ გვერდზე. (გამომდინარე იქიდან, რომ ჩვენს შემთხვევაში გამოყენებულია მხოლოდ ერთი alert() ვებ გვერდი ბოლომდე ჩაიტვირთება ). გამოჩნდება ტექსტი **„აბზაცი N2“.** (იხ.სურ. 5.3)




5.2 კოდის ჩატვირთვის წყვეტა alert() მეთოდის მიერ

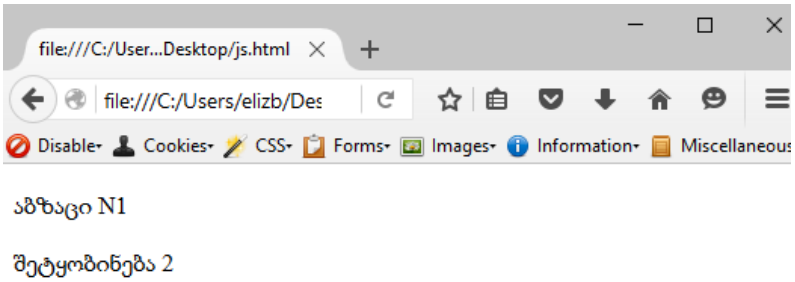



5.3 alert() მეთოდის დასტურის შედეგად მიღებული შედეგი

alert() მეთოდისაგან განსხვავებით document.write('შეტყობინება 2') არ წარმოადგენს მცურავ ფანჯარას, მას გამოაქვს ინფორმაცია ბეჭდური სახით, იმ კონკრეტულ სტრიქონზე, რომელზეც ის არის განთავსებული.

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
</head>
```

```
<body>
  <p> აბზაცი N1 </p>
 <script>
          document.write('შეტყობინება 2');
  </script>
</body>
</html>
```




დავალების შესაბამისად „აბზაცი N1“ ფორმირდება  ტეგის მეშვეობით, ხოლო „შეტყობინება 2“_ის გამოტანა ხორციელდება document.write() მეთოდის დახმარებით (იხ.სურ.5.4)

5.4 document.write() ფუნქციის გამოტანილი „შეტყობინება 2“

Javascript სინტაქსი ხაზოვანი ტიპის კითხვადობის რეჟიმის მიმდევარია. თითოეული პროგრამული სტრიქონის ბოლოს რეკომენდირებულია წერტილმძიმის - **„ ; “** გამოყენება, რაც სტრიქონის დასასრულის აღმნიშვნელია. (სხვადასხვა პროგრამირების ენაში - მაგ. php, c# „;“ გამოყენება სავალდებულოა ). სტრიქონში ედიტორის ერთი კონკრეტული ხაზზე განთავსებული ინფორმაცია არ მოაზრება, პროგრამირებაში სტრიქონად ერთი უწყვეტი კოდი იგულისხმება. იხილეთ მოცემული მაგალითი

```
1. <script>
2.       alert('შეტყობინება 1');
3.       document.write(' აქ განთავსდება ნებისმიერი სიგრძის ინფორმაციული ტექსტი და ის აღიქმება
         ერთ სტრიქონად მიუხედავ იმისა ედიტორში რამდენ ხაზზეა იგი განლაგებული');
4.  </script>
```

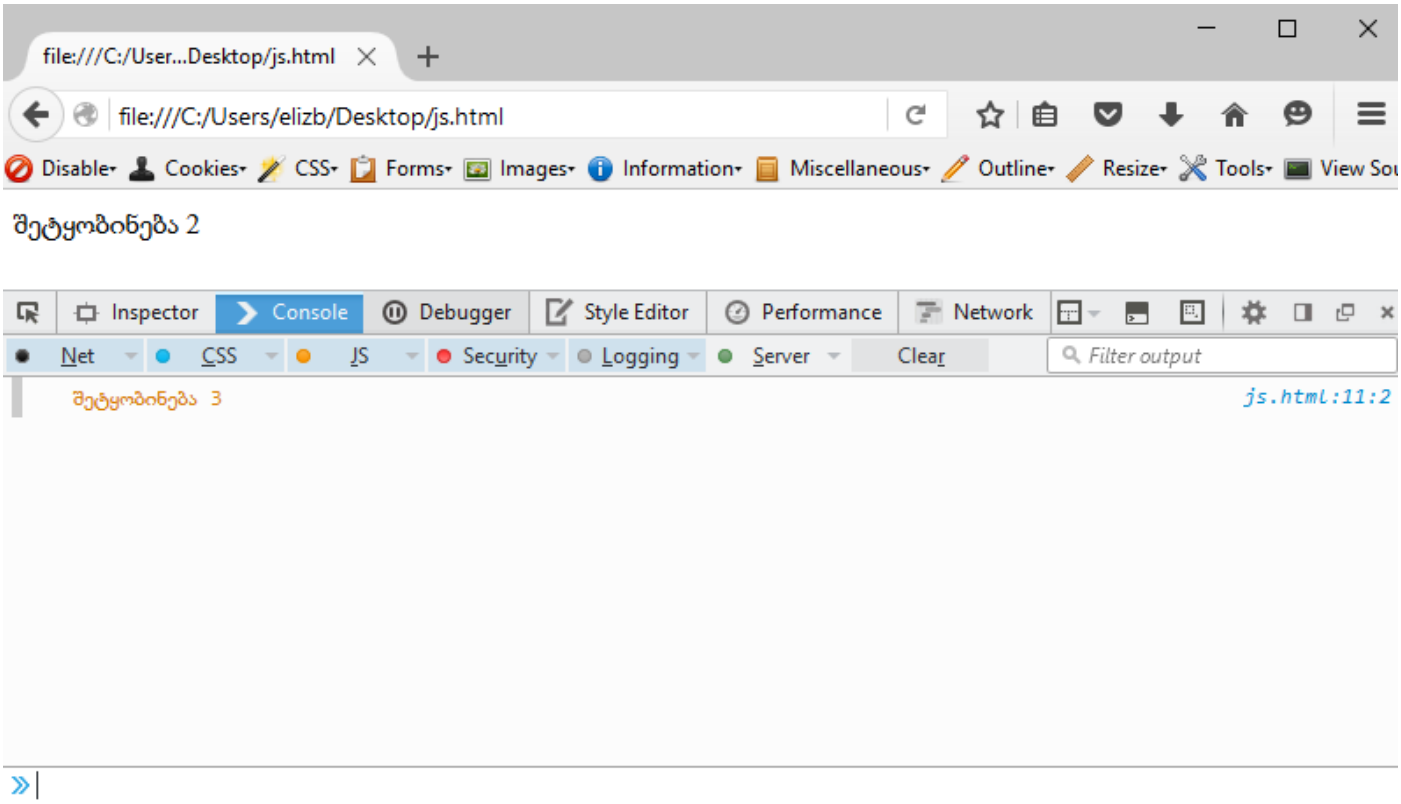
რეკომენდირებულია ასევე სხვადასხვა აზრობრივი კოდის სხვადასხვა ხაზზე განთავსება. მიუხედავად იმისა, რომ მაგალითზე ნაჩვენები კოდი უპრობლემოდ იმუშავებს უმჯობესია alert() ფუნქციები განთავსდეს სხვადასხვა სტრიქონზე.

```
<script>
          alert('შეტყობინება 1'); alert('შეტყობინება 2');
 </script>
```

```
<script>
          alert('შეტყობინება 1');
```

```
        alert('შეტყობინება 2');
    </script>
```

**console.log()** _ ის მეშვეობით გამოტანილი ინფორმაცია გამოისახება ბრაუზერის კონსოლში, რომლის გამოძახებაც შესაძლებელია **ბრაუზერზე მაუსის მარჯვენა კლიკი > Inspect Element** ან **F12**_ის მეშვეობით. (იხ.სურ.5.5). რა უპირატესობა გააჩნია კონსოლს თუ ის ხილვადია მხოლოდ ვებში გათვითცნობიერებული ადამიანისათვის. ვებ დეველოპერების უდიდესი ნაწილი მოიხმარს კონსოლს. რამეთუ კეთების პროცესში პროგრამისტი არ არის დაზღვეული შეცდომებისაგან. კონსოლი საშუალებას გაძლევთ მარტივად დაადგინოთ შეცდომის ტიპი და მდებარეობა მინიშნებების საშუალებით.




5.5 კონსოლის ვიზუალური გამოხატულება

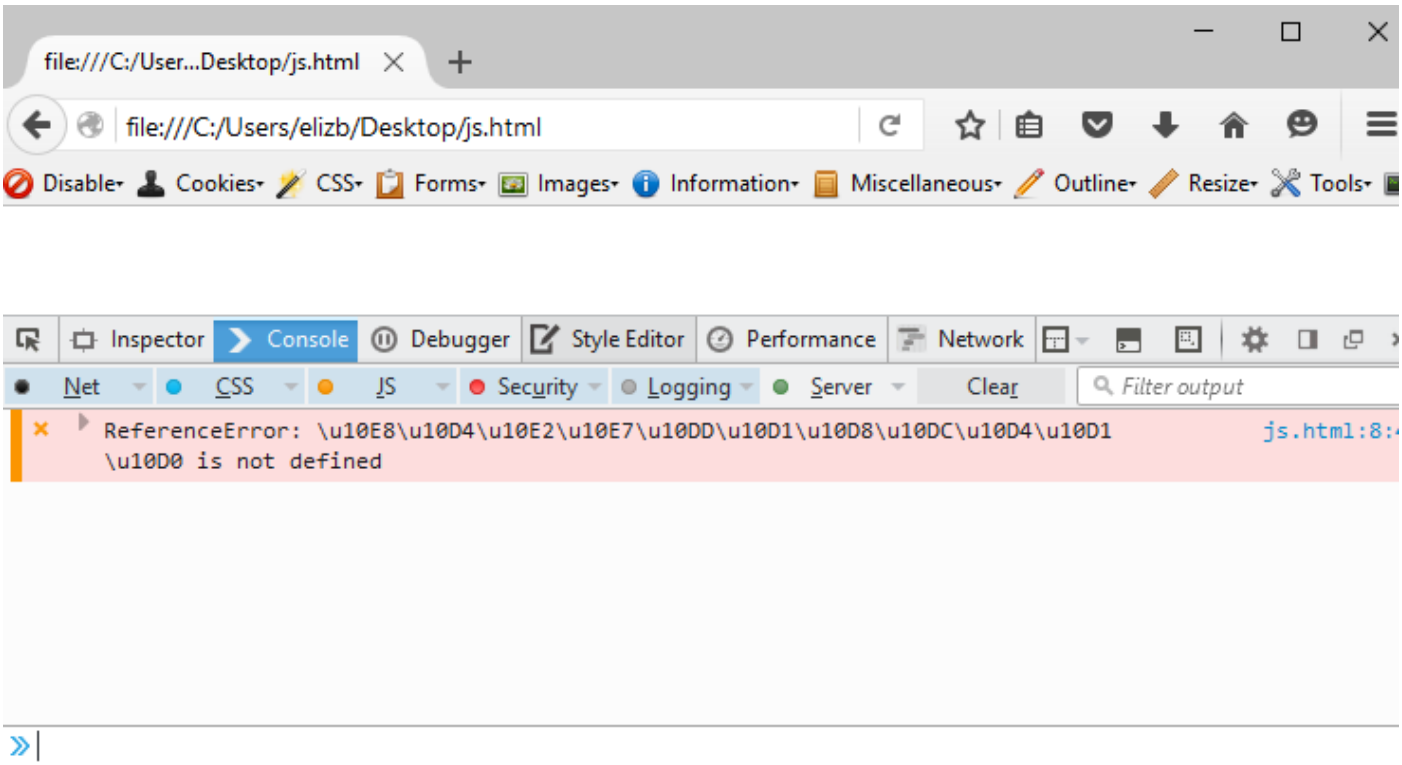
```
    <script>
        console.log('შეტყობინება 3');
    </script>
```

სურათ 5.5_ზე მოცემულია კონსოლი სადაც წინასწარ მაგალითზე ნაჩვენები console.log() მეთოდის მეშვეობით აისახა შედეგი.

დავუშვათ  განზრახ შეცდომა სკრიპტში, რათა მოვახდინოთ  კონსოლის რეაგირების დემონსტრირება. console.log() მეთოდში გადაცემული არგუმენტი **შეტყობინება 3** ბრჭყალების გარეშეა. როგორც აღვნიშნეთ ეს უპატიებელი შეცდომაა. ვნახოთ კონსოლის რეაგირება (იხ.სურ.5.6)

```
<!DOCTYPE HTML>
```

```
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
 <script>
          console.log(შეტყობინება 3);
  </script>
</body>
</html>
```




5.6. კონსოლის რეაირება შეცდომაზე

ძირითად ნაწილში განსაზღვრულია შეცდომის ტიპი, ხოლო მარჯვენა კიდეში ასახულია სტრიქონის ნომერი, თუ რომელ ხაზზეა შეცდომა დაშვებული js.html:8

## კომენტარები

Javascript_ში ისევე როგორც თქვენს მიერ აქამდე შესწავლილ სხვა ენებში (html, css) აქტიურად გამოიყენება კომენტარების სისტემა.

დიდი კოდის არსებობის შემთხვევაში ბევრი რამ ბუნდოვანი ხდება მისი შემდგომი გარჩევისას. ამიტომ ხშირად გამოიყენებენ კომენტარების სისტემას, რომელიც მიანიშნებს კონკრეტული სტრიქონის ან ლოგიკური ბლოკის დანიშნულებას.

კომენტარები შესაძლოა გამოყენებულ იქნას სკრიპტის ნებისმიერ ადგილას და ის არანაირ უარყოფით ზეგავლენას არ ახდენს კოდის მუშაობაზე, პირიქით დადებით ტონად გასდევს დეველოპერის ნამუშევარს.

**ერთ ხაზიანი** კომენტარები აღინიშნება // სიმბოლოთი და ზემოქმედებს მხოლოდ 1 სტრიქონზე არსებულ ინფორმაციაზე. დაიმახსოვრეთ: ერთ ხაზში მოაზრება უწყვეტი ტექსტი, რომელიც ედიტორში შესაძლოა წარმოდგენილი იყოს რამდენიმე სტრიქონადაც. ახალი სტრიქონის დასაწყისს უზრუნველყოფს კლავიში **ENTER** აქტივაცია.

```
 // არსებულ მეთოდს გამოაქვს დიალოგური ფანჯარა
alert( 'შეტყობინება 1' );
alert( 'შეტყობინება 2' ); // მიღებულია კომენტარის ამ ადგილას განთავსებაც
```

არსებობს ასევე **მრავალ ხაზიანი** კომენტარი, რომელსაც შესაბამისად გააჩნია გამხსენი ( **/*** ) და დამხურავი ( ***/** ) სიმბოლოების ერთობლიობა. მოცემულ მაგალითზე შეტყობინება 2 და 3 არ გამოისახება.

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
<script>
            /*
            document.write('შეტყობინება 2');
            document.write('შეტყობინება 3');
            */
            document.write('შეტყობინება 4');
  </script>
</body>
</html>
```

## ცვლადები და მათი ტიპები

ისევე, როგორც პროგრამირების დანარჩენი ენებში Javascript_შიც ინტენსიურად გამოიყენება ტერმინი **ცვლადები.**

რას წარმოადგენს ცვლადი?

ცვლადი არის პირობითად კონტეინერი, რომელსაც შეუძლია შეინახოს გარკვეული მონაცემი. ყოველ ცვლადს გააჩნია **სახელი**, ხოლო შენახულ მონაცემს ცვლადის **მნიშვნელობა** ეწოდება.

ახლი ცვლადის გამოსაცხადებლად javascript_ში გამოიყენება საკვანძო სიტყვა var. ის იშიფრება როგორც **variable** - ცვლადი.

როგორც აღვნიშნეთ ცვლადს გააჩნია სახელი. Javascript_ში ცვლადის გამოცხადებისას ყურადღება უნდა გაამახვილოთ ცვლადის სახელებზე, რადგანაც სახელი შეიძლება შედგებოდეს მხოლოდ: **სიმბოლოების $ და _ , ასოების, ციფრებისაგან.** ამასთანავე დაუშვებელია ცვლადის სახელის დაწყება ციფრით.

მაგ. I_ზე მოყვანილია ცვლადის გამოცხადების სწორი გზა, ხოლო მაგ. II მცდარი.

მაგ. I

```
// არსებული მეთოდით ცვლადების გამოცხადება სწორია

 var myName;
var test123;
var $;
var _;
```

მაგ. II

```
// არსებული მეთოდით ცვლადების გამოცხადება მცდარია

var 123test;  // ცვლადის სახელის რიცხვით დაწყება არ შეიძლება;
   var my-name;   //  არ შეიძლება სიმბოლო ‘ - ’  ტირეს გამოყენება მხოლოდ ქვედა ტირე;
```

რაც შეეხება ცვლადების სახელების მიმართ გამოთქმულ რეკომენდაციებს მდგომარეობს შემდგომში:

- გამოიყენეთ მხოლოდ ლათინური ანბანის ასოები;
- უმჯობესია თუ მოერიდებით რამდენიმე სიტყვიანი ცვლადებს ან მათი გამოყენებისას მაქსიმალურად შეზღუდეთ თუნდაც დასაშვები სიმბოლო „_“. მაგ. უმჯობესია გამოიყენოთ myFirstName ვიდრე my_first_name;
- მაქსიმალურად მიუახლოვეთ ცვლადის სახელი მასში შესანახ მნიშვლობის არსთან;

Javascript_ში არსებობს რეზერვირებული სიტყვები, რომელთა სახელებად გამოყენება ასევე დაუშვებელია. ცხრილ I_ში გამოსახულია  არსებულ სიტყვათა ნუსხა და პრობლემების თავიდან ასაცილებლად შეეცადეთ არ გამოიყენოთ ჩამონათვალიდან არც ერთი. ამ სიტყვების ერთიანად

დაზეპირება სავალდებულო არ არის. გზადაგზა უმეტეს მათგანს შეეხებით და დაიმახსოვრებთ. შესაძლოა ასევე მათი ინტერნეტში მოძიება შემდეგი ტექსტის მეშვეობით - **„Javascript reserved words“**

| ცხრილი I | | | |
|---|---|---|---|
| abstract | else | instanceof | switch |
| boolean | enum | int | synchronized |
| break | export | interface | this |
| byte | extends | long | throw |
| case | false | native | throws |
| catch | final | new | transient |
| char | finally | null | true |
| class | float | package | try |
| const | for | private | typeof |
| continue | function | protected | var |
| debugger | goto | public | void |
| default | if | return | volatile |
| delete | implements | short | while |
| do | import | static | with |
| double | in | super | |

როგორ ზედა ქვეთავში იყო განხილული ცვლადი შედგება 2 ძირითადი ნაწილისაგან. ჩვენს მიერ გარჩეულ იქნა ცვლადის სახელი. ახლა გადავიდეთ ცვლადის **მნიშვნელობაზე**.

მაგალითად ისეთ პროგრამირების ენებში, როგორებიცაა : C++, C# ცვლადის გამოცხადებისას ხდება მისი ტიპის განსაზღვრა და მასში მხოლოდ მითითებული ტიპის მონაცემების ჩალაგება შესაძლებელი. იხ. მაგალითი

```
int n;  // გამოცხადებული ცვლადი n შეიძლება შეიცავდეს მხოლოდ  მთელ რიცხვებს
 string txt;  // ცვლადი txt შეიძლება შეიცავდეს მხოლოდ  ტექსტუალურ მნიშვნელობას

// და ა.შ;
```

Javascript_ში არ გვხვდება მკაცრად გამოხატული ტიპიზაცია. რას ნიშნავს ეს ყოველივე ?!

ეს ნიშნავს შემდგომს - Javascript_ში შესაძლებელია ერთი და იგივე ცვლადისათვის სხვადასხვა ეტაპზე სხვადასხვა ტიპის მნიშვნელობის მინიჭება, ისე რომ არავითარი „ტიპზე დაყვანის ოპერაციები“ არ არის საჭირო, ის ავტომატურად გარდაიქმნება იმ ტიპის ცვლადად რა ინფორმაციასაც მიიღებს.

ზოგადად პროგრამირების ენები, რომლებიც არ ხასიათდება მკაცრად გამოხატული ტიპიზაციით მაგ. Javascript, php  ამ ამოცანას აკისრებენ ინტერპრეტატორს, ამიტომაც Javascript_ში მათ გამოსაცხადებლად საკმარისია მხოლოდ უკვე ნახსენები საკვანძო სიტყვა var გამოყენება.

```
 var myName;      // მისთვის შესაძლებელია ნებისმიერი ტიპის მნიშვნელობის მინიჭება
```

პროგრამირების ენებში არსებობს სხვადასხვა ტიპის ოპერატორები: **არითმეტიკული, ლოგიკური, ბინარული, სტრიქონული, პირობითი.** მათ სრულყოფილად ორიოდ წუთში მივუბრუნდებით ამ ეტაპზე კი მხოლოდ მინიჭების ოპერატორ **„=“** შევეხოთ. დაიმახსოვრეთ ეს ნიშანი ‘=’ პროგრამირებაში არ გამოხატავს „ტოლობას“ ის არის მინიჭება.

ცვლადში დროებითი მნიშვნელობის განსათავსებლაბლად საჭიროა დავწეროთ მაგალითის შესაბამისი გამოსახულება;

```
var myCar = 'Ford Mustang GT 500 Eleanor';

var price = 550000;
```

ყურადღება გაამახვილეთ შემდეგზე: ცვლადი myCar არ უდრის, არ არის ტოლ, მას უბრალოდ ენიჭება 'Ford Mustang GT 500 Eleanor' მნიშვნელობა. თანდათან უფრო ცხადი წარმოდგენა შეგექმნებათ მინიჭების ოპერატორზე.

ახლა უშუალოდ რაც შეეხება ამ კონკრეტულ ჩანაწერთა ერთობლიობას. ერთი შეხედვით სხვაობა არ არის არანაირი უბრალოდ myCar_ის მნიშვნელობა მოთავსებულია ბრჭყალებში, ხოლო price_ის არა. ეს მათი ტიპების დამსახურება.

Javascript_ში  ტიპები დაყოფილია 2 ჯგუფად:

1. პრიმიტიული ანუ მარტივი ტიპები:
    a. String
    b. Number
    c. Boolean
    d. Null
    e. Undefined
2. შედგენილი ტიპი - **Object;**

Null და Undefined ტიპებს ხშირად სპეციალურ ტიპებადაც მოიხსენიებენ.

გავეცნოთ პრიმიტიულ ტიპებს დაწვრილებით, ხოლო **Object** გადავდოთ მომავალისათვის.

### String - ტექსტური ტიპი

ტექსტური ტიპი აერთიანებს ყველა სახის ტექსტს. კანონია ყველა ტექსტური ტიპის მნიშვნელობა უნჯა იჯდეს **ერთმაგ** ან **ორმაგ** ბრჭყალებში. Javascript_არ არის სხვაობა არსებულ ბრჭყალებს შორის. იხ. მაგ.

```
var txt = "აქ განთავსდება ტექსტური მნიშვნელობა";      // ორმაგი ბრჭყალი
txt = 'აქ განთავსდება ტექსტური მნიშვნელობა';           // ერთმაგი  ბრჭყალი
```

### Number - რიცხვითი ტიპი

რიცხვითი ტიპი თავის თავში აერთიანებს, როგორც  **მთელ,** ასევე **ათწილად** რიცხვებს. არსებობს ასევე სპეციალური რიცხვითი ტიპები - **`Infinity`** , **`+ Infinity`** , **`NaN`**.

```
var n = 123;
n = 12.345;
```

**`Infinity`** წარმოადგენს უსასრულობას;

```
// მაგ.: 1_ის  ნულზე გაყოფის შედეგად მიიღება უსასრულობა;

alert( 1 / 0 );  // +Infinity
alert( -1 / 0 ); // -Infinity
```

**`NaN`** - იშიფრება როგროც -  `Not a Number`, ანუ რიცხვითი მნიშვნელობა, რომელიც არ უდრის არც ერთ სხვა რიცხვს, ასევე არ უდრის თავის თავსაც. შესაძლოა ცოტა გაუგებარია?! ეს არის ტიპი რომელიც რამენაირ კავშირშია რიცხვთან მაგრამ ამავდროულად არ არის რიცხვი. მაგ.: რიცხვისა და

ტექსტის არითმეტიკულ ნამრავლ არის **NaN**, ასევე მსგავსი ტიპი მიიღება კვადრატულ ფესვი -1 დან და ა.შ.

```
alert( "ტექსტი" * 15 ); // NaN
```

## Boolean - ბულის ანუ ლოგიკური ტიპი

ის თავის თავში აერთიანებს მხოლოდ 2 მნიშვნელობას **true** - ჭეშმარიტია , **false** - მცდარია;

```
var checked = true; //  მიმდინარე სტრიქონზე  ცვლადი ჭეშმარიტია
checked = false;    // მიმდინარე სტრიქონზე  ცვლადი მცდარია
```

## Null - მნიშვნელობა, რომელიც არ არსებობს

არსებული ტიპი არ განეკუთვნება არც ერთ ტიპს. ზოგადად მიჩნეულია, რომ null წარმოადგენს ობიექტის ტიპს და მიუთითებს ობიექტის არ არსებობას.. დაიმახსოვრეთ null არ არის 0. ის არის შემთხვევა როცა კონკრეტული ცვლადი უდრის არაფერის;

```
var age = null;
```

## Undefined - მნიშვნელობა განუსაზღვრელია

არსებული ტიპი მხოლოდ ერთ მნიშვნელობას შეიცავს და ეს არს „მნიშვნელობა განუსაზღვრელია“.
თუ მოვახდენთ ცვლადის დეკლარირებას მაგრამ მას არ მივანიჭებთ საწყის ეტაპზე არაფერს არ წარმოადგენს არანაირ პრობლემას. უბრალოდ თუ შევეცდებით მის ეკრანზე გამოტანას მნიშვნელობად გვიჩვენებს undefined

```
var x;
alert( x ); // გამოიტანს "undefined"
```

არსებობს სპეციალური ოპერატორი და ფუნქცია **typeof,** რომელსაც შეუძლია მონაცემის ტიპის დადგენა და მისი ჩანაწერი შემდეგნაირია:

- ოპერატორის სინტაქსი **typeof x;**
- ფუნქციის სინტაქსი **typeof(x);**

ორივე შემთხვეაში **typeof** მუშაობს იდენტურად. გამოიყენეთ რომელიც გსურთ

```
alert(typeof undefined) // "undefined"

alert(typeof 0) // "number"
```

```
alert(typeof true) // "boolean"

alert(typeof "foo") // "string"

alert(typeof null) // "object"
```

შესაძლოა კონკრეტული მნიშვნელობის მაგივრად გადავაწოდოთ ცვლადად

```
var x = 2.25;

alert(typeof x ) // " number "
```

## ცვლადის გამოცხადების გზები

ჩვენს მიერ განხილულ იქნა ცვლადის სახელის შერჩევა, var საკვანძო სიტყვა, ცვლადის მნიშვნელობა შესაბამისად შეგვიძლია ცვლადის გამოცხადების გზებზე საუბარი.

შესაძლებელია ცვლადის გამოცხადება მნიშვნელობის გარეშე და ასევე მასთან ერთად

```
var x;                  // ცვლადის გამოცხადება მნიშვნელობის გარეშე
var y;                  // ცვლადის გამოცხადება მნიშვნელობის გარეშე

var age = 15;                           // ცვლადის გამოცხადება - მნიშვნელობის მინიჭებით
var city =  "თბილისი";          // ცვლადის გამოცხადება - მნიშვნელობის მინიჭებით
```

შესაძლებელია ასევე ცვლადების ჩამოთვლა და მათი ერთდროული დეკლარირება 1 var სიტყვის საშუალებით, გამყოფებად კი გამოიყენება მძიმე “,” .

```
var x,      age = 15, city =  "თბილისი";
```

მრავალჯერ აღნიშნული მიუხედავად კვლავ დავკონკრეტდები, რომ Javascript ხაზოვანი კითხვადობის მიმდევარია.

```
var  age = 15;          // ცვლადისთვის პირველადი დეკლარირება;

alert(age);             // შედეგი ამ ეტაპზე 15

age = 103;// ცვლადისთვის ყოველ შემდეგ ჯერზე მნიშვნელობის მინიჭებისას სიტყვა var  აღარ არის საჭირო, არსებული სტრიქონის ქვემოთ გამოყენებული ცვლადის მნიშვნელობა ყველგან იქნება 103. ახალი მნიშვნელობის მინიჭებამდე
```

```
alert(age);            // შედეგი ამ ეტაპზე  103
```

### არითმეტიკული ოპერატორები

არითმეტიკული მოქმედებები გამოიყენება რიცხვითი ტიპის ცვლადებზე. ცხრილ II მოცემულია Javas-cript_ში არსებული არითმეტიკული მოქმედებები.

ცხრილი II

| ოპერატორი | აღწერა | მაგალითი | y მნიშვნელობა | შედეგი |
|---|---|---|---|---|
| + | მიმატება | x = y + 2 | y = 5 | x = 7 |
| - | გამოკლება | x = y - 2 | y = 5 | x = 3 |
| * | გამრავლება | x = y * 2 | y = 5 | x = 10 |
| / | გაყოფა | x = y / 2 | y = 5 | x = 2.5 |
| % | მოდულით გაყოფა | x = y % 2 | y = 5 | x = 1 |
| ++ | ინკრიმენტი | x = ++y | y = 5 | x = 6 |
| | | x = y++ | y = 5 | x = 5 |
| -- | დეკრიმენტი | x = --y | y = 5 | x = 4 |
| | | x = y-- | y = 5 | x = 5 |

პირველი ოთხი ოპერატორი არ საჭიროებს დამატებით განმარტებას, მათთან შეხება სკოლაში საწყისი ასაკიდანვე გიწევდათ.

რაც შეეხება **%** ნიშანს მისი მოქმედება ოდნავ განსხვავებულია და ბევრისთვის შეიძლება გამოცანად რჩება ტერმინი მოდულით გაყოფა. მარტივად, რომ ვთქვათ მოდულით გაყოფისას შედეგი მიიღება ნაშთი. თუ პასუხად მოგვცა 0 ე.ი. გასაყოფი გამყოფზე იყოფა უნაშთოდ.

```
var  x = 19 % 4;      // 3 - ნაშთი
     x = 20 % 4;      // 0  და არა 5 რადგანაც გაყოფა მოხდა უნაშთოდ
```

ინკრიმენტი და დეკრიმენტი პროგრამირებაში ფართოდ გამოყენებადი ოპერატორებია. რეალურ რეჟიმში ისინი შემოკლებულ ჩანაწერს წარმოადგენენ. მაგ. ინკრიმენტი: **i++** იგივეა, რაც **i+1** , ასევე **++i** იდენტურია **i+1** ერთი არც თუ ისე უმნიშვნელო განსხვავებისა: **i++** ზრდის ცვლად i მნიშვნელობას 1_ით მხოლოდ **მომავალ სტრიქონზე,** ხოლო **++i მიმდინარეზე,** სადაც მოხდა მისი გამოყენება;

მაგ. I

```
var  i = 5;     // საწყის მნიშვნელობად  i  = 5
  alert(i);     //  5
  alert(i++); //  ამ ეტაპზე i  ისევ 5_ია  ის განიცდის მომატებას და გაზრდილ მნიშვნელობად მხოლოდ ზემდეგი სტრიქონიდან ფიქსირდება
    alert(i);     //  აქ უკვე  i გახდა 6 და  ყველა შემდეგ ხაზზე მისი მნიშვნელობა 6_ია თუ უცვლელად დავტოვებთ
```

მაგ. II

```
var  i = 5;     // საწყის მნიშვნელობად  i  = 5
  alert(i);     //  5
  alert(++i); //  აქ უკვე  i გახდა 6  და  ყველა შემდეგ ხაზზე მისი მნიშვნელობა 6_ია თუ უცვლელად დავტოვებთ
    alert(i);     //  6
```

რაც შეეხება დეკრიმენტს ინკრიმენტის მსგავსად მუშაობს უბრალოდ ახორციელებს 1_ით **კლებას.**

მაგ. I

```
var  i = 5;     // საწყის მნიშვნელობად  i  = 5
  alert(i);     //  5
  alert(i--); //  ამ ეტაპზე i  ისევ 5_ია ის განიცდის კლებას  და დაკლებულ  მხოლოდ ზემდეგი სტრიქონიდან ფიქსირდება
    alert(i);     //  აქ უკვე  i გახდა 4  და  ყველა შემდეგ ხაზზე მისი მნიშვნელობა 4_ია თუ უცვლელად დავტოვებთ
```

მაგ. II

```
var  i = 5;     // საწყის მნიშვნელობად  i  = 5
  alert(i);     //  5
  alert(--i); //  აქ უკვე  i გახდა  4  და  ყველა შემდეგ ხაზზე მისი მნიშვნელობა 4_ია თუ უცვლელად დავტოვებთ
    alert(i);     //  4
```

როგორც ქვეთავის დასაწყისში აღვნიშნეთ არსებული არითმეტიკული მოქმედებები იდეალურად მუშაობენ მხოლოდ ციფრებთან მიმართებაში. რაც შეეხება მათ მუშაობას ტექსტური (string) ტიპიც ცვლადებთან ცოტა განსხვავებულად მოქმედებენ.

მაგალითად თუ მოცემულია შემთხვევა

```
var  i = "თბილისი" * 5;            //  NaN
```

ყველა არითმეტიკული მოქმედება გარდა **„+“** შედეგად დააბრუნებს NaN, რაც ცხადზე ცხადია გამომდინარე იქიდან, რომ ტექსტუალური ტიპის ინფორმაციაზე არითმეტიკული მოქმედებების გამოყენება დაუშვებელია. რაც შეეხება „+“ ორიოდ წუთში დავუბრუნდებით. მაგრამ ვთქვათ ტექსტუალური ტიპით წარმოვადგინეთ რიცხვი რა მოხდება ამ შემთხვევაში? ინტერპრეტატორი ავტომატურად მოახდენს ტიპის დაყვანას თუ ის შესაძლებელია

```
var  i = "6" * 5;            // 30
     i = "11" - 3;           //  8
     i = "10" / 2;           //  5
     i = "10" / 4;           //  2
```

რაც შეეხება “+” ნიშანს გარდა არითმეტიკული მოქმედებისა javascript_ში ის წარმოადგენს **კონკატინაციის / მიერთების** ნიშანს. თუ ეს ნიშანი მოხვდება ტექსტუალურ მონაცემთან ის ავტომატურად მიერთებას მოახდენს ხოლო თუ რიცხვით ტიპთან არითმეტიკულ მოქმედებას. ამიტომ საწყის ეტაპზე ხშირად იწვევს დაბნეულობას როცა არითმეტიკული შეკრება სულ სხვა შედეგით სრულდება ვიდრე ამას მოელოდით. მაგ.:

```
var  i = "6" + 5;            // 65
         i = "25" + "23";    // 2523
```

როგორც ზედა მაგალითიდან გამოჩნდა „+“ ნიშნის გამოყენებით მიღებული შედეგი NaN არ არის, არამედ მოხდა ტექსტების ერთმანეთის გვერდიგვერდ დალაგება. რა თქმა უნდა შესაძლებელია ტექსტური ინფორმაციის რიცხვითში გადაყვანა სპეციალური ფუნქციების გამოყენებით, მაგრამ არსებობს ასევე საკმაოდ მარტივი გზაც, რომელიც მაგალითზეა ნაჩვენები

```
var  i = +"6" + 5;           // 11
         i = +"25" +  +"23";   // 48
```

„+“ ნიშანი, როგორც აღვნიშნეთ კონკატინაციას წარმოადგენს და მისი გამოყენება საკმაოდ ინტენსიურად ხორციელდება. მაგ.: მოცემული ცვლადებით მიიღეთ სრულფასოვანი წინადადება და დაბეჭდეთ ეკრანზე.

```
var  name= 'თედო';
var  txt = 'საოცნებო მანქანა';
var  myCar = 'Ford Mustang GT 500 Eleanor';

// საოცნებო მანქნაFord Mustang GT 500 Eleanor
document.write(txt + myCar   );          // ორი ცვლადის ტექსტი წვეტის გარეშე მიმეწებება ერთმანეთს

// საოცნებო მანქანა Ford Mustang GT 500 Eleanor
document.write(txt +' - '+ myCar   );
```

```
// Ford Mustang GT 500 Eleanor  - საოცნებო მანქანა
document.write(myCar + ' - '+txt  );

// Ford Mustang GT 500 Eleanor ჩემი  საოცნებო მანქანა
document.write(myCar + ' ჩემი '+txt+ ' ა'  );

// თედოს საოცნებო მანქანა არის Ford Mustang GT 500 Eleanor
document.write(name + 'ს '+ txt+ ' არის '+ myCar );
```

### მინიჭების ოპერატორები

ზედა ნაწილში განხილულ იქნა მინიჭების ოპერატორი „=“. არსებობს სხვა ოპერატორებიც, რომლებით აერთიანებს მინიჭების და არითმეტიკულ ოპერატორებს და საკმაოდ ამარტივებს მიდგომას. ცხრილი III განხილულია შემთხვევა სადაც საწყის მნიშვნელობებად **x = 10** და **y = 5.**

ცხრილი III

| ოპერატორი | მაგალითი | მსგავსი  გამოსახულება | შედეგი |
|---|---|---|---|
| = | x = y | x = y | x = 5 |
| += | x += y | x = x + y | x = 15 |
| -= | x -= y | x = x - y | x = 5 |
| *= | x *= y | x = x * y | x = 50 |
| /= | x /= y | x = x / y | x = 2 |
| %= | x %= y | x = x % y | x = 0 |

### შედარების ოპერატორები

პროგრამირებაში დიდია შედარების ოპერატორების გამოყენების ინტენსივობა. ხშირია პროგრამის შემუშავების პროცესში პირობის წამოჭრა, რომლის ჭეშმარიტების დასადგენად შედარების პრინციპის გამოყენებაა საჭირო. ნებისმიერი შედარების შედეგად მიღებული შედეგი წარმოადგენს Boolean ტიპის მნიშვნელობას: ის არის ჭეშმარიტი true ან მცდარი false.იხ. ცხრილი IV

ცხრილი IV

| ოპერატორი | აღწერა | პირობა | შედეგი |
|---|---|---|---|

| | | | |
|---|---|---|---|
| == | ტოლობა | 5 == 8 | false |
| | | 5 == 5 | true |
| === | ექვივალენტურია, ანუ ტოლია როგორც მნიშვნელობით ასევე ტიპიც საერთო აქვთ | 5 === "5" | false |
| | | 5 === 5 | true |
| != | არ უდრის | 5 != 8 | true |
| !== | არა ექვივალენტურია | 5 !== "5" | true |
| | | 5 !== 5 | false |
| > | მეტია | 5 > 8 | false |
| < | ნაკლებია | 5 < 8 | true |
| >= | მეტია ან უდრის | 5 >= 8 | false |
| <= | ნაკლებია ან უდრის | 5 <= 8 | true |

რიცხვითი მნიშვნელობების დროს შედარების ოპერატორებზე ფაქტობრივსა სასაუბრო არც არაფერი იმდენად მარტივია ყველაფერი. რაც შეეხება ტექსტური ტიპის მნიშვნელობებს არც მათი შედარებაა წარმოუდგენლად რთული. გამომდინარე იქიდან, რომ თითოეულ სიმბოლოს საკუთარი ASCII კოდი შეესაბამება სირთულეც მოხსნილია. ყურადღება გაამახვილეთ, რომ დადარების პროცესი ხდება ინდივიდუალურად თითოეული სიმბოლოს მიხედვით და ტექსტის დიდი სიგრძე ყოველთვის უპირატესობას არ წარმოადგენს. როგორც მაგალითზეა მოცემული ერთ შემთხვევაში **“a” > “aba”** მცდარია, ხოლო **“b” > “aba”** ჭეშმარიტი. სწორედაც რომ ასეა. გამომდინარე იქიდან რომ დადარება უშალოთ თითოეული სიმბოლოსი ხდება მეორე შემთხვევაში **“b”** შეედარა “**aba**” სიტყვის პირველ ასო “**a**” აღმოჩნდა მასზე მეტი და დაჯაბნა მთლიანი სიტყვა დანარჩენ ასეობზე დადარება საერთოდ აღარ ჩათვალა საჭიროდ.

| | |
|---|---|
| “a” > “aba” | false |
| “b” > “aba” | true |
| “abas” > “aba” | true |

## ლოგიკური ოპერატორები

ლოგიკურ ოპერატორებს საკმაოდ მნიშნველოვანი როლი ენიჭება პირობით ოპერატორებში. მათი დახმარებით დეველოპერებს ეძლევათ საშუალება პირობა უფრო დააკონკრეტონ. ლოგიკური ოპერატორები წარმოდგენილია ცხრილი V სადაც x = 6 და y = 3;

| ცხრილი V | | | |
|---|---|---|---|
| **ოპერატორი** | **აღწერა** | **პირობა** | **შედეგი** |
| ! | უარყოფა | !(x == y) | true |
| && | ლოგიკური „და“ | (x < 10 && y > 1) | true |
| \|\| | ლოგიკური „ან“ | (x == 5 \|\| y == 5) | false |

## 8.2. ამოცანის გადაჭრა ძირითადი კონსტრუქციების გამოყენებით

**მიმდინარე პარაგრაფის თემატიკა**

- ამოცანის გადაჭრის ალგორითმის შემუშავება
- ენის ძირითად კონსტრუქციები

### პირობითი ოპერატორები

პროგრამირებაში საკმაოდ დიდი როლი უკავიათ პირობით ოპერატორებს. ამოცანების უდიდეს ნაწილში პირობის დასმა და შესაბამისი ლოგიკის მიღება ფაქტობრივად წინა პლანზეა წამოწეული. Javascript_ში გვხვდება **if** და **switch** კონსტრუქციები. ამათგან if უფრო ფართოდ გამოყენებადი და ბევრისთვის საყვარელი ოპერატორია, მაგრამ არც switch_ს უნდა დავუკარგოთ ის დადებითი რაც გააჩნია. გზადაგზა უფრო ვრცლად მაგალითებზე დაყრდნობით გავეცნობით თითოეულ მათგანს.

if ოპერატორის რამდენიმე სახის ჩანაწერი არსებობს. ყველაზე გავრცელებული შემდეგი სკრიპტი

```
if ( x  > 10) {
 alert( 'გამოიტანოს  შედეგი' )
}
```

ნებისმიერი სახით if ოპერატორის გამოყენება, იქნება ეს ფიგურული ფრჩხილებით, მის გარეშე თუ სხვა მეთოდით უცვლელი რჩება არსი: **if ოპერატორს ჭირდება სტანდარტული ფრჩხილები ( ... ) პირობის დასასმელად;**

```
if (პირობა ) {
        ბლოკი რომელიც სრულდება პირობის დაკმაყოფილების შემთხვევაში
}
```

ბლოკში არსებული ინფორმაცია იმუშავებს მხოლოდ იმ შემთხვევაში თუ შესრულება if_ის ფრჩხილებში () დასმული პირობა, ანუ პასუხი იქნება ჭეშმარიტი - true.

როგორც დასაწყისში აღვნიშნეთ შესაძლებელია if ჩაწერა ბლოკის - ფიგურული ფრჩხილების გარეშე თუ მხოლოდ ერთი ხაზია მნიშვნელობა

```
if ( 20 > 10)
alert( 'შედეგი 1' )  // ეკრანზე გამოისახება შედეგი 1
```

თუ შედეგი რამდენიმეა მაგ.:

```
if ( 20  > 10)
alert( 'შედეგი 1' )  // ეკრანზე გამოისახება შედეგი 1
alert( 'შედეგი 2' )  // ეკრანზე გამოისახება შედეგი 2
```

მაგრამ იმ შემთხვევაში თუ if პირობა არ დაკმაყოფილდა

```
if (3  > 10)
alert( 'შედეგი 1' )  // ეკრანზე არ  გამოისახება შედეგი 1
alert( 'შედეგი 2' )  // ეკრანზე გამოისახება მხოლოდ შედეგი 2
```

დასკვნა: თუ რამდენიმე ხაზიანი შედეგის მიღება გვესაჭიროება if პირობის შესრულების დროს ფიგურული { ... } ფრჩხილების გარეშე არსებული შედეგი არ მიიღება.

ზოდად რეკომენდაციის სახით: უმჯობესია ყოველთვის გამოიყენოთ ფიგურული ფრჩხილები. ის უფრო ნათელს ხდის კოდის ბლოკის დასაწყისსა და დასასრულს.

if პირობის დასაკმაყოფილებლად საჭიროა პირობაში იყოს შედეგი - true. პირობაში განთავსებული მნიშვნელობიდან გარდა 0, null, undefined, '', NaN, false დანარჩენი ყველა მნიშვნელობა ითვლება ჭეშმარიტად.

```
if (0) { // კონსტრუქცია არ იმუშავებს  }
if (null) { // კონსტრუქცია არ იმუშავებს  }
if (undefined) { // კონსტრუქცია არ იმუშავებს  }
if ('') { // კონსტრუქცია არ იმუშავებს  }
if (NaN) { // კონსტრუქცია არ იმუშავებს  }
if (false) { // კონსტრუქცია არ იმუშავებს  }
```

```
//  გარდა ზემოთ ჩამოთვლილი შემთხვევებისა სხვა ყველა შემთხვევაში პირობა აღიქმება დადებითად

if (-1) { // კონსტრუქცია იმუშავებს  }
```

დაიმახსოვრეთ Javascript_ში if კონსტრუქციის პირობის დაკმაყოფილების შედეგად კოდის კითხვა გრძელდება ბლოკის ფრჩხილებში და მხოლოდ ამის შემდგომ გადადის მის ქვემოთ არსებულ სკრიტების შესრულებაზე.

მაგ.: ამოცანა მდგომარეობს შემდეგში. შემოვიტანოთ ცვლადი რომელშიც განვსაზღვრავთ პიროვნების ასაკს. მოვახდინოთ if ოპერატორის მეშვეობით დადგენა სრულწლოვანია თუ არასრულწლოვანია არსებული პიროვნება. პირობა დაისმის შემდეგნაირად: თუ ადამიანის ასაკი აღემატები 18 წელს ის წარმოადგენს სრულწლოვანს.

```
<!DOCTYPE HTML>
<html>
<head>
   <meta charset="utf-8">
</head>
<body>
<script>
            var  age =  25;
            var  age1 =  10;

            var  user = 'სრულწლოვანი'; // საწყის ეტაპზე განისაზღვრა, რომ ეს მომხმარებელი არის  18_ზე მეტი წლის.

            if (age < 18) {
                        user = 'არა სრულწლოვანი';
             }

            // თუ if პირობა შესრულდებოდა user ცვლადი მიიღებდა მნიშვნელობად 'არა სრულწლოვანი' და შემდეგ ხაზზე დაიბეჭდებოდა სწორედაც ეს სიტყვა, გამომდინარე იქიდან, რომ   25 ნაკლები არ არის 18_ზე  if  კონსტრუქცია აღარ გააქტიურდა და ცვლად user _ მა შეინარჩუნა პირველადი მნიშვნელობა

            document.write(user); // სრულწლოვანი


            if (age1 < 18) {
                        user = 'არა სრულწლოვანი';
             }
```

```
        // არსებულ შემთხვევაში 10 ნაკლებია 18ზე პირობა დაკმაყოფილებულია. user ცვლადმა მიიღო მნიშვნელობა 'არა სრულწლოვანი'

        document.write(user);          // შედეგია:  არა  სრულწლოვანი

    </script>
</body>
</html>
```

გარდა ცალკეული if კონსტრუქციისა არსებობს **if-else** ოპერატორი და მისი ჩანაწერი შემდეგია

```
        if (age1 < 18) {
                // შესრულდება ყოველთვის, როცა დაკმაყოფილდება if პირობა
        } else {
                // შესრულდება ყოველთვის, როცა არ დაკმაყოფილდება if პირობა
        }
```

თუ ყურადღებით დაუკვირდებით else პირობის ფრჩხილები (...) არ გააჩნია, რადგანაც ის if_ის ურყოფას წარმოადგენს და დამატებით პირობებს არ საჭიროებს და if ოპერატორში დასმულ პირობის „მცდარობის" შემთხვევაში ყოველთვის შესრულდება.

დაიმახსოვრეთ: else ოპერატორს დამოუკიდებლად არსებობა არ შეუძლია, განსხვავებით if ოპერატორისაგან.

```
    if (age1 < 18) {
            document.write('არა სრულწლოვანი');
    } else {
            document.write('სრულწლოვანი');
    }
```

არსებობს if-else კონსტრუქციის შემოკლებული ჩანაწერი, რომელსაც დეველოპერები საკმაოდ აქტიურად გამოიყენებენ მცირე კოდის შემთხვევაში

```
    (age1 < 18) ?  document.write('არა სრულწლოვანი') : document.write('სრულწლოვანი');
```

სადაც სიტყვიერი if არ იწერება, იხსნება პირობის ფრჩხილები, მას მოსდევს „? „ რომელიც აღნიშნავს if_ის ბლოკს, ხოლო ":" else _ს ბლოკია .

არსებობს ასევე შედარებით რთული კონსტრუქცია, რომელიც **else-if**_ად მოიხსენება. ეს არის კონსტრუქცია, რომელშიც if პირობის უარყოფასთან ერთად ხდება ახალი პირობის წამოყენება.

```
if (age < 18) {
        document.write('მოზარდი');
} else if (age < 60) {
        document.write('ზრდასრული');
}
```

მოცემულ მაგალითი შემდეგნაირად იშიფრება: თუ ცვლადი age ნაკლებია 18_ზე მაშინ დაბეჭდოს 'მოზარდი', **წინააღმდეგ შემთხვევაში თუ** age ნაკლებია 60_ზე დაბეჭდავს 'ზრდასრული'.

ბევრს შეიძლება გაუჩნდეს კითხვა, თუ age = 15 მაშინ შესრულდება if ბლოკში არსებული კოდი ( დაიბეჭდება 'მოზარდი') , მაგრამ 15 ხომ 60_ზეც ნაკლებია? რა თქმა უნდა ნაკლებია მაგრამ else if პირობაზე გადასვლა მხოლოდ იმ შემთხვევაში ხდება, როდესაც არ სრულდება if პირობა.

თუ if პირობა შესრულდა არა აქვს მნიშვნელობა რამდენი else-if იარსებებს მის ქვემოთ ისინი არასდროს დაკმაყოფილდებიან. იხ. მოცემული მაგალითი

```
var age = 15;

    if (age < 18) {
        // 15<18 შესაბამისად შესრულდება მხოლოდ ეს კოდი
         document.write('მოზარდი');
    } else if (age < 30) {
        document.write('ზრდასრული');
    }
  else if (age < 60) {
        document.write('ასაკოვანი');
    }
  else if (age > 60) {
        document.write('ხნიერი');
    }
```

შესაძლებელია else-if კონსტრუქციის გამოყენებისას ის დასრულდეს else ოპერატორით, რომელიც ყველა შესაძლო else if პირობის უარყოფას წარმოადგენდეს

```
var car = 'acura';
```

```
    if (car == 'bmw') {
            document.write('bmw');
    }else if (car == 'volvo') {
            document.write('volvo');
    }else if (car == 'opel') {
            document.write('opel');
    }else {
            document.write('მარკა სრულიად განსხვავებულია ის არის '+ car);
    }
```

**switch – case კონსტრუქცია**

ბოლო მაგალითზე დაკვირვებით სია, რომელთანაც არსებულ car _ს ვადარებთ შესაძლოა საკმაოდ დიდი იყოს. შემთხვევა, როცა ერთი კონკრეტული მნიშვნელობის რამდენიმესთან ერთდროულ შედარებაა საჭირო გამოიყენება **switch – case** კონსტრუქცია რომელიც ბევრად მოხერხებულია, გამომდინარე მისი სინტაქსიდან.

```
 switch(შესადარებელი მნიშვნელობა) {

   case 'მნიშვნელობა 1':
            შესასრულებელი არეალი .....
   break;                  // თუ case არსებული მნიშვნელობის თანხვედრა მოხდა break აჩერებს დანარჩენი
დადარების ოპერაციებს.

   case 'მნიშვნელობა 2':
            შესასრულებელი არეალი .....
   break;

   default:
            შესასრულებელი არეალი .....
   break;
}
```

```
 var car = 'acura';

 switch(car) {
   case 'bmw':  // if (x === ' bmw')
            document.write('bmw');
   break;

   case 'volvo':  // if (x === 'volvo')
            document.write('volvo');
   break;
```

```
  case 'opel':  // if (x === 'opel')
          document.write('opel');
  break;

  default:
          document.write('მარკა სრულიად განსხვავებულია ის არის '+ car);
  break;
}
```

ოპერატორი break წყვეტს კოდის კითხვადობას თუ მოხდა გადმოცემული მნიშვნელობის და case_ის გატოლება (ექვივალენტურად). არსებული ოპერატორის გამოტოვების შემთხვევაში გამოვა switch_ში არსებული უკანასკნელი მნიშვნელობა, რაც არასასურველ შედეგამდე მიგიყვანთ.

მინიშნება არ დაგავიწყდეთ პირობის შემდეგ break ოპერატორის გამოყენება.

რაც შეეხება ოპერატორ default მის „ტანში“ გამოტანილი მოქმედება სრულდება მაშინ, როცა არ მოხდება ცვლადის არცერთ case მნიშვნელობასთან გატოლება (ექვივალენტურად).

ფაქტობრივად switch-case კონსტრუქცია, რაღაც კუთხით else-if კონსტრუქციის გამარტივებულ ვარიანტს წარმოადგენს (ყველა შემთხვევაში არა), შესაბამისად თუ მას else-if _თან გავაიგივებთ ოპერატორი default წარმოადგენს ყველა პირობის უარმყოფ else ოპერატორს, რომელიც სრულდება დანარჩენი პირობის დაუკმაყოფილებლობის შემთხვევაში.

default გამოყენება არ არის სავალდებულო.

switch-case კონსტრუქციაში შესაძლებელია რამდენიმე case მნიშვნელობის დაჯგუფება შემდეგი სახით

```
var a = 2+2;

switch (a) {
 case 4:
  alert('სწორია!');
 break;

 case 3:                  // თუ a === 3 ან a === 5 გამოვიდოდა მიმდინარე შეტყობინებები
 case 5:
  alert('შეცდომაა!');
  alert('დაგაკლდათ მცირეოდენი');
 break;

 default:
```

```
  alert('თქვენ საერთოდ არ ხართ არითმეტიკულ მოქმედებებთან ახლოს ☺ ');
 break;
}
```

## ციკლები

სკრიპტის შემუშავების პროცესში ხშირად დგება ამოცანა ერთგვაროვანი მოქმედება განმეორდეს რაიმე ინტენსივობით გარკვეულის ლოგიკის შესაბამისად. კოდის ნაწილის გამეორებადობისათვის Javascript პროგრამირების ენა იყენებს ციკლებს. ადვილად აღქმადი, რომ გამოვიდეს კიდევ ერთხელ განვმარტოთ: ციკლი წარმოადგენს პროცესს, რომელიც რაღაც კანონზომიერებით მეორდება.

მაგ. 1: წელიწადში 4 დროა: გაზაფხული, ზაფხული, შემოდგომა, ზამთარი. ისინი სწორედ იმ თანმიმდევრობით მონაცვლეობენ, როგორც ჩამოვთვალეთ ანუ ემორჩილებიან კონკრეტულ კანონზომიერებებს. ეს არის ციკლი, რომელიც არასდროს (ყოველ შემთხვევაში იმედია) დასრულდება.

მაგ. 2: დღე-ღამე შედგება 24 საათისაგან. სწორედ 24 საათიანი ინტერვალის გავლის შემდგომ იწყება ახალი დღე ანუ საათის ათვლა უბრუნდება ნიშნულ 0_ს.

ამ ყოველივეს გასამარტივებლად Javascript_ში გამოიყენება ციკლის ოპერატორები **while, do while, for.** განვიხილოთ თანმიმდევრულად.

ციკლი while სინტაქსი მოცემულია მაგალითზე.

```
while (პირობა) {
        მოქმედების შესასრულებელი არეალი
}
```

როგორც მონახაზიდან ჩანს ციკლი while ვიზუალი ძალიან ახლოსაა if კონსტრუქციასთან. მასაც გააჩნია პირობის ჩასაწერი ფრჩხილები ( ... ) და იდენტური ბლოკის გამხსენი და დამხურავი ფიგურული ფრჩხილები { ... } . შესაბამისად არსიც იგივეა თუ პირობა სრულდება ანუ პასუხი ჭეშმარიტია true, მაშინ სრულდება ბლოკის ფრჩხილებში მოთავსებული სკრიპტი და ეს პროცესი გრძელდება იქამდე სანამ პირობის **ჭეშმარიტება** ძალაშია. მაგ.:

```
var a = 5;

while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');
}
```

მოცემული ციკლი იმუშვებს უსასრულოდ. ტექსტი 'გამარჯობა მე ვარ ციკლი' alert დიალოგური ფანჯრის მეშვეობით გამოტანება უწყვეტ რეჟიმში და თითოეულის გათიშვა გამოიწვევს საპასუხოდ ახლის გამოჩენს. **რატომ ?** გამომდინარე იქიდან, რომ 5 ყოველთვის ნაკლებია 9_ზე ეს პირობა არასდროს შეიცვლება. ვთქვათ ამ პროცესის გამეორება გვესაჭიროება მხოლოდ 4_ჯერ. როგორ მოვიქცეთ? შევეცადოთ დავხვეწოთ კოდი და მოვარგოთ ჩვენს მოთხოვნილებებს.

```
var a = 5;

while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');
        a++;                                    // გამოვიყენე ინკრიმენტი
}
```

a ცვლადის ზრდადობა გამოიწვევს იმას, რომ როდესღაც a მიაღწევს მნიშვნელობას როცა ის აღარ იქნება ნაკლები 9_ზე და ციკლი შეწყვეტს მუშაობას. არსებულ შემთხვევაში ეს იქნება მეხუთე ნაბიჯზე, შესაბამისად შეტყობინება ეკრანზე გამოვა 4_ჯერ.

```
var a = 5;

// ეტაპი I
// a = 5 პირობა სრულდება
while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');       // გამოდის ეკრანზე
        a++;                                    // a ცვლადმა განიცადა ინკრიმენტაცია - შემდეგი ხაზიდან
a=6
// აქ უკვე a არის 6 – a=6
}


// ციკლის უზრუნველყოფს ინფორმაციის განმეორებადობას ე.ი ის აბრუნებს კოდის მნიშვნელობას
დასაწყისში და ეკითხება არის a < 9_ზე? გამომდინარე იქიდან, რომ a =6 ის აშკარად ნაკლები გამოდის
9_ზე და პროცესი კიდევ ერთხელ მეორდება

// ეტაპი II
// a = 6 პირობა სრულდება
while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');       // გამოდის ეკრანზე
        a++;                                    // a ცვლადმა განიცადა ინკრიმენტაცია - შემდეგი ხაზიდან
a=7
```

```
// აქ უკვე a არის 7 – a=7
}


// ეტაპი III
// a = 7 პირობა სრულდება
while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');      // გამოდის ეკრანზე
        a++;                                   // a ცვლადმა განიცადა ინკრიმენტაცია - შემდეგი ხაზიდან
a=8
// აქ უკვე a არის 8 – a=8
}


// ეტაპი IV

// a = 8 პირობა სრულდება
while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');      // გამოდის ეკრანზე
        a++;                                   // a ცვლადმა განიცადა ინკრიმენტაცია - შემდეგი ხაზიდან
a=9
// აქ უკვე a არის 9 – a=9
}


// ეტაპი V
// a = 9 პირობა არ სრულდება. 9 არ არის ნაკლები 9_ზე
// შესაბამისად სკრიპტი აღარ კითხულობს ბლოკის შიგთავს და ეკრანზე  alert-აღარ გამოვა.
while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');
        a++;
}
```

ციკლის გამეორებადობის პროცესს **იტერაცია** ეწოდება. მაგალითის შემთხვევაში ციკლი ასრულებს 4 - ოთხ იტერაციას.

უსასრულო ციკლის საჭიროების შემთხვევაში არ არის აუცილებელი ვრცელი გზის გამოყენება

```
var a = 5;

while (a < 9) {
        alert('გამარჯობა მე ვარ ციკლი');
```

```
}
```

შესაძლებელია პირობაში გაიწეროს true

```
while (true) {
        alert('გამარჯობა მე ვარ ციკლი');
}
```

**დავალება 1**:

გამოვსახოთ ეკრანზე 1_დან 10_მდე ყველა ლუწი რიცხვი შემდეგი სახით. ამ ეტაპზე დავუშვათ , რომ ბოლო მნიშვნელობასაც ექნება გამყოფად ტირე „-“

```
2 – 4 – 6 – 8 -
```

დავალების ამოსახსნელად გვჭირდება ციკლი, ინფორმაციის გამოტანის ოპერატორი, და ცვლადი რომელიც ყველა ჯერზე მოიმატებს 2_ით.

```
var a = 2;

while (a < 10) {
        document.write( a + ' - ' );  // დაბეჭდავს 2 - ..... და აშ ყველა ნაბიჯზე
        a += 2;     // იგივეა რაც  a = a + 2  (მინიჭების ოპერატორის შემოკლებული  ჩანაწერი )
}
```

**დავალება 2**:

ციკლის მეშვეობით გამოვიტანოთ ეკრანზე ყველა რიცხვის ჯამი 1_დან 5_მდე.

```
15
```

დავალების ამოსახსნელად გვჭირდება ციკლი, ინფორმაციის გამოტანის ოპერატორი, და ცვლადი, რომელიც ყველა ჯერზე მოხდება არსებული რიცხვის მიმატება წინა ჯამთან.

```
var i = 1;
var sum = 0;


while (i <= 5) {
        sum += i;  //sum = sum+i I ეტაპი 0+1, I I ეტაპი 1+2,  I I I ეტაპი 3+3,  IV ეტაპი 6+4, V ეტაპი 10+5
        i++;                //უზრუნველყოფს ციკლის მუშაობას
}
```

```
document.write(sum);          // 15
```

განასხვავებით დავალება1_ის ამოხსნისაგან სადაც `document.write()` მეთოდი უშუალოდ while ციკლის ტანში { ... } იყო გამოძახებული, დავალება2 შედეგის გამოტანის მეთოდი გარეთაა გამოძახებული. ეს რა თქმა უნდა დავალებიდან გამომდინარეა. პირველ შემთხვევაში საჭირო იყო ყველა ეტაპზე შედეგის გამოტანა, ხოლო მეორე შემთხვევაში მხოლოდ საბოლოო შედეგის. მეორე დავალების ციკლი უბრალოდ უზრუნველყოფდა დაჯამებას თითოეულ ეტაპზე (ეტაპები კომენტარის ველშია გაშლილი) და გამოტანა ერთჯერადი პროცედურით შემოიფარგლებოდა.

ციკლში შესაძლებელია break და continue ოპერატორების გამოყენება. break_ის შესახებ ინფორმაცია მოწოდებული იყო switch-case კონსტრუქციის განხილვისას. ის გამოიყენებოდა პროცესის შესაჩერებლად / გასაწყვეტად. იგივე ფუნქციონალი აკისრია მას ციკლთან მიმართებაშიც.

break ციკლში მოქმედების უკეთ გასაგებად დავალება 2 შევასრულოთ სახეცვლილი პირობით

**დავალება 3:**

ეკრანზე დაბეჭდოს რიცხვთა ჯამი 1_დან 5_მდე, სანამ ის იქნება 12_ზე ნაკლები. ანუ ეკრანზე უნდა გამოიტანოს 10 ( I ეტაპი 0+1, I I ეტაპი 1+2, I I I ეტაპი 3+3, IV ეტაპი 6+4, V ეტაპი 10+5 ), რადგანაც ბოლო ეტაპზე მიიღება რიცხვი 15, რომელიც შესაბამისად აღემატება 12_ს.

```
var i = 1;
var sum = 0;

while (i <= 5) {

// არსებულ ეტაპზე ვამოწმებთ ჯამს დამატებული მიმდინარე რიხვი მეტი ხომ არ არის 12, თუ პირობა შესრულდა sum ცვლადში აღარ მოხდეს მიმდინარე ბიჯის დამატება და ციკლმა შეწყვიტოს მუშაობა;

        if ( (sum + i) > 12) {
                break;
        }
        sum += i;  //sum = sum+i I ეტაპი 0+1, I I ეტაპი 1+2, I I I ეტაპი 3+3, IV ეტაპი 6+4
        i++;                //უზრუნველყოფს ციკლის მუშაობას
}

document.write(sum);          // 10
```

break ოპერატორისაგან განსხვავებით continue არ წყვეტს ციკლის მუშაობას ის უბრალოდ ახტება იმ ნაბიჯს, რომელზეც გვაქვს შესაბამისი პირობა დადებული.

**დავალება 4:**

გამოვსახოთ ეკრანზე 1_დან 20_მდე ყველა ლუწი რიცხვი,გარდა **6** და **14** შემდეგი სახით

2 – 4 –8 – 10 – 12 – 16 – 18

```
var a = 2;

while (a < 20) {

// პირობა: თუ a მნიშვნელობა ტოლია 6 ან 14 მოახდინოს a გაზრდა 2_ით და დაბრუნდეს ციკლის პირობაში, შესაბამისად ტოვებს document.write()_ს რომელიც უზრუნველყოფს დაბეჭდვას

        if (a==6 || a==14 ) {
                a += 2;
                continue;
        }

        document.write( a + ' - ' );  // დაბეჭდავს 2 - ..... და აშ ყველა ნაბიჯზე
        a += 2;     // იგივეა რაც  a = a + 2  (მინიჭების ოპერატორის შემოკლებული  ჩანაწერი )
}
```

როგორც ქვეთავის დასაწყისში აღვნიშნეთ Javascript_ში გვხვდება do…while ციკლიც. მისი მუშაობის პრინციპი ოდნავ განსხვავებულია სტანდარტული while ციკლისაგან, ასევე მცირეოდენ შეცვლილია წერის სინტაქსიც, რამეთუ ჯერ მოდის შესასრულებელი მოქმედების ბლოკი ხოლო შემდგომ პირობა. ეს ყოველივე კი უზრუნველყოფს იმას, რომ მოქმედება მინიმუმ **ერთჯერადად** მაინც სრულდება.

```
var i = 2;

do {

   document.write( i + ' - ' );
   i++;

} while (i > 5)

// ეკრანზე მიღებული შედეგი 2 -
```

ციკლი ასრულებს შემდეგ მოქმედებას :

- do_ს მეშვეობით შედის ციკლის ტანში {...} ასრულებს ლოგიკურ ოპერაციას ( 2- );
- ახდენს **i** ცვლადის გაზრდას 1_ით;
- ამის შემდგომ უყურებს while ციკლის პირობას, გამომდინარე იქიდან, რომ პირობა არ კმაყოფილდება ( 3>5 ), ხელახალი პროცესის გამეორებადობა არ მოხერხდა;

სტანდარტული while ციკლის გამოყენების შემთხვევაში შედეგი საერთოდ არ გამოისახებოდა ეკრანზე, რადგანაც იქ ჯერ შედარების ოპერაცია ხორციელდება

```
var i = 2;

while (i > 5) {

   document.write( i + ' - ' );
   i++;

}

// ეკრანზე არავითარი შედეგი არ გამოისახება
```

მაგრამ მოცემულ მაგალითს თუ შემდეგ პირობას წავუყენებთ i < 5 , არავითარი განსხვავება არ იქნება, რომელი ციკლის მეშვეობით შევასრულებთ მას იქნება ეს while თუ do…while ციკლები

```
// მაგალითი while  - ციკლით

var i = 2;

while (i < 5) {

   document.write( i + ' - ' );     // ეკრანზე მიღებული შედეგი 2 – 3 -  4 –

   i++;

}


// მაგალითი do…while  - ციკლით

var i = 2;
```

```
do {

   document.write( i + ' - ' );    // ეკრანზე მიღებული შედეგი 2 – 3 -  4 -

   i++;

} while (i < 5)
```

პროგრამისტების მიერ ციკლებიდან ყველაზე ხშირად - for კონსტრუქციაა, რომელიც თავისი სინტაქსით მეტად მობილურია და შემდეგნაირად გამოიყურება. ის ფაქტობრივად იყენებს შაბლონს, რომელშიც თავისთავად მოიაზრება ციკლი while.

```
for ( დასაწყისი; პირობა; ბიჯები) {
  // ... ციკლის ტანი ...
}
```

while ციკლისაგან განსხვავებით for_ის ფრჩხილებში ( ... ) კომპაქტურადაა მოთავსებული ცვლადის გამოცხადება, პირობა და  იტერაცია რაც თვალნათელს ხდის თითოეულ ბიჯს.

```
for (var i = 0; i < 3; i++) {
  alert( i );                          // alert(0),  alert(1),  alert(2)
}
```

როგორც while და do…while შემთხვევაში იყო შესაძლებელი იდენტურად მათი ამუშავებისა ასევე შესაძლებელია  for ციკლი ვამუშაოთ while_ის მსგავსად. for ციკლში დასაშვებია ნებისმიერი მნიშვნელობის გამოტოვება.

```
// ვარიანტი I

var i = 0
for ( ; i < 3; i++) {
  // ... ციკლის ტანი ...
}


// ვარიანტი II
var i = 0
for ( ; i < 3;) {
  // ... ციკლის ტანი ...
```

```
i++;
}

// ვარიანტი III
for ( ;;) {
  // ... ციკლის ტანი ...
}
```

**ვარიანტი I** - ფაქტობრივად არ განსხვავდება for სტანდარტული ჩანაწერისაგან უბრალოდ საწყისი მნიშვნელობა ამოღებულია ფრჩხილებიდან (...) და გატანილია ცალკე.

**ვარიანტი II** - ანალოგიურია while ციკლისა

**ვარიანტი III** - ციკლი, რომელიც იმუშავებს უსასრულოდ.

Javascript_ში არსებობს for in კონსტრუქცია ამ ეტაპზე მისი განხილვა აზრს მოკლებულია, დამხმარე თემებზე, რომელთანაც ის კავშირშია ჯერ არ გქონიათ შეხება, შესაბამისად შემდეგ ქვეთავებში მაგალითებზე დაყრდნობით გავარჩევთ მასაც დეტალურად.

## 8.3. მასივებთან მუშაობა

**მიმდინარე პარაგრაფის თემატიკა**

- მასივებთან მუშაობის ძირითად ფუნქციები
- ამოცანის გადაჭრის გზები მასივზე დაფუძნებული აბსტრაქტული მონაცემთა სტრუქტურების მეშვეობით
- ამოცანის იმპლემენტაცია მასივების მეშვეობით

მასივი ეს არის მონაცემთა ერთგვარი ტიპი, რომელიც შეიცავს ერთ ან რამდენიმე მნიშვნელობას ერთდროულად, რომლებიც მასივში განთავსებულნი არიან საკუთარ პოზიციებზე. არსებულ პოზიციებს ინდექსები ეწოდებათ და ისინი წარმოადგენენ რიცხვით მნიშვნელობას 0_დან დადებითი მიმართულებით. გამომდინარე Javascript_ის არა მკაცრი ტიპიზაციიდან მასივს მკვეთრად გამოხატული

ტიპიზება არ სჭირდება, ასევე მისი თითოეული მნიშვნელობა შეიძლება ერთმანეთისაგან რადიკალურად განსხვავებული ტიპის გახლდეთ. არის შესაძლებლობა მასივის მნიშვნელობა ასევე მასივი იყოს.

მასივის შესაქმნელად ერთ ერთი გზა შემდეგი. საჭიროა ახალი ცვლადის გამოცხადება მისთვის სახელის მინიჭება და კვადრატული ფრჩხილები [....]. მაგალითზე მოცემულია ცარიელი მასივი.

```
var arr = [];
```

მასივის ელემენტების შესავსებად საკმარისია კვადრატულ ფრჩხილებში მძიმით მოხდეს ჩამონათვალის გაკეთება

```
var arr = ['მარწყვი', 'ატამი', 'ფორთოხალი'];
```

არსებული მასივი შედგება 3 ელემენტისაგან, შესაბამისად ისინი განთავსებულნი არიან პოზიციებზე რომელიც ათვლას ყოველთვის 0 დან იწყებს. თითოეულ მათგანთან წვდომისათვის უნდა მიმართოთ მასივს სახელწოდებით , კვადრატული ფრჩხილები და იმ კონკრეტული მნიშვნელობის ინდექსით რომელის გამოტანაც გსურთ.

```
var arr = ['მარწყვი', 'ატამი', 'ფორთოხალი'];

alert( arr[ 0 ] );              // მარწყვი
alert( arr[ 1 ] );              // ატამი
alert( arr[ 2 ] );              // ფორთოხალი
```

იმ ინდექსით მასივისათვის მიმართვა რომელზეც არანაირი მნიშვნელობა არ არის განთავსებული შედეგად დაგიბრუნებთ undefined

```
var arr = ['მარწყვი', 'ატამი', 'ფორთოხალი'];

alert( arr[ 3 ] );              // undefined
alert( arr[ 80 ] );             // undefined
```

თუ მასივის ელემენტების ჩამოთვლისას მათ შორის ხელოვნურად ან შემთხვევით ჩავსვამთ დამატებით მძიმეებს ისინი გამოყოფს მასივის ელემენტისათვის ინდექსებს და მათ მნიშვნელობად undefined მოიაზრებს. მაგალითზე მოყვანილი მასივის მნიშვნელობა 'ატამი'_თან წვდომისათვის ინდექსი 1 აღარ გამოდგება, რადგანაც არსებულ ინდექსზე მასივმა განალაგა მნიშვნელობა რომელიც ამ ეტაპზე undefined არის. მაგალითზე ხელოვნურად არის ადგილები გამოტოვებული ინდექსების ზუსტად აღსაქმელად.

```
var arr = ['მარწყვი',   ,      ,'ატამი',      ,'ფორთოხალი'];
//          0   1  2   3   4    5

alert( arr[ 1 ] );                  // undefined
alert( arr[ 3 ] );                  // ატამი
```

მასივისათვის ახალი ელემენტის მინიჭება ან არსებულის ჩანაცვლება ყოველთვის შესაძლებელია, საჭიროა მხოლოდ მასივის კონკრეტულ ინდექსს მივანიჭოთ მნიშვნელობა. იმ შემთხვევაში თუ ამ კონკრეტულ ინდექსზე  უკვე არსებობს მნიშვნელობა ის ჩანაცვლდება წინააღმდეგ შემთხვევაში დაემატება ახალი.

```
var arr = ['მარწყვი' , 'ატამი','ფორთოხალი'];
//          0       1       2

arr[ 0 ] = 'ვაშლი';   // arr = ['ვაშლი' , 'ატამი','ფორთოხალი'];
arr[ 3 ] = 'ყურძენი';             // 'ვაშლი' , 'ატამი','ფორთოხალი', 'ყურძენი'];
```

როგორც ქვეთავის დასაწყისში აღვნიშნეთ აუცილებელი არ არის მასივი შეიცავდეს ერთგვაროვანი ტიპის ელემენტებს. Javascript_ში შესაძლებელია შერეული ტიპის მნიშვნელობების მინიჭება ერთი კონკრეტული მასივისათვის. აქამდე განხილულ მაგალითებში საქმე მხოლოდ ტექსტური (string) ტიპის მნიშვნელობები ფიგურირებდნენ. გასათვალისწინებელია ის რომ მასივის ინდექსზე მნიშვნელობის განთავსებისას ტიპი სტანდარტული ჩანაწერით (იქნება ის ბრჭყალებით - string, მის გარეშე-number, იქნება ობიექტი და ა.შ) გამოისახება.

```
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];
```

მასივის ინდექსაცია ყოველთვის იწყება 0_დან. ხელოვნურად მისი ცვლილება შეუძლებელია. მაგ.: ინდექსად 5_ის მნიშვნელობის განსაზღვრა არ განაპირობებს მის ამთვლელ ინექსს. საწყისი ინდექსები ავტომატურად შეივსება undefined მნიშვნელობებად.

```
arr[ 5 ] = 'საქართველო';       // arr = [ , , , , , 'საქართველო'];
```

როგორც აღვნიშნეთ მასივის კონკრეტული მნიშვნელობის გამოსატანად სჭირო მასივის სახელი და მისი ინდექსის გამოტანის ოპერატის ატრიბუტად განსაზღვრა ( მაგ.: document.write(**arr[5]**) ), მაგრამ Javascript_ში შესაძლებელია მასივის სრული ვიზუალიზაცია უშუალოდ მისი სახელის გამოტანით.

დაიმახსოვრეთ არსებული გზა ემსახურება მხოლოდ მასივის ვიზუალური შედეგის მიღებას და ძირითადად სატესტოდ გამოიყენება.

```
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];

document.write(arr) // 'მარწყვი' , true ,  null, 'კივი', 155;
```

**არსებობს მასივის შექმნის კიდევ ერთი სახის სინტაქსი new Array(); ის შედარებით იშვიათად** გამოიყენება გამომდინარე იქიდან, რომ კვადრატული ფრჩხილებით [...] ჩანაწერი ბევრად შემოკლებულია.

```
var arr = new Array('მარწყვი' , true ,  null, 'კივი', 155);
```

ასევე არსებობს ერთი თავისებურება და სხვაობა ამ ორ სინტაქსს შორის. თუ მასივის შედგება რიცხვითი მნიშვნელობისაგან და ის მხოლოდ 1 ცალია, **new Array()** მეშვეობით მისი გამოცხადება მოგცემთ განსხვავებულ შედეგს. მაგ.:

```
var arr = [155];
alert( arr[ 0 ] );                          // 155;

var arr1 = new Array( 155);
alert( arr1[ 0 ] );                 // undefined
```

მაშინ რა მისია აკისრია ამ შემთხვევაში 155 - new Array( 155 ) ? **ის განსაზღვრავს მასივის სიგრძეს.** (არსებულ საკითხს დავუბრუნდებით) სხვა ყველა დანარჩენ შემთხვევაში ეს 2 ჩანაწერი მუშაობს იდენტურად იქნება ეს სხვა ტიპის 1 ან 1_ზე მეტი მნიშვნელობა, თუ რიცხვითი მნიშვნელობა მხოლოდ ერთ ელემენტზე მეტი.

```
// შედეგი იდენტური
var arr = new Array('მარწყვი' , true ,  null, 'კივი', 155);
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];

// შედეგი იდენტური
var arr = ['მარწყვი'];
var arr = new Array('მარწყვი');
```

```
// შედეგი იდენტური
var arr = [5 , 120];
var arr = new Array(5 , 120);

// შედეგი განსხვავებული
var arr = [10];
var arr = new Array( 10  );
```

რა არის მასივის სიგრძე, რომელმაც მცირეოდენი გაუგებრობა წარმოშვა?

მასივის სიგრძე ეს არის მასივის თვისება რომელიც განსაზღვრავს თუ რამდენი ელემენტისაგან შედგება ესა თუ ის კონკრეტული მასივი. მისი სინტაქსი შემდეგია **მასივის სახელი.length**

```
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];

alert( arr.length );   // 5
```

მასივის სიგრძე დინამიურ თვისებას წარმოადგენს და იცვლება მასივის ელემენტების რაოდენობიდან გამომდინარე- უფრო ზუსტად, რომ აღვნიშნოთ მასივის ინდექსების რაოდენობიდან გამომდინარე. შემთხვევას როცა კონკრეტულ ინდექსზე (გარდა 0 ) ვათავსებთ ერთ ელემენტს მასივის სიგრძეს განსაზღვრავს ინდექსი და არა ელემენტი, რადგანაც ინდექსზე განთავსება ნიშნავს იქამდე არსებული გამოტოვებული ინდექსების ავტომატურ შევსებას undefined მნიშვნელობებით.

`.length` თვისების გამოყენებით შესაძლებელია არსებული მასივის დამოკლება, რომელიც არც ისე ხშირად გამოყენებადი ამ დანიშნულებით. მაგ.:

```
var arr = [1, 2, 3, 4, 5];

arr.length = 2; // შემცირდა 2 ელემენტრამდე
alert( arr ); // [1, 2]

arr.length = 5; // შევეცადოთ  დავაბრუნოთ მასივის საწყისი სიგრძე
alert( arr[3] ); // undefined: სამწუხაროდ მცდელობამ უშედეგოდ ჩაიარა
```

მასივის ელემენტების უწყვეტ რეჟიმში ამოსაღებად შესაძლებელია უკვე ცნობილი რომელიმე ციკლის ოპერატორის გამოყენება, რომელიც შეასრულებს იმდენ იტერაციას რამდენიც იქნება  მასივის სიგრძის (`arr.length`) .ეს ყოველივე უზრუნველყოფს ყველა ელემენტთან წვდომას. მაგ.:

```
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];

for (var i = 0; i <  arr.length; i++) {
  alert( arr[i] );                              // alert('მარწყვი'),  alert(true),  ....
}
```

ციკლების განხილვისას აღვნიშნეთ, რომ არსებობს for_ის სახეცვლილი ჩანაწერი რომელსაც იმ კონკრეტულ ეტაპზე არ შევეხეთ. ეს არის for…in კონსტრუქცია, რომელიც გამოიყენება ობიექტის ყველა მნიშვნელობის გადასასვლელად. არსებულ შემთხვევაში ზედა მაგალითის პირობის გასამარტივებლად ეს კონსტრუქცია იდეალური ვარიანტი იქნებოდა, სადაც key მნიშვნელობა ხოლო arr არსებული მასივი.

```
var arr = ['მარწყვი' , true ,  null, 'კივი', 155];

for (var key in arr) {
  alert( arr[key] ); // 'მარწყვი' , true ,  null, 'კივი', 155;
}
```

გარდა ორგანზომილებიანი მასივებისა Javascript_ში გვხვდება მრავალგანზომილებიანი მასივები. იმ შემთხვევაში თუ მასივის ინდექსებზე მნიშვნელობებად ასევე მასივები მოგვევლინებიან მაშინ საქმე მრავალგანზომილებიან მასივებთან გქნოდეთ. ამ მეთოდის გამოყენებით შესაძლოა მაგ. მატრიცის მიღება. (ვიზუალისათვის წარმოდგენილია სხვადასხვა ხაზზე, ერთ ხაზზე განლაგებაც არ წარმოადგენს შეცდომას)

მის კონკრეტულ ელემენტთან წვდომა შესაძლებელია მთავარი და ქვე მასივის  მასივის ინდექსის მეშვეობით

```
var matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

  alert( matrix [1] [2] );          // 6
```

გარდა length თვისებისა არსებობს რიგი ფუნქციების / მეთოდებისა,  რომლებიც აქტიურ რეჟიმში გამოიყენებიან მასივებთან სამუშაოდ. იხ ცხრილი VI

| ცხრილი VI |
| --- |
| `var arr = ['მარწყვი' , 'ატამი','ხილი'];` |

| მეთოდი | აღწერა | მაგალითი | შედეგი |
|---|---|---|---|
| **toString()** | ტექსტურ მონაცემად გარდაქმნა | `arr.toString()` | მარწყვი , ატამი , ფორთოხალი |
| **join**() | მნიშვნელობების შეერთება | `arr.join( * )` | მარწყვი * ატამი * ფორთოხალი |
| **pop()** | ბოლო ელემენტის ამოჭრა | `arr.pop()` | 'მარწყვი' , 'ატამი' |
| **push()** | ელემენტის/ელემენტების დამატება მასივის ბოლოში | `arr.push('ვაშლი')` | 'მარწყვი' , 'ატამი','ხილი', 'ვაშლი' |
| **shift()** | პირველი ელემენტის ამოჭრა | `arr.shift('კივი')` | 'კივი','მარწყვი' , 'ატამი','ხილი'; |
| **unshift**() | ელემენტის/ელემენტების დამატება მასივის ბოლოში | `arr.unshift()` | 'ატამი','ხილი' |
| **splice()** | შესაძლოა ელემენტების ჩამატება კონკრეტულ ინდექსზე, არსებულის ამოჭრა ან ჩანაცვლება | `arr.splice(0, 2 , 'კივი')` | 'კივი','ხილი' |
| | | `arr.splice(1, 0 , 'კივი')` | 'მარწყვი' , 'კივი', 'ატამი','ხილი' |
| | | `arr.splice(1, 2)` | 'მარწყვი' |
| **slice()** | მასივის ინდექსების კონკრეტული დიაპაზონიდან მნიშვნელობების ამოღება | `var arr2=arr.slice(0,2)` | 'მარწყვი' , 'ატამი' |
| **sort()** | სორტირება / დალაგება | `arr.sort()` | 'ატამი', 'მარწყვი', 'ხილი' |
| reverce() | შეტრიალება | `arr.reverce()` | 'ხილი', 'ატამი','მარწყვი' |
| **concat()** | | `arr2= ['კივი' , 'ბუ'];` | 'მარწყვი' , 'ატამი','ხილი', 'კივი' , 'ბუ' |

| | | | |
|---|---|---|---|
| | რამდენიმე მასივის შეერთება | `arr.concat(arr2)` | |

## 8.4. მზა ფუნქციების გამოყენება

**მიმდინარე პარაგრაფის თემატიკა**

- დროისა და თარიღის ფუნქციები
- სტრიქონული ფუნქციები
- ბრაუზერთან სამუშაო ფუნქციები
- დოკუმენტთან სამუშაო ფუნქციები
- ტაიმერის ფუნქციები

ობიექტი ამომავალი წერტილია პროგრამირებაში. ყველაფერი მის გარშემო კრავს მოქმედ ჯაჭვს. რეალურ რეჟიმში კონკრეტულ ობიექტთან მუშაობის პროცესი საკმაოდ საინტერესო, რამეთუ დიდი აბათობით ობიექტების უმრავლესობა საკმაო ინფორმაციისა და ფუნქციონალის მატარებელია.

ძირითად ობიექტებს, რომლებიც Javascript_ში აქტიურად გამოიყენება საკუთარი მზა თვისებებისა და მეთოდების / ფუნქციების ერთობლიობა გააჩნია, რომლებიც საკმაოდ ამარტივებს ობიექტთან მუშაობას და ხელს უწყობს დეველოპერს მარტივი ფუნქციონალის გამოყენებით ისეთი შედეგის მიღებაში, რომლის თავად შესაქმნელად შესაძლოა არც ისე მცირე დროითი რესურსის დახარჯვა მოუხდეს.

არსებულ ქვეთავში შეძლებისდაგვარად მაქსიმალური რაოდენობის მეთოდების გარჩევა მოხდება. და თუ შემთხვევით მათ გარდა კიდევ სხვა მეთოდებს მიაკვლიეთ არ იფიქროთ მორიგ მსოფლიო დონის აღმოჩენასთან გქონდეთ შეხება. გამომდინარე იქიდან, რომ ფუნქციათა რაოდენობა არც ისე მცირეა მათი სრულყოფილი ახსნა ფაქტობრივად შეუძლებელია.

### **String**_თან მუშაობის მეთოდები

იმედია შეხსენება არ გჭირდებათ, რომ Javascript_ში არსებობს ტექსტუალური ტიპის ინფორმაცია. ერთის შეხედვით რას წარმოადგენს იგი? **String** ეს არის სიმბოლოთა ერთობლიობა.

```
var txt = '21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული'; //ლიტერალი / მნიშვნელობა
```

```
var txt = new String('21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული'); //ობიექტი
```

არსებული შექმნის მეთოდებიდან პირველი მეტად პრიმიტიულია. ამასთანავე Javascript_ში ლიტერარს შეუძლია ობიექტის მეთოდების მიღება და მათთან მუშაობა, საჭიროების შემთხვევაში ინტერპრეტატორი ავტომატურად ახდენს მის გარდაქმნას ობიექტად.

თუ კარგად დაუკვირდებით შეატყობთ, რომ ტექსტი შედგება სიმბოლოებისაგან, ე.ი. ის თავის მხრივ წარმოადგენს ერთგვარ მასივს. შესაბამისად მასივის თვისება **length** არსებულო ობიექტის თვისებასაც წარმოადგენს. ის განსაზღვრავს ტექსტის სიგრძეს.

```
var txt = '21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული'; //ლიტერალი / მნიშვნელობა

alert ( txt.length )  // 55
```

მასივის მსგავსად ტექსტის კონკრეტულ სიმბოლოსთან წვდომა შესაძლებელია მის ინდექსზე მიმართვით

```
var txt = '21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული';

alert ( txt[ 0 ] )  // 2

alert ( txt[ 3 ] )  // ე
```

რა თქმა უნდა გარდა ამ მსგავსებისა მასივთან String_ს გააჩნია მეთოდთა ერთობლიობა რომელიც მოცემულია ცხრილ VII

ცხრილი VII

```
var txt = '21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული';
```

| მეთოდი | აღწერა | მაგალითი | შედეგი |
|---|---|---|---|
| **charAt()** | კონკრეტულ ინდექსზე არსებული მნიშვნელობის გამოტანა, იდენტურია txt[…] ჩანაწერისა | `txt`.charAt(3) | ე |

| | | | |
|---|---|---|---|
| **charCodeAt**() | მისი საშუალებით შესაძლებელია დადგინდეს სიმბოლოს კოდური რიცხვითი მნიშვნელობა ASCII | `txt`.charCodeAt(3) | 4608 |
| **String.fromChar-Code()** | რიცხვითი მნიშვნელობის შესატყვისი სიმბოლოს მიღება | `String`.fromCharCode(97) | a |
| **concat()** | შეერთება ერთი ან რამდენიმე ტექსტური ინფორმაციის | `var txt2 ='ამბობენ, რომ '`<br>`txt`.concat(txt2) | ამბობენ, რომ 21_ე საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული |
| **indexOf()** | ახორციელებს სიმბოლოს ძებნას ტექსტში და გამოაქვს მისი პირველი დამთხვევის ინდექსი, სიმბოლოს ვერ მოძებნის შემთხვევაში პასუხია ( -1 ) | `txt`.indexOf('საუკუნე ') | 5 |
| **lastIndexOf()** | ახორციელებს სიმბოლოს ძებნას ტექსტში და გამოაქვს მისი უკანასკნელი დამთხვევის ინდექსი.(ძიება ბოლოდან) სიმბოლოს ვერ მოძებნის შემთხვევაში პასუხია ( -1 ) | `txt.lastIndexOf('ულ')` | 52 |
| **replace ()** | ტექსტის ჩანაცვლება | `txt.replace('21_ე','XIX')` | XIX საუკუნე ინფორმაციული ტექნოლოგიების ერადაა მიჩნეული |
| **search()** | ფაქტობრივად მუშაობს ისე როგორც მეთოდი **in-dexOf()** | `txt`.**search**('საუკუნე ') | 5 |
| **slice()** | იღებს ტექსტიდან კონკრეტულ მონაკვეთს | `txt`. **slice(5,11)** | საუკუნ |

| | | | |
|---|---|---|---|
| | მითითებულს ფუნქციის არგუმენტებად | | |
| **substring()** | იდენტურია ფუნქციისა slice() | `txt.` **substring (5,11)** | საუკუნ |
| **substr()** | იღებს ტექსტიდან კონკრეტულ მონაკვეთს პირველი არგუმენტი საწყისია მეორე რაოდენობა | `txt.` **substr(5,11)** | საუკუნე ინ |
| **toUpperCase()** | რეგისტრის ცვლილება ზედა ინდექსზე. ქართულ ტექსტზე აზრი არ აქვს გამოყენებას | `var txt2 = 'hello'`<br>`txt2.`**toUpperCase()** | HELLO |
| **toLowercase()** | რეგისტრის ცვლილება ქვედა ინდექსზე. ქართულ ტექსტზე აზრი არ აქვს გამოყენებას | `var txt2 = 'HEllo'`<br>`txt2.` **toLowercase()** | hello |

### ტექსტური მნიშვნელობის გარდაქმნა რიცხვით მნიშვნელობად

არსებობს რიგი ფუნქციების რომლებიც უზრუნველყოფენ ტექსტური ტიპის ინფორმაციის გარდაქმნას რიცხვით ტიპად. რა თქმა უნდა თუ ტექსტური ტიპის ცვლადიდან შესაძლებელია რიცხვითი მნიშვნელობის მიღება, წინააღმდეგ შემთხვევაში მიიღება NaN . არსებული მეთოდებია:

- parseInt()
- parseFloat()
- Number();

**მოკლედ მიმოვიხილოთ ისინი:**

მეთოდი parseInt() უზრუნველყოფს ტექსტიდან **მთელი** რიცხვის გამოყოფას, მხოლოდ იმ შემთხვევაში თუ ტექსტი იწყება რიცხვითი მნიშვნელობით, წინააღმდეგ შემთხვევაში ნებისიმერი სიმბოლო აღიქმება არარიცხვით ტიპად და შედეგი იქნება NaN

```
alert ( parseInt('21_ე საუკუნე'))                    // 21
alert ( parseInt('21.55'))                           // 21
alert ( parseInt('ტექსტით დაწყებული 21.55'))        //NaN
```

მეთოდი parseFloat() იღებს ათწილად მნიშვნელობებს. მაგ.:

```
alert ( parseInt('21_ე საუკუნე'))                    // 21
alert ( parseInt('21.55'))                           // 21.55
```

```
alert ( parseInt('ტექსტით დაწყებული 21.55'))                    //NaN
```

ხოლო მეთოდი Number() ახდენს მხოლოდ იმ ტექსტის დაყვანას როგორც მთელ ასევე აღწილად მნიშვნელობად რომელიც შედგება მხოლოდ რიცხვითი მნიშვნელობისაგან და ასევე წერტილისაგან „.“

```
alert ( Number(true))                                    // 1
alert ( Number(false))                                   // 0
alert ( Number('21'))                                    // 21
alert ( Number('21.55'))                                 // 21.55
alert ( parseInt('21 55'))                               // NaN
alert ( parseInt('ტექსტით დაწყებული 21.55'))            //NaN
```

## დროისა და თარიღის ფუნქციები

Javascript დროსთან სამუშაოდ იყენებს  ობიექტ Date(). მიმდინარე დროის შესაქმნელად გამოიყენება new Date() ობიექტი.

```
var now = new Date();
alert( now );
```

თუ კონსტრუქტორს გადაეცემა ერთი რიცხვითი არგუმენტი ის აღიქმება,  როგორც მილი წამის (1/1000 წამი ) მნიშვნელობა. თარიღის ამთვლელ წერტილად ნაგულისხმევია 1970 წლის 1 იანვარი GMT+0 (დროითი სარტყლის აღმნიშვნელია)

```
// 24 საათი 01.01.1970 GMT+0 წლის შემდეგ
var getD = new Date(1000 * 60 * 60 * 24 );
 // მილიწამი * 1000 = 1 წამი
// 60 * 60 = 1 საათი
// .... * 24 = 1 დღე
alert(getD); // შედეგი  02.01.1970  2 იანვარი 1970 წლის
```

დროის შექმნა შესაძლებელია სხვა არსებული პარამეტრების გადაწოდებით. სადაც წინასწარ არის განსაზღვრული შემდეგი თანმიმდევრობით:

```
var now = new Date(წელი, თვე, რიცხვი,საათი, წუთი, წამი, მილიწამი);

// 1 იანვარი 2016, 00:00:00
var date = new Date(2016, 0, 1, 0, 0, 0, 0);
```

```
// შედეგი იგივეა საათი,წუთი,წამი,მილიწამი ავტომატურად 0_ად არის განსაზღვრული
var date = new Date(2016, 0, 1 );


// დროის შექმნა მილიწამების სიზუსტით
var date = new Date(2016, 0, 1, 2, 3, 4, 567);  // 1.01.2016, 02:03:04.567
```

როგორც მაგალითიდან ჩანს აუცილებელი პირობაა:

- წელი შედგებოდეს 4 სიმბოლოიანი ჩანაწერისაგან ( 2016 და არა 16)
- თვის ათვლა იწყება ნულიდან და არა ერთიდან ( იანვარი - 0 )

გასათვალისწინებელია ისიც, რომ მოცემული სტილით დროის შექმნისას აუცილებელი პირობა მინიმუმ 2 არგუმენტის (წელი, თვე) გადაცემაა. დანარჩენი არგუმენტების არ მიწოდების შემთხვევაში ისინი ავტომატურად 0_ებად აღიქმებიან და არაფერი დაშავდება.

დაიმახსოვრეთ Javascript_ში დროის გამოსახვა ლოკალური დროის მიხედვით ხდება და არა UTC (Universal Coordinated Time), ანალოგიურია GMT ( Greenwich Mean Time )

დროით ობიექტს გააჩნია მთელი რიგი მეთდები, რომელიც უზრუნველყოფს არსებული დროის კონკრეტული მნიშვნელობის წარმოჩინებას. მაგ: სასურველია ინფორმირებულნი ვიყოთ მხოლოდ მიმდინარე წლის შესახებ და საერთოდ არ არის საჭირო მთლიანი დროის (წელი,თვე,რიცხვი, საათი,წუთი,წამი,მილიწამი,) გამოსახვა.

ძირითადად დროით ობიექტის მეთდები ორი ფაქტობრივად ერთგვაროვანი სახითაა წარმოდგენილი, რაც უზრუნველყოფს როგორც ლოკალურ ასევე UTC დროსთან მუშაობას. ასევე არსებობს მეთოდთა ერთობლიობა რომელიც უზრუნველყოფს ობიექტიდან მონაცემების მიღებას და პირიქით - მინიჭებას. მათგან get… თავსართით დაწყებული მეთდები შედეგს იღებს ობიექტიდან, ხოლო set… უზრუნველყოფს მინიჭებას.

ცხრილში VIII მოცემულია დროის ობიექტის მეთოდები განმარტებებითურთ.

| ცხრილი VIII | | | |
|---|---|---|---|
| მაგ. მიმდინარე დრო არის `var now = new Date();` 1 იანვარი 2016, 02:23:14.557 | | | |
| **მეთოდი** | **აღწერა** | **მაგალითი** | **შედეგი** |

| | | | |
|---|---|---|---|
| **getFullYear()** | მიიღება 4 ციფრანი წელი | now.**getFullYear**() | 2016 |
| **getMonth**() | გამოაქვს თვე ( გაითვალისწინეთ საწყისი თვე 0_ით სიმბოლორდება ) | now.**getMonth**() | 0 |
| **getDaTe()** | მიიღება თვის დღე 1_დან 31_მდე | now.**getDay**() | 1 |
| getHours() | მიიღება საათი | now.**getHours**() | 2 |
| **getMinutes()** | მიიღება წუთი | now.**getMinutes**() | 23 |
| **getSeconds()** | მიიღება წამი | now.**getSeconds**() | 14 |
| **getMilliseconds()** | მიიღება მილიწამი | now.**getMilliseconds**() | 557 |
| **setFullYear()** | წლის მინიჭება, შესაძლოა გადაეცეს თვეც და რიცხვიც | now.**setFullYear**(2022) | 1 იანვარი 2022, 02:23:14.557 |
| **setMonth()** | თვის მინიჭება, შესაძლებელია რიცხვისაც | now.**setMonth(5)** | 1 ივნისი 2016, 02:23:14.557 |
| **setDate()** | რიცხვის მინიჭება | now.**setDate(31)** | 31 იანვარი 2016,02:23:14.557 |
| **setHours()** | მიენიჭება საათი, შესაძლოა წუთი,წამი,მილიწამიც | now.**setHours**(5) | 31 იანვარი 2016,05:23:14.557 |
| **setMinutes()** | ენიჭება წუთი, შესაძლოა წამი, მილიწამიც | now.**setMinutes**(35) | 31 იანვარი 2016,02:35:14.557 |
| **setSeconds()** | ენიჭება წამი, შესაძლოა მილიწამიც | now.**setSeconds**(19) | 31 იანვარი 2016,02:23:19.557 |
| **setMilliseconds()** | ენიჭება მილიწამი | now.**setMilliseconds**(2) | 31 იანვარი 2016,02:23:14.2 |

აღსანიშნავია: ცხრილში ჩამოთვლილ ყველა მეთოდს გააჩნია მსგავსი მორგებული UTC_სარტყელზე. მაგ: **getUTCFullYear()**.

## ბრაუზერთან სამუშაო ფუნქციები

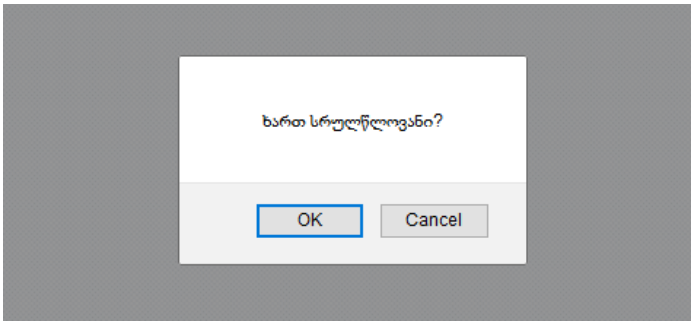
Javascript_ში window წარმოადგენს როგორც გლობალურ ელემენტს ასევე ბრაუზერის ფანჯარას. რეალურ რეჟიმში არსებულმა ობიექტმა განიცადა სრულყოფა და დროთა განმავლობაში მიიღო უამრავი დამატებითი ფუნქცია და მეთოდი. ბრაუზერთა სწრაფმა განვითარებამ შეცვალა დამოკიდებულება window ობიექტის ბევრი თვისებისადმი, გამომდინარე იქიდან რომ სხვადასხვა ბრაუზერი სხვადასხვა მეთოდებთან განსხვავებული თავსებადობით გამოიხატება. მაგრამ ეს სულაც არ ნიშნავს window ობიექტის მეთოდების ცოდნისაგან თავის არიდებას.

Javascript_ის პირველივე შემხებლობისას გავარჩიეთ გამოტანის ოპერატორი alert(). თუ იმ ლოგიკიდან გამოვალთ, რომ window წარმოადგენს სუპერ გლობალურ ობიექტს მაშინ ჩანაწერი window.alert() არ უნდა იყოს ლოგიკას მოკლებული. სინამდვილეში ასეცაა. მაგარამ გამომდინარე იქიდან, რომ window ყველა ობიექტის მშობელია სიმარტივის მიზნით მისი გამოუყენებლობა დასაშვებია. გარდა alert() მეთოდისა window ობიექტს გააჩნია რიგი მეთოდებისა, რომელიც აქტუალურობას არ კარგავს მიმდინარე ეტაპზეც. მათი სრული სია შეგიძლიათ იხილოთ შემდეგ მისამართზე http://www.w3schools.com/jsref/obj_window.asp . ჩვენს მიერ კი განხილული იქნება რამდენიმე მათთაგანი. (გამოყენებული იქნება სრული ჩანაწერი)

**window.print()**

არსებული ფუნქცია უზრუნველყოფს მითითებული ადგილის, არეალის ბეჭდვას.

**confirm()**

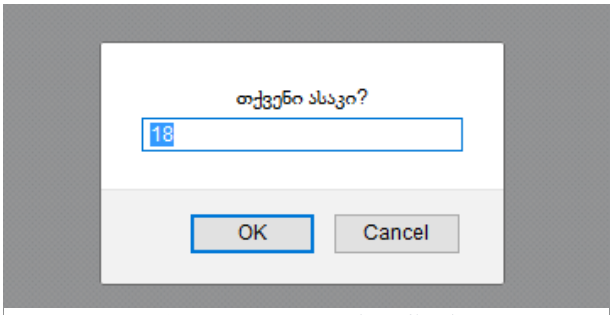



5.7 დიალოგური ფანჯარა confirm()

არსებობს დიალოგური ფანჯარა, რომლის მეშვეობითაც შესაძლებელია შეტყობინების გამოტანა ეკრანზე. (იხ.სურ.5.7) მისი სინტაქსი შემდგომია **confirm**("შეკითხვა"), დიალოგურ ფანჯარას გააჩნია ორი ღილაკი **OK** და **Cencel.** რომელთა გააქტიურებაც რეზულტატში გვაძლევს OK->**true** , Cencel->**false.**

**prompt()**

დიალოგური ფანჯარა **prompt()** შედგება 2 ნაწილისაგან ეს არის ტექსტის გამოტანის არეალი ინფორმაციის შესაყვანი ფორმა (იხ.სურ. 5.8). **prompt("თქვენი ასაკი","18").** მეორე მნიშვნელობა default_ად სიცარიელეს წარმოადგენს და მისი საერთოდ არ მითითებით არაფერი დაშავდება. **prompt**("თქვენი ასაკი")



5.8 prompt დიალოგური ფანჯარა

window მეთოდებიდან გამოსაყოფია **window.open(),** რომელიც ახალი ფანჯრის გახსნის ფუნქციონალს წარმოადგენს. მისი სრულყოფილი სინტაქსი შემდეგი სახისაა **window.open**("საიტის URL მისამართი","გახსნილი ფანჯრის სახელი","პარამეტრები"). **window.open()** მეთოდს შესაძლოა გადაეწოდოს მხოლოდ საიტის მისამართი დამატებითი პარამეტრებისა და სახელის გარეშე. ის უპრობლემოდ იმუშავებს.

```
window.open("http://google.ge");
```

არსებული მეთოდის ასეთი სახით გამოყენებისას გაიხსნება ახალი ფანჯარა ბრაუზერის ახალი ჩანართში - ტაბში . თანამედროვე ბრაუზერების უმეტეს ნაწილში მეთოდის ამგვარი ჩანაწერი ფანჯარას სწორედაც, რომ ახალ ჩანართში ხსნის და არა დამოუკიდებელ **popup** ფანჯრად. სრულყოფილი სინტაქსი უზრუნველყოფს ფანჯრის დამატებითი პარამეტრების განსაზღვრას, რაც თავისთავად შესაძლებელს ხდის გახსნილი ფანჯარა დამოუკიდებელი ზომებით და პარამეტრებით მოგვევლინოს.

```
win = window.open(url, name, params)
```

**url** - ის წარმოადგენს გასახსნელი ფანჯრის მისამართს.

**name** – მიეთითება ფანჯრის სახელი. რამდენიმე ფანრის გახსნის მცდელობისას თუ სახელი ერთი იქნა ახალ ფანჯარაში ჩანაცვლდება საიტის ვიზუალი და ყოველჯერზე ახალში არ გაიხსნება.

**params** – პარამეტრები უზრუნველყოფს ფანჯრის სრულყოფას. აღსანიშნავია, რომ დღეს მათი ნაწილი სხვადასხვა ბრაუზერებში სხვადასხვანაირად გამოისახება, შესაძლოა ზოგმა საერთოდ არ იმუშაოს. ასე რომ ყურადღებით გაარჩიეთ დამოუკიდებლად სხვადასხვა პარამეტრის თავსებადობა ბრაუზერებთან. იხ. ცხრილი IX

ცხრილი IX .

| N | მეთოდი | აღწერა | მნიშვნელობა |
|---|---|---|---|
| 1 | **left** | ფანჯრის დაშორება ეკრანის / ბრაუზერის **მარცხენა** კიდიდან | რიცხვი |
| 2 | **top** | ფანჯრის დაშორება ეკრანის / ბრაუზერის **ზედა** კიდიდან | რიცხვი |
| 3 | **width** | ფანჯრის სიგანე | რიცხვი |

| | | | |
|---|---|---|---|
| 4 | **height** | ფანჯრის სიმაღლე | რიცხვი |
| 5 | **menubar** | ბრაუზერის მენიუს ზოლის დამალვა/გამოჩენა | yes/no |
| 6 | **toolbar** | ნავიგაციის პანელის დამალვა / გამოჩენა | yes/no |
| 7 | **location** | საიტის URL სამისამართე ველის დამალვა / გამოჩენა | yes/no |
| 8 | **statusbar** | სტატუსის ველი | yes/no |
| 9 | **resizable** | ბრაუზერის ფანჯრის დაპატარავება / გაზრდა | yes/no |
| 10 | **scrollbar** | სქროლის დამალვა / გამოჩენა | yes/no |

როგორც აღვნიშნეთ მოცემული პარამეტრები დღევანდელ ბრაუზერებში, არც ისე სრულყოფილად მუშაობს, მაგრამ თუ ახალი ფანჯრის გახსნა არა ბრაუზერის ჩანართად (ტაბი) ჩამატება არამედ ფანჯრად ვიზუალიზება გსურთ აუცილებელია width და height პარამეტრების მინიჭება.

```
var newWin = window.open("http://google.ge/", "google", "width=200,height=200");
```

გახსნილი ფანჯრის დასახურად გამოიყენება window.close() ფუნქცია, რომელიც სხვადასხვა ბრაუზერებში განსხვავებულად მოქმედებს აქტიური ფანჯრის პირობებში, ზოგან ხურავს ავტომატურად, ზოგან გამოაქვს შეკითხვის ფორმა დაახლოებით „დარწმუნებულნი ხართ რომ ნამდვილად გინდათ ფანჯრის დახურვა?“ **newWin - ფანჯრის დასახურად გამოიყენეთ შემდგი კონსტრუქცია**
**newWin.close();**

არსებობს ასევე ფანჯრის გადატანის / გადაადგილების, ზომის ცვლილებისა და სქროლის ფუნქციები მეთოდები

- **win.moveBy(x,y) -** ფანჯრის მოძრაობა - წანაცვლება არსებული პოზიციიდან x , y მნიშვნელობების შესაბამისად. მაგ. ფანჯრის საწყისი მდებარეობაა left-0; top-0; win.moveBy(100,100) მეთოდის გამოყენებით პირველ ნაბიჯზე ფანჯარა განთავსდება **left-100px**; **top-100;** შემდეგი მოქმედებისას **left-200px; top-200;** და ა.შ

- **win.moveTo(x,y)** - ფანჯრის გადაადგილება ეკრანის / ბრაუზერის x, y კოორდინატებზე. მაგ. ფანჯრის საწყისი მდებარეობაა left-0; top-0; ხოლო win.moveTo(100,100) მეთოდის გამოყენებით ფანჯარა განთავსდება ფიქსირებულ **left-100px; top-100;**

- **win.resizeBy(width,height) -** მითითებული ფუნქციით შესაძლებელია არსებული ფანჯრის ზომების ცვლა მისი საწყისი ზომიდან გამომდინარე მრავალჯერადად

- **win.resizeTo(width,height)** მითითებული ფუნქციით შესაძლებელია არსებული ფანჯრის ზომის ცვლა ერთჯერადად.

- **win.resizeTo(width,height)** მითითებული ფუნქციით შესაძლებელია არსებული ფანჯრის ზომის ცვლა ერთჯერადად.

- **win.scrollBy(x,y)** მითითებული ფუნქციით შესაძლებელია სკროლის გადატანა x, y პოზიციებზე მრავალჯერადად

- **win.scrollTo(x,y)** მითითებული ფუნქციით შესაძლებელია სკროლის გადატანა x, y პოზიციებზე ერთჯერადად

**object screen**

**მომხმარებლის მონიტორის რეზოლუციის დასადგენად გამოიყენება ობიექტი screen. მისი ჩაწერა დასაშვებია, როგორც window ობიექტთან, ასევე მის გარეშე :**

- **screen**
- **window.screen**

**screen** ობიექტის თვისებათა ერთობლიობა წარმოდგენილია ცხრილ X:

ცხრილი X .

| N | მეთოდი | აღწერა |
|---|---|---|
| 1 | **screen.width** | გამოაქვს მონიტორის სრული სიგანე |
| 2 | **screen.height** | გამოაქვს მონიტორის სრული სიმაღლე |
| 3 | **screen.availWidth** | გამოაქვს მხოლოდ გამოყენებადი არეალის სიგანე ( მაგ.: თუ სტარტის ზოლი ან დამატებითი ინსტრუმენტების პანელი გამოყენებულია მარჯვენა ან მარცხენა მხარეს ) |
| 4 | **screen.availWidth** | გამოაქვს მხოლოდ გამოყენებადი არეალის სიმაღლე ( მაგ.: თუ სტანდარტულ მდგომარეობაშია windows ინსტრუმენტების პანელი განთავსებულია ქვედა პანელზე და მისი სიმაღლეა 40 px რაც აკლდება სრულ სიმაღლეს ) |
| 5 | **screen.colorDepth** | გამოაქვს ფერთა პალიტრის ბიტური მნიშვნელობა |

**object Navigator**

**ობიექტი Navigator შეიცავს ინფორმაციას მომხმარებლის ბრაუზერის შესახებ. არსებული ობიექტის გამოყენებით შესაძლებელია დადგენა თუ რომელ ბრაუზერს იყენებს მომხმარებელი.**
**navigator** ობიექტს გააჩნია შემდეგი თვისებები (იხ. ცხრილ XI):

ცხრილი XI .

| თვისებები | აღწერა |
|---|---|

| | |
|---|---|
| **navigator.appName** | გამოაქვს ბრაუზერის სახელი |
| **navigator.appCodeName** | გამოაქვს ბრაუზერის კოდური დასახელება |
| **navigator.appVersion** | გამოაქვს ბრაუზერის ვერსია დასახელება |
| **navigator.cookieEnabled** | ახდენს დადგენას ჩართულია თუ არა cookie_ების მხარდაჭერის რეჟიმი |
| **navigator.platform** | ადგენს ოპერაციულ სისტემას თუ რაზეა მორგებული ბრაუზერი |
| **navigator.userAgent** | ახდენს მომხმარებლის ბრაუზერის სათაურის გამოტანას, რომელსაც უგზავნის ბრაუზერი სერვერს გვერდის მოთხოვნის მომენტში |

**object History**

ობიექტი **History** ინახავს გვერდების ნავიგაციის მონაცემებს. მას გააჩნია 2 მეთოდი, რომელიც უზრუნველყოფს წინა და მომდევნო მისამართებზე გადასვლას:

- **history.back()** - უკან
- **history.forward()** - წინ


5.9 სანავიგაციო ისრები

ისინი იდენტურ მოქმედებებს ასრულებენ რასაც მოიცავს სურათ 5.9_ზე მონიშნული ისრები.

**object Location**

Location ობიექტი უზრუნველყოფს სამისამართე პანელი ინფორმაციის მართვას. მისი ფუნქციაა მიიღოს / მიანიჭოს URL მისამათი და მისი კომპონენტები.

Location ობიექტს გააჩნია რიგი მეთოდებისა და თვისებებისა. ცხრილ XII მოცემულია იმ თვისებების ჩამონათვალი, რომელსაც იყენებს Location ობიექტი

| | | ცხრილი XII |
|---|---|---|
| მაგ. მისამართი შემდეგია http://www.google.com:80/search?q=javascript#test | | |
| **მეთოდი** | **აღწერა** | **მნიშვნელობა** |
| **location.hash** | URL_ის ნაწილი რომელიც გამოსახულია სიმბოლო '#'_ის მერე | #test |

| | | |
|---|---|---|
| **location.host** | ჰოსტი და პორტი ერთდროულად | www.google.com:80 |
| **location.href** | მთლიანი URL მისამართი | http://www.google.com:80/search?q=javascript#test |
| **location.hostname** | ჰოსტი პორტის გარეშე | www.google.com |
| **location.pathname** | გზა საქაღალდისკენ | /search |
| **location.port** | გამოაქვს პორტი | 80 |
| **location.protocol** | პროტოკოლი | http: |
| **location.search** | ტექსტი სიმბოლო „?“ შემდგომ | ?q=javascript |

გარდა თვისებებია Location ობიექტს გააჩნია შემდეგი მეთოდები:

- **location.assign(url)** - გადაეცემა url მისამართი, რომელზეც ახდენს გადამისამართებას. ახდენს მითითებული მისამართის დოკუმენტის ჩატვირთვას
- **location.replace(url)** - ახდენს არსებული დოკუმენტის ჩანაცვლებას იმ დოკუმენტით რომელიც იტვირთება მითითებული url_ის შემთხვევაში. (დოკუმენტში მოიაზრება საიტის გვერდი ). **replace()** მეთოდი **assign()**_საგან განსხვავდება იმით, რომ ის არ ინახება ისტორიაში შესაბამისად replace() მეთოდით გახსნილ გვერდს არ გააჩნია **back** უკან დაბრუნების ღილაკი ის დეაქტივირებულია.
- **location.reload()** - არსებული მეთოდის მეშვეობით შესაძლებელია მიმდინარე გვერდის განახლება. შესაძლოა მეთოდს გადავცეთ ატრიბუტი **true** ან **false** . თუ მნიშვნელობა იქნება **true** განიცდის განახლებას ყოველთვის სერვერიდან, ხოლო **false** . ის შემთხვევაში ბრაუზერის ქეშიდან.

## ტაიმერის ფუნქციები

ზემოთქმულ მასალაზე დაყრდნობით შეგვიძლია დავასკვნათ, რომ ობიექტი window საკმაოდ მნიშვნელოვანი ფუნქციების გამაერთიანებელია. ვებ გვერდზე ხშირია მოცემულობა, როცა საჭიროა გარკვეული ინფორმაციის პერიოდული გამეორება, ან რომელიმე ინფორმაციის ერთჯერადად, მაგრამ კონკრეტული დროის შემდგომ ასახვაა საჭირო. window ობიექტის ძირითად მეთოდებად მოიხსენიება „ტაიმერის“ ფუნქციები: **setInterval( )** და **setTimeout**().

ორივე მეთოდის სინტაქსი ანალოგიურია განსხვავებულია მათი მუშაობის პრინციპი. **setInterval( )** უზრუნველყოფს მოქმედების მრავალჯერადად განმეორებადობას, ხოლო **setTimeout**() თვისი არსით ერთჯერადად. რა თქმა უნდა შესაძლებელია ისინი იდენტური ფუნქციონალით ვამუშაოთ, მაგრამ ყოველი აზრს არის მოკლებული, რადგანაც ისინი იმიტომ არსებობენ, რომ გამოიყენოთ დანიშნულებისამებრ.

მეთოდის სინტაქსი შემდეგია: ის შედგება ფუნქციისა და დროის მნიშვნელობისაგან, შესაძლოა გამოყენებულ იქნას არგუმენტებიც. დრო განისაზღვრება მილი წამებად ( 1 წთ - 1000 მილიწამი)

```
var timerId = setTimeout(func / code, delay[, arg1, arg2...])
```

განხილვისას შეგვხვდება ტერმინი ფუნქცია და მისი გამოყენება, რომელიც შემდეგ 5.5 თავშია განხილული, უმჯობესია გაურკვევლობის შემთხვევაში ეწვიოთ მითითებულ თავს.

**ამოცანა:**

> საიტის გახსნიდან 2 წთ_ში გამოგვიტანოს ერთჯერადად alert("მოგესალმებათ საიტი") ფანჯარა შეტყობინებით

ამოცანის გადასაჭრელად საჭიროა `setTimeout()` ფუნქციის გამოყენება. არსებულ დავალების გადასაწყვეტად გამოვიყენოთ ყველა შესაძლო გზა, რაც გაამარტივებს `setTimeout()` ფუნქციის არსის გაგებას.

გზა I - შეიქმნას ანონიმური ფუნქცია უშუალოდ `setTimeout()` მეთოდში

```
setTimeout( function (){ alert("მოგესალმებათ საიტი") } , 2000 )
```

ფუნქცია ვიზუალურად საკმაოდ სასიამოვნოდ გამოიყურება, უბრალოდ დიდი კოდის შემთხვევაში ერთობ მოუხერხებელია. ამისათვის უმჯობესია შეიქმნას ცალკე ფუნქცია.

გზა II ბევრად უფრო მისაღები და კლასიკურია

```
function showMessage() {
  alert("მოგესალმებათ საიტი")
}

setTimeout(showMessage, 2000 )
```

არსებული დავალება იმდენად მარტივია შესაძლებელია ფუნქციის გამოყენებისაგან საერთოდ თავის შეკავება და alert() მეთოდის სტრინგად გამოყენება

გზა III

```
setTimeout(  "alert("მოგესალმებათ საიტი")", 2000 )
```

არსებული ფუნქციაც ანალოგიურად იმუშავებს, მაგრამ ამგვარი ჩანაწერი მაინც არა რეკომენდირებულია.

თუ ფუნქცია იღებს მნიშვნელობად არგუმენტებს მაშინ დროით მნიშვნელობის შემდეგ მძიმის მეშვეობით შესაძლებელია არგუმენტების გადაცემა. მაგ:

```
function sum(a,b,c) {
```

```
 alert(a+b+c)
}

setTimeout(sum, 2000, 5,25,15 )   // 2 წთ _ ში ეკრანზე გამოდის შედეგი 45
```

არსებული ფუნქციის განულება, მწყობრიდან გამოყვანა შესაძლებელია **clearTimeout**() ფუნქციის მეშვეობით. რომელსაც არგუმენტად გადაეცემა ფუნქციის სახელი/ცვლადი რომელსაც ის მიენიჭება გამოცხადება/გამოძახებისას:

```
function sum(a,b,c) {
 alert(a+b+c)
}

var  t = setTimeout(sum, 2000, 5,25,15 )

clearTimeout(t) // ეკრანზე შედეგი აღარ გამოვა
```

ანალოგური სინტაქსი აქვს **setInterval**() მეთოდს. მისი შესაძლებელია **clearInterval**() მეთოდის საშუალებით.

**დავალება:**

ეკრანზე გამოვიდეს მთვლელი, რომელიც 10_დან უკუთვლით წამოვა 1_მდე 1 წამის ინტერვალით. და საბოლოოდ გადაამისამართოს გვერდი google.com_ზე.

```
var i = 10;
function timer() {
 alert(i)


  if (i>1){
   i--;
  }else{
   clearInterval(t);
  location.assign("http://google.com");
  }

}

var  t = setInterval (timer, 1000 )
```

# DOM

კლიენტზე ორიენტირებული Javascript_ის შესაძლებლობებში შედის სტატიკური HTML დოკუმენტის დინამიურად გარდაქმნა, ასევე ინტერაქციული ეფექტების შემუშავება. მისი მთავარი ორიენტირი ვებ გვერდების შემცველობასთან მუშაობაა. ცალკე საუბრის თემაა ობიექტები, რომლებიც წარმოადგენს უშუალოდ ვეგ ვერდის შემცველობას.

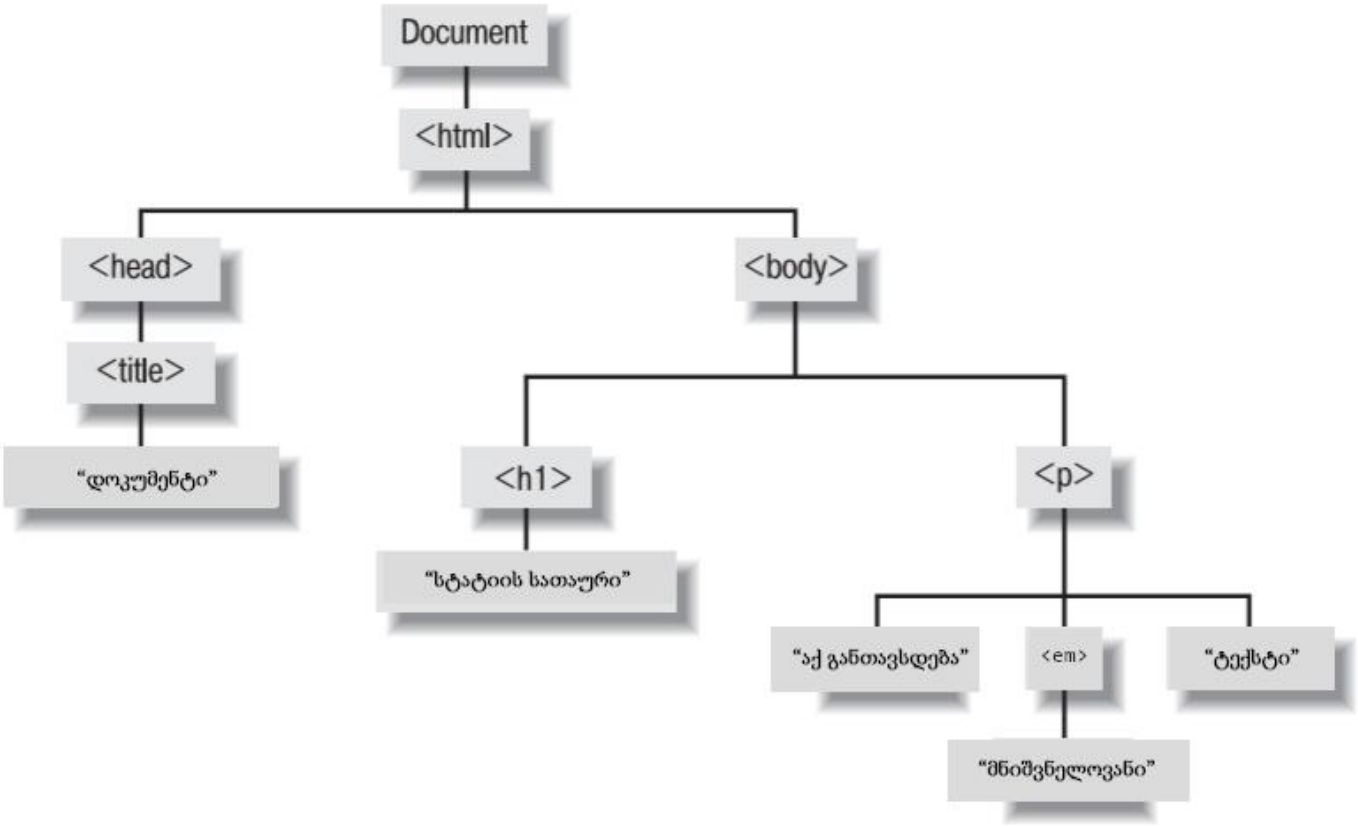
HTML დოკუმენტი შეიცავს ტექსტებს, სურათებს, ჰიპერბმულებს, ფორმის ელემენტებს და ა.შ. Javascript შეუძლია არსებულ ობიექტებთან მუშაობა და მათი მანიპულირება.

DOM (Document Object Model) დოკუმენტის ობიექტური მოდელი - ეს არის დოკუმენტის ობიექტებთან წვდომის განმსაზღვრელი ინტერფეისი.

HTML დოკუმენტი წარმოადგენს ერთმანეთში ჩადგმული ტეგების იერარქიულ სტრუქტურას, რომელიც DOM _ში წარმოდგენილია, როგორც „ობიექტების ხე“. ხის **კვანძები** წარმოადგნენ დოკუმენტის სხვადასხვა ტიპის ინფორმაციას, როგორიცაა HTML ელემენტები/ტეგები, ტექსტური ინფორმაცია, კომენტარები და ა.შ

მაგალითზე მოცემულია მარტივი HTML დოკუმენტი რომლის DOM სტრუქტურა წარმოდგენილია სურათ 5.10_ზე

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
  <h1> სტატიის სათაური </h1>
  <p>აქ განთავსდება <em>მნიშვნელოვანი</em> ტექსტი</p>
</body>
</html>
```




5.10 დოკუმენტის DOM სტრუქტურა

კვანძები ერთმანეთთან მიმართებაში შეიძლება „გენიალოგიურ“ ხის სტრუქტურას მივამსგავსოთ, სადაც კვანძი შესაძლოა წარმოადგენდეს **მშობელ** კვანძს (მაგ.:html). კვანძები რომლებიც განლაგებულნი არიან ერთი დონით ქვემოთ მშობელ კვანძებზე იწოდებიან **შვილეული** კვანძებად (html კვანძსთვის -> head და body). ერთ დონეზე მდებარე კვანძებს რომლებსაც გააჩნიათ ერთი მშობელი ერთმანეთისათვის წარმოადგენენ **დედმამიშვილ** კვანძებს ( მაგ.: head და body ერთმანეთისათვის, ასევე h1 და p )

DOM სტრუქტურაში კვანძი მოხსენიებულია როგორც - **Node.** სურათ 5.10_ზე წარმოდგენილია სხვადასხვა ტიპის Node_ები ( კვანძები ). ესენია: დოკუმენტი, ელემენტი, ტექსტი. ცხრილ XIII წარმოდგენილია ძირითადი კვანძები შესაბამისი ტიპის მნიშვნელობით. სრული სია იხილეთ http://www.w3schools.com/jsref/prop_node_nodetype.asp

ცხრილი XIII

| ინტრეფეისი | კონსტანტა nodeType | მნიშვნელობა |
|---|---|---|
| **Element** | Node.ELEMENT_NODE | 1 |
| **Text** | Node.Text_NODE | 3 |
| **Document** | Node.DOCUMENT_NODE | 9 |
| **Comment** | Node.COMMENT_NODE | 8 |

| | | |
|---|---|---|
| **DocumentFragment** | Node.DOCUMENT_FRAGMENT_NODE | 11 |
| **Attr** | Node.ATTRIBUTE_NODE | 2 |

DOM სტრუქტურის ფუძისეულ კვანძს წარმოადგენს Document ობიექტი. შევეცადოთ გადავიდეთ უშუალოდ სხვა ელემენტების მიმართიანობაზე და ვნახოთ როგორ ხდება მათთან წვდომა.

**document.documentElement** ახდენს წვდომას <html> ტეგთან. არსებობს თვისება რომელიც გამოსახავს ეკრანზე კვანძის სახელს **nodeName.**

```
<html>
<head>
  <title>დოკუმენტი</title>
</head>
<body>
 <h1> სტატიის სათაური </h1>
 <script>
          alert( document.documentElement.nodeName ) // html
   </script>
</body>
</html>
```

ძირითადად მიმართვიანობა არა html არამედ body ტეგზე ხდება ამისათვის გამოიყენება შემდეგი ჩანაწერი **document.body**

უშუალოდ ინდივიდუალურად ელემენტებზე მიმართვისათვის შესაძლოა გამოიყენოთ:

- **document.getElementById()-** ელემენტზე მიმართვა კონკრეტული id_ის მიხედვით;
- **document.getElementsByClassName()**-მიმართვა ელემენტზე class სელექტორის მიხედვით
- **document.getElementsByTagName()**-ელემენტებზე ტეგების სახელწოდებებით მიმართვა
- **document.getElementsByName()-** ელემენტებზე ატრიბუტ name_ის მიხედვით მიმართვა

გასათვალისწინებელია, რომ getElement მხოლობითში იხმარება მხოლოდ ID_სთან მიმართებაში, რადგანაც საიტზე შესაძლებელია  მხოლოდ ერთი განუმეორებელი id მქონე ელემენტის არსებობა. დანარჩენ შემთხვევებში ერთი და იმავე კლასის, ტეგის სახელისა და ატრიბუტ name_ის ელემენტები მრავლად შეგხვდეს. მათ ოდნავ მოგვინაებით შევეხოთ. მანამდე გავიგოთ უშუალოდ მიმართვიანობის არსი.

**დავალება:**

Javascript_ის მეშვეობით შევიტანოთ ტეგში ტექსტი.

მოცემული დავალების შესასრულებლად საჭიროა მოვახდინოთ ტეგზე მიმართვიანობა, და გამოვიყენოთ მეთოდი **innerHTML**  მისთვის ინფორმაციის მისანიჭებლად:

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
 <h1 id="caption"> </h1>  // საწყის ეტაპზე h1  ცარიელია
 <script>
            document.getElementById("caption").innerHTML = "საიტის სათაური";
// არსებული ჩანაწერის გამოყენებით ტექსტ - „საიტის სათაური“ ხელოვნურად მოვათავსებთ  ტეგ
<h1>_ში და მიიღება შედეგი  <h1 id="caption"> საიტის სათაური </h1>
    </script>
</body>
</html>
```

კონკრეტული თვისების უმრავლესობა ორმაგი ხასიათისაა და როგორც მინიჭების ასევე არსებული ელემენტის თვისების ამოღების უნარი გააჩნია. მაგ.: იმავე innerHTML_ის გამოყენება შემდეგ სახეს მოგვცემს

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
 <h1 id="caption"> განთავსებულია სათაური </h1>
 <script>
            alert( document.getElementById("caption").innerHTML )
// არსებული ჩანაწერის გამოყენებით ეკრაბზე გამოვა შედეგი  “განთავსებულია სათაური”
    </script>
</body>
</html>
```

არსებული მოცემულით შესაძლებელია ასევე საკმაოდ მარტივი და საინტერესო დავალების შესრულება.: მოახდინეთ ერთ ტეგში არსებული ტექსტის მეორეში კოპირება და ჩანაწერის **„ეს საინტერესოა**“_ს თანდართვა;

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
 <h1 id="caption"> განთავსებულია სათაური </h1>
 <em  id="copyText"> </em>
```

```
 <script>
        document.getElementById("copyText").innerHTML  = document.getElementById("caption") .in-
nerHTML + "ეს საინტერესოა"
    </script>
</body>
</html>
```

შედეგი ბრაუზერში

<h1>

# განთავსებულია სათაური

*განთავსებულია სათაური ეს საინტერესოა*

Javascript_ის მეშვეობით შესაძლებელია კონკრეტული ელემენტების სტილირება. რა თქმა უნდა სტილების სრულფასოვანი გაწერა აზრს მოკლებულია და მიის გამოყენება მხოლოდ კონკრეტული ლოგიკის შესაბამისადაა საჭირო. მაგ. საჭიროა ღილაკზე ხელის დაჭერისას ბლოკის გამოჩენა / გაქრობა. რა თქმა უნდა ასეთ შემთხვევაში დასაშვებია სტილების მინიჭება.
არსებობს სტილის მინიჭების ორი გზა. ეს არის ობიექტ **style** გამოყენება ან ატრიბუტის მინიჭება. ორივე შემთხვევაში სტილები ელემენტს ენიჭება ხაზოვანი სტილის მინიჭების მეშვეობით (Inline style):

```
 <h1 id="caption"> განთავსებულია სათაური </h1>
 <script>
        document.getElementById("caption ").style.color ="red"   // h1 ტექსტი გახდება წითლი
    </script>
```

მაგალითზე მოცემული ჩანაწერით შესაძლებელია 1 სტილის მინიჭება და ყოველი ახალი სტილის მიცემისათვის გამოყენებული უნდა იქნას ახალი ჩანაწერი.

```
<h1 id="caption"> განთავსებულია სათაური </h1>
<script>
  document.getElementById("caption ").style.color ="red"   // h1 ტექსტი გახდება წითლი
  document.getElementById("caption ").style.backgroundColor ="#fff44c"   // h1 ფონი - ყვითელი
</script>
```

Javascript_ით სტილების ბრძანებების სრული ჩამონათვალი შეგიძლიათ იხილოთ შემდეგ მისამართზე http://www.w3schools.com/jsref/dom_obj_style.asp

DOM სტრუქტურის მეშვეობით შესაძლებელია კონკრეტულ ელემენტის ატრიბუტზე წვდომა, მისი მინიჭება ან კითხვადობა. style წარმოადგენს ელემენტის ატრიბუტს როცა ის გაწერილია ხაზოვან რეჟიმში. თუ ერთდროულად რამდენიმე სტილის მინიჭება გჭირდებათ სასურველია გამოიყენოთ მეორე გზა.

```
<h1 id="caption"> განთავსებულია სათაური </h1>
<script>
  document.getElementById("caption ").setAttribute("style","color: red; background-color: #fff44c; font-size: 15px" )
</script>
```

**setAttribute()** მეთოდის გამოყენებით შესაძლებელია ნებისმიერი ატრიბუტი მინიჭება html ტეგებისათვის. არსებული მეთოდი მოითხოვს ორ არგუმენტს: ეს არის ატრიბუტის სახელი რომელსაც ანიჭებთ და მნიშვნელობა რომელიც მიეკუთვნება ამ ატრიბუტს.

მაგ.: setAttribute() მეთოდის მეშვეობით მივანიჭოთ ელემენტს **class="logo"**

```
<h1 id="caption"> განთავსებულია სათაური </h1>
<script>
  document.getElementById("caption ").setAttribute("class","logo" )
</script>
```

მიღებულ შედეგში ტეგ h1 ექნება დამატებული ატრიბუტი class

```
<h1 id="caption" class="logo" > განთავსებულია სათაური </h1>
```

გარდა ელემენტის ატრიბუტის მინიჭებისა შესაძლებელია მოხდეს ელემენტის ატრიბუტის დადგენა **getAttribute()** მეთოდის მეშვეობით, რომელიც თავის მხრივ ერთ არგუმენტს საჭიროებს, თუ რომელი ატრიბუტის მნიშვნელობის დადენა გვინდა:

მაგ.:

```
<h1 id="caption" class="logo" > განთავსებულია სათაური </h1>

<script>
    alert( document.getElementById("caption ").getAttribute("class" ) // logo
)
</script>
```

გარდა **document.getElementById()** მიმართვისა შესაძლებელია ელემენტებზე მიმართვა ტეგის სახელისა, კლასებისა და ატრიბუტ name_ის მეშვეობით. ისინი თავის მხრივ საიტზე რამდენიმე ერთგვაროვანი შესაძლოა იყოს. ამიტომ ისინი აღიქმებიან როგორც მასივი და უშუალოთ მათგან ერთ ან რამდენიმეზე მიმართვისას საჭიროა დაკონკრეტებია იმ **ინდექსისა** თუ რომელს უკავშირდებით. მაგ.:

```
<html>
<head>
  <title>დოკუმენტი</title>
</head>
```

```
<body>
  <div class="container"> ტექსტი 3 </div>
  <h1 id="caption"> განთავსებულია სათაური </h1>
  <h2> განთავსებულია  მეორე სათაური </h2>
  <p> ვრცელი ტექსტი 1 </p>
  <h2> განთავსებულია მესამე სათაური </h2>
  <p> ვრცელი ტექსტი 2 </p>


  <div class="container"> ტექსტი 5 </div>

  <div class="about"> ტექსტი კომპანიის შესახებ </div>

    <input type="radio" name="gender" />
    <input type="radio" name="gender" />
  <div class="container"> ტექსტი 4 </div>

</body>
</html>
```

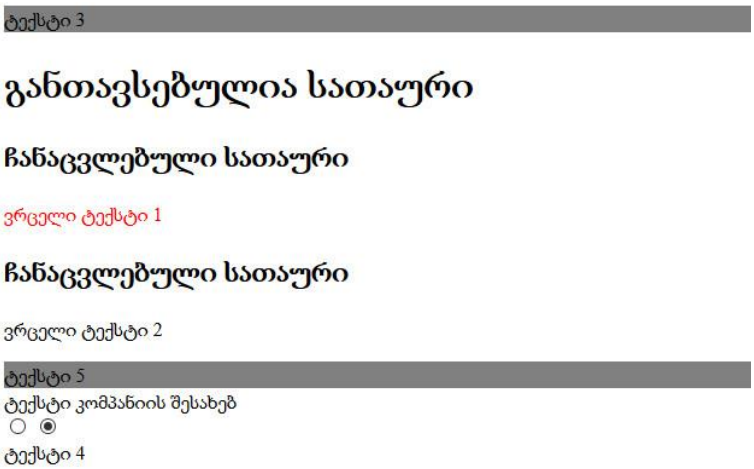
არსებულ კოდზე დაყრდნობით სასურველია შემდეგი სტილის დოკუმენტის მიღება:

- ყველა h2_ში გამოისახოს ტექსტი 'ჩანაცვლებული სათაური';
- მხოლოდ პირველი <p> ტეგის ფერი გახდეს წითელი;
- საიტზე ნებისმიერ ადგილას განთავსებული პირველი და მეორე class="container" ფონად გამოესახოს სერი ფერი;
- მხოლოდ მეორე რადიო name_ით **gender** საწყის მნიშვნელობად იყოს მონიშნული (checked).

```
<script>
// 1 -  ყველა h2_ში გამოისახოს ტექსტი 'ჩანაცვლებული სათაური';

// გამომდინარე იქიდან რომ არ ვიცით წინასწარ რამდენი ელემენტი h2 იქნება საიტზე გამოყენებული უმჯობესია ციკლის მეშვეობთ ჩავწვდეთ და ყველა h2 მივაგნოთ ეს მეტად დინამიური იქნება :


for (var i = 0; i <  document.getElementsByTagName("h2").length; i++) {
            document.getElementsByTagName("h2")[i].innerHTML = "ჩანაცვლებული სათაური"
}

// 2-  მხოლოდ პირბელი p ტეგის გერი გახდეს წითელი
document.getElementsByTagName ("p")[0].style.color = "red"

// 3-  მხოლოდ პირბელი p ტეგის გერი გახდეს წითელი
```

```
document.getElementsByClassName("container")[0].style.backgroundColor = "gray";
document.getElementsByClassName("container")[1].style.backgroundColor = "gray";

შესაძლოა ეს ჩანაწერი შემდეგნაირადაც გამოისახოს
document.getElementsByClassName("container")[0].style.backgroundColor =
document.getElementsByClassName("container")[1].style.backgroundColor = "gray";



// 4 -  მხოლოდ მეორე რადიო იყოს cheked
document.getElementsByName("gender")[1].setAttribute("checked","checked")

// უფრო მოკლე გზაა
 document.getElementsByName("gender")[1].checked = true
</script>
```

მოცემული კოდის მანიპულირებით მიღებული შედეგი გამოსახულია 5.11 სურათზე. შეეცადეთ გაარჩიოთ მოცემული მაგალითი, ააწყოთ თქვენთვის სასურველი html დოკუმენტის ჩონჩხი და მოახდინოთ წვდომა ტეგებთან, ცვალოთ მათი სტილები და ინფორმაცია.




5.11 სტილშეცვლილი დოკუმენტი

Dom სტრუქტურის მეშვეობით შესაძლებელია ახალი ელემენტების, ტექსტური node_ების შექმნა მათთვის ატრიბუტების გენერირება და მინიჭება, ასევე უკვე არსებული ელემენტების წაშლა. მიდგომა საკმაოდ ხშირად გამოყენებადია. მისი საშუალებით შესაძლებელია html დოკუმენტის ჩონჩხის ცვლილება.

ახალი ელემენტის შესაქმნელად გამოიყენეთ მეთოდი **createElement().** ის მოითხოვს ატრიბუტად მხოლოდ ტეგის დასახელებას და თავად  ახდენს მის ტეგად გარდაქმნას.

```
<script>
        document.createElement("div");
</script>
```

მაგრამ ტეგის შექმნა მის ავტომატურ დამატებას html ჩონჩხში არ ნიშნავს, ასევე ის ცარიელია, შევეცადოთ შევუქმნათ მას ტექსტური შემცველობა, მივანიჭოთ ატრიბუტი id="newElement". და მხოლოდ ამის შემდეგ განვათავსოთ საჭირო ადგილას ჩვენს მიერ წინასწარ შექმნილ html დოკუმენტში

```
<script>
//  <div> ელემენტის შექმნა
var elem = document.createElement("div");

//  ატრიბუტ id შექმნა
var attr = document.createAttribute("id");

//  ატრიბუტ id_ის მნიშვნელობის განსაზღვრა
attr.value = " newElement";

//  ატრიბუტის div ელემენტზე მიბმა / მინიჭება
elem.setAttributeNode(attr); // არსებულ დონეზე მიღებული შედეგია <div id='newElement'></div>

//  ტექსტური კვანძის - ტექსტური ინფორმაციის შექმნა
var txt = document.createTextNode("ტექსტი ახალ ელემენტში");

//  ტექსტური ინფორმაციის div _ში განთავსება
elem.appendChild(txt) // არსებულ დონეზე მიღებული შედეგია <div id='newElement'>"ტექსტი ახალ
ელემენტში" </div>



//  div_ის ბოლო ელემენტად განთავსება body_ტეგში
document.body.appendChild(elem);
</script>
```

მოცემული სკრიპტი შემდეგნაირად იმუშავებს უშუალოდ html დოკუმენტში:

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
  <div class="container"> ტექსტი 3 </div>
  <h1 id="caption"> განთავსებულია სათაური </h1>
  <div class="about"> ტექსტი კომპანიის შესახებ </div>

//  body_ის უკანასკნელ ელემენტად დააამატებს
//-------------------------------------------------------------
<div id="newElement"> ტექსტი ახალ ელემენტში </div>
//-------------------------------------------------------
<script>
var elem = document.createElement("div");
var attr = document.createAttribute("id");
```

```
attr.value = " newElement";
elem.setAttributeNode(attr);
var txt = document.createTextNode("ტექსტი ახალ ელემენტში");
elem.appendChild(txt)
document.body.appendChild(elem);
</script>

</body>
</html>
```

თუ არსებული ბლოკის არა ბოლოში არამედ კონკრეტულ ადგილას განთავსებაა საჭირო საჭიროა მეთოდ **insettBefore()** გამოყენება. ის მოითხოვს ორ ატრიბუტს, პირველი ეს არის რას ვსვამთ, მეორე რომელი ბრძანების წინ.

ზედა მაგალითზე დაყრდნობით შექმნილი ბლოკი არა body ტეგის ბოლოში არამედ  ..... <h1 id="caption"></h1> ტეგებს შორი განთავსდეს (წერტილების ადგილას).

ამისათვის საჭიროა scritp_ის ბოლო ბრძანება `document.body.appendChild(elem)` შეიცვალოს

```
document.body.insertBefore(elem, document.getElementById("caption"))
```

Dom_ის მეშვეობით შესაძლებელია კონკრეტული ელემენტის პირველ(**firstChild**) და უკანასკნელ (**lastChild**) ელემენტებზე წვდომა. ასევე ერთ დონეზე განლაგებულ შემდეგ (**nextSibling**) და წინა (**previousSibling**) ელემენტებზე.

```
<html>
<head>
  <title>დოკუმენტი</title>
</head>
<body>
 <div> ტექსტი 1 </div>
   <ul id="partner">
      <li> განათლების სამინისტრო </li>
      <li> ჯანდაცვის სამინისტრო </li>
      <li> სპორტისა და ახალგაზდობის სამინისტრო </li>
      <li> ენერგეტიკის სამინისტრო </li>
   </ul>
 <div> ტექსტი 2 </div>

<script>
var elem = document.getElementById("partner");
alert( elem.firstChild.innerHTML); // განათლების სამინისტრო
```

```
alert( elem.lastChild.innerHTML);  //  ენერგეტიკის სამინისტრო
alert( elem.nextSibling.innerHTML);  //  ტექსტი 2
alert( elem.previousSibling.innerHTML);  //  ტექსტი 1
</script>

</body>
</html>
```

გარდა ელემენტების შექმნისა არსებობს ელემენტების წაშლის **removeChild()** და ჩანაცვლების **replace.Child()** მეთოდები.

```
<html>
<head>
   <title>დოკუმენტი</title>
</head>
<body>
  <div> ტექსტი  1  </div>
    <ul id="partner">
        <li> განათლების სამინისტრო </li>
        <li> ჯანდაცვის სამინისტრო  </li>
        <li> სპორტისა და ახალგაზდობის სამინისტრო   </li>
        <li> ენერგეტიკის სამინისტრო   </li>
    </ul>
  <div id="txt"> ტექსტი 2  </div>
<script>
var elem = document.getElementById("partner");
document.body.removeChild(elem);  //  წაიშლება მთლიანი <ul> თავისი li_ებით

var elem2 = document.getElementById("txt");
var txt = document.createTextNode("შეცვლილი ტექსტი ");
document.body.replaceChild(txt, elem2);  // შედეგი  <div id="txt"> შეცვლილი ტექსტი  </div>
</script>
</body>
</html>
```

## 8.5. ფუნქციებთან მუშაობა

**მიმდინარე პარაგრაფის თემატიკა**

- ფუნქციების შექმნა და გამოყენება
- „ლამბდა“ გამოსახულებები

ფუნქცია თავისი არსით ავტონომიურად მოქმედ კოდს წარმოადგენს, რომლის ფუნქციონალის გაწერა ერთჯერადად ხდება, ხოლო გამოყენება შესაძლებელია მრავალჯერ. მაგ. საჭიროა ეკრანზე რაიმე რიცხვის კვადრატის გამოტანა. როგორც თქვენთვისაა ცნობილი რიცხვის კვადრატში ასაყვანად საჭიროა არსებული რიცხვის თავის თავზე გამრავლება. ე.ი პროცესი უცვლელია და მისი ყოველ ეტაპზე თავიდან დაწერა კოდის დუბლირებას წარმოადგენს, რაც შესაბამისად არც ისე კარგ ტონად მიიჩნევა.

Javascript_ში ფუნქციები გვევლინებიან ობიექტებად და სახელი რომლის მინიჭებაც ხდება, წარმოადგენს მის იდენთიფიკატორს, შესაბამისად ის რეგისტრზე დამოკიდებულია. მაშასადამე დაუშვებელია ფუნქციის სახელებად გამოყენებულ იქნას Javascript_ის რეზერვირებული სიტყვები (იხ.ცხრილი I), ასევე სხვა გლობალური ობიექტების სახელები. მაგ.: უკვე არსებული ცვლადების დასახელებები და ობიექტის იდენთიფიკატორები. თუ კოდში ერთიდაიგივე სახელით აღწერილია ორი ფუნქცია მაშინ არსებული ფუნქციის გამოძახებისას ხდება წვდომა არსებული სახელით შექმნილ ბოლო ფუნქციასთან. ფუნქცია შედგება ორი ნაწილისაგან ეს არის ფუნქციის სახელი და ტანი, სადაც ხდება სკრიპტის მობილიზება. სტანდარტული (სახელდებული) ფუნქციის დეკლარირება ხორციელდება შემდეგნაირად

```
function showMessage() {  // showMessage - სახელი
  ... ფუნქციის ტანი
}
```

ხოლო ანონიმური ტიპის ფუნქცია

```
var showMessage = function() {  // showMessage - სახელი
  ... ფუნქციის ტანი
}
```

ფუნქციის დეკლარირება მის ავტომატურ გამოყენებას არ გულისხმობს და არც ითვალისწინებას. როგორც ითქვა ერთგვაროვანი მოქმედების მრავალჯერ გამოყენებაა საჭირო ამისთვის სასურველ ადგილზე საჭიროა ფუქციის გამოძახება რომელიც მდგომარეობს შემდეგში

```
function showMessage() {
  alert("შეტყობინება");
}
// ფუნქციის გამოძახება შესაძლებელია მრავალჯერადად. არსებული მაგალითზე დაყრდნობით ეკრანზე გამოვა alert_დიალოგული ფანჯარა ორჯერ ტექსტით 'შეტყობინება'

showMessage()       // ფუნქციის გამოძახება
showMessage()       // ფუნქციის გამოძახება
```

სტანდარტული და ანონიმური ტიპის ფუნქციები ერთმანეთისაგან ფუნქციის წვდომით განსხვავდებიან. თუ სტანდარტულ ფუნქციასთან წვდომა ნებისმიერი ადგილიდან არის შესაძლებელი, ანონიმურთან მხოლოდ მისი დეკლარირების შემდგომ.

```
//სტანდარტული ფუნქცია
showMessage()       // შედეგი ეკრანზე 'შეტყობინება'

function showMessage() {
  alert("შეტყობინება");
}

showMessage()       // შედეგი ეკრანზე 'შეტყობინება'

//ანონიმური ფუნქცია
showMessage()       // ფიქსირდება შეცდომა

var showMessage  = function() {
  alert("შეტყობინება");
}

showMessage()       // ამ ეტაპზე მუსაობს --- შედეგი ეკრანზე 'შეტყობინება'
```

ფუნქციას, როგორც მნიშვნელობის დაბეჭდვა ( რომელიმე გამოტანის ოპერატორის alert(), console.log(), document.write() ) , ასევე მისი დაბრუნება შეუძლია. ფუნქციიდან ინფორმაციის დაბრუნება საკმაოდ მნიშვნელოვანი დეტალია, გამომდინარე იქიდან, რომ შესაძლოა მიღებული მნიშვნელობა გამოყენებულ იქნას სხვა დანიშნულებით. მაგ.: დავალება მდგომარეობს შემდეგში ერთი ფუნქციის მეშვეობით უნდა მოხდეს რიცხვის კვადრატის დადგენა, მეორე ფუნქცია ადგენს ამ რიცხვის უდიდეს გამყოფს და შედეგად უნდა მივიღოთ მათი სხვაობა. ამისათვის საჭიროა ფუნქციები რომლების უზრუნველყოფენ: პირველი - კვადრატის დადგენას, მეორე უდიდესი გამყოფის პოვნას. როგორც ატყობთ დავალება არ მდგომარეობს მათ დაბეჭდვაში. შესაბამისად გამოტანის ოპერატორების ფუნქციაში გამოყენების არავითარი საჭიროება არ არსებობს. პროგრამისტისთვის მნიშვნელოვანია მიიღოს ინფორმაცია, რომელზეც შემდგომ იმოქმედებს. ასეთ შემთხვევაში ფუნქციაში გამოიყენება დაბრუნების ოპერატორი **return**

```
function showMessage() {
        return "შეტყობინება";
}
```

არსებულ შემთხევაში ფუნქციის გამოძახება არ მოგვცემს ვიზუალურ შედეგს ეკრანზე, მაგრამ იძლევა საშუალებას არსებულ მნიშვნელობაზე მოვახდინოთ მანიპულირება.

```
function showMessage() {
        return "შეტყობინება";
}
```

```
alert ( "საგანგაშო" + showMessage())    // შედეგი ეკრანზე - საგანგაშო შეტყობინება
```

ფუნქციიდან **return** მეთოდით დაბრუნებული შედეგის მინიჭება შესაძლებელია ასევე რაიმე ცვლადზე. მაგ.:

```
function showMessage() {
        return "შეტყობინება";
}
var sms = showMessage( )
alert ( "სასარგებლო" + sms)   // შედეგი ეკრანზე - სასარგებლო შეტყობინება
```

გაითვალისწინეთ ფუნქციაში შეუძლებელია რამდენიმე მნიშვნელობის რამდენიმე return მეთოდით მიღება, რადგანაც პირველივე return _ის გამოძახება იწვევს ფუნქციის მოქმედების შეწყვეტას და მის შემდგომ არსებული კოდის შესაბამისად აღარ აღიქმება. უხეშად რომ შევადაროთ, რა გავლენასაც ახდენდა break ოპერატორი switch კონსტრუქციასა და ციკლებში, ფაქტობრივად იგივე ფუნქციონალის შეთავსება უხდება return_ს გარდა ძირითადი ფუნქციისა, რომელიც აბრუნებს მნიშვნელობას

```
function showMessage() {
        return "სასიამოვნო";
        return "სიახლე";
}
var sms = showMessage( )
alert (sms)          // შედეგი ეკრანზე მხოლოდ - სასიამოვნო     და არავითარ შემთხვევაში სასიამოვნო სიახლე
```

ფუნქციების მნიშნველოვან დანამატს წარმოადგენს **არგუმენტები.** არგუმენტები ეს ის მნიშვნელობებია რომლის გადაცემაც და დამუშავებაც შესაძლებელია ფუნქციის მიერ. შესაძლებელია ფუნქციისათვის, როგორც ერთი ასევე რამდენიმე არგუმენტის ერთდროულად გადაცემა. მაგ.: დავალების შესაბამისად დავწეროთ რიცხვის კვადრატში აყვანის ფუნქცია.

```
function square(x ) {   // x  წარმოადგენს არგუმენტეს
        return x * x ;
}

alert ( square( 5  ) )  // შედეგი ეკრანზე  25
alert ( square( 10  ) )          // შედეგი ეკრანზე  100
alert ( square( 7  ) )  // შედეგი ეკრანზე  49
```

შესაძლოა არგუმენტად  გადაეცემოდეს ესა თუ ის კონკრეტული ცვლადი

```
function square( x  ) {    // x  წარმოადგენს არგუმენტეს
            return x * x ;
}

var n = 11;

alert ( square( n  ) )  // შედეგი ეკრანზე  121
```

დაიმახსოვრეთ არგუმენტის  სახელი (მოცემულ შემთხვევაში - **x** ) არ წარმოადგენს მნიშვნელობას რომელიც ხილულია ფუნქციის გარეთ. ის მხოლოდ ფუნქციის შიგნით აღიქმება. ფუნქციის გარეთ x_ის დაბეჭდვა არ მოხერხდება , შედეგი ეკრანზე არ გამოისახება.

```
function square( x  ) {    // x  წარმოადგენს არგუმენტეს
            return x * x ;
}

alert ( x )  // Uncaught ReferenceError: x is not defined - შეგიძლიათ გადაამოწმოთ consol_ში
```

არგუმენტების განსაზღვირსას გასათვალისწინებელია, რომ ფუნქციამ უნდა მიიღოს იგივე რაოდენობის არგუმენტების გამოძახებისას. მაგ:

```
function test( x,  y , z ) {
            return x + y + z ;
}
alert ( test(5, 7, 3 ) ) // 15
```

არგუმენტების არასრულად გადაცემის შემთხვევაში არსებული არგუმენტები მიენიჭება საწყის ცვლადებს სანამ ისინი იარსებებს, დანარჩენი არგუმენტის მნიშვნელობები გახდება undefined. მაგ.: მხოლოდ 2 პარამეტრის მიწოდების შემთხვევაში ისინი იქნებიან **x,y** არგუმენტები და არა **x, z** ან  **z,y** შესაბამისად  z არგუმენტი იქნება undefined.

```
function test( x,  y , z ) {
            return x + y + z ; // 5+3+undefined
}
alert (test(5, 3 ))       // NaN
```

```
alert (test(5, 3, 8))    // 16
```

დამატებითი არგუმენტების გადაცემის შემთხვევაში, როცა ისინი არ იყო წინასწარ განსაზღვრული ფუნქცია არ იღებს მათ მხედველობაში, ფაქტობრივად უგულველყოფს მათ და არ წარმოაჩენს. ისინი არ აღიქმებიან არც შეცდომებად. მათი არსებობა საერთოდ არსად ჩანს

```
function test( x,  y , z ) {
        return x + y + z ; // 5+3+2
}
alert (test(5, 3, 2, 8, 9, 4))      //  შედეგი 10 ---- 8,9,4  უგულველყოფილია, არც შეცდომა ,არც ყურადღების გამახვილება
```

არსებობს გზა თავის დასაცავად, თუ რომელიმე კონკრეტული არგუმენტი არ იქნა გადაცემული მოხდეს ამ არგუმენტის ჩანაცვლება რაიმე მნიშვნელობით რათა ფუნქციამ არ გამოიტანოს შეცდომა. მაგ.:

```
function test( x,  y , z ) {
        x = x || 1; // თუ x არგუმენტი არ გადმოეცა მნიშვნელობა x გახდება 1_ის ტოლი
        y = y || 10; // თუ y არგუმენტი არ გადმოეცა მნიშვნელობა y გახდება 10_ის ტოლი
        z = z || 20; // თუ z არგუმენტი არ გადმოეცა მნიშვნელობა z გახდება 20_ის ტოლი

        return x + y + z ;
}

alert( test(5, 3, 2) )  // 10
alert( test(5, 3) )     //  28   - - - - x = 5, y = 3, z = 20 რადანაც z არ გადაეცა მნიშვნელობად აიღო 20
alert( test(5) )        //  35   - - - - x = 5, y = 10, z = 20
alert( test() )         //  31   - - - - x = 5, y = 10, z = 20
```

Javascript_ს გააჩნია **arguments** ობიექტი, რომლის საშუალებით შესაძლებელია ყველა არგუმენტის მიღება წინასწარ განუსაზღვრელი რაოდენობის მიუხედავად. **arguments** ობიექტი იგივე მასივია, შესაბამისად მასივის მეთოდებისა და ფუნქციები მიესადაგება

```
function test() {

        return arguments.length;
```

```
}

alert( test(5, 3, 2) )  // 3
```

თითოეულ ელემენტთან წვდომა თქვენთვის ასევე ცნობილია

```
function test() {
        return arguments[0]+arguments[2];
}
alert( test(5, 3, 2) )  // 7
```

მაგ. საჭიროა დაიწეროს ფუნქცია რომელიც მოახდენს რიცხვთა ჯამის წარმოჩენას, მაგრამ საწყის ეტაპზე არ არის გარკვეული თუ რამდენი რიცხვის დაჯამება იქნება საჭირო. ეს შეიძლება იყოს 4,10, და ა.შ. ამისათვის გამოიყენება ობიექტი arguments რომლის მეშვეობითაც შესაძლებელია არგუმენტების რაოდენობის დადგენა და მათზე მანიპულირება შეცდომების გარეშე ( როგორც იყო წინა შემთხვევაში განუსაზღვრელი  არგუმენტის მნიშვნელობა undefined ).

```
function test() {
        var sum = 0;
          for (var i = 0; i <  arr.length; i++) {
                  sum+= arguments[i];
          }
return sum;
}

alert( test(5, 3, 2) )  // 10
alert( test(2,11,25, 4, 3) )       // 45
alert( test(10, 23) )  // 33
```

ფუნქციის თავისებურებას წარმოადგენს „ურთიერთობა ცვლადებთან“. რაც შემდეგში მოიაზრება. ნებისმიერი ცვლადი რომელიც აღწერილია სკრიპტში ფუნქციისათვის წარმოადგენს გლობალური ტიპის ცვლადს. ფუნქციიდან შესაძლებელია მის გარეთ მდებარე ცვლადთან წვდომა და მისი ცვლილება. მაგ:

```
var k = 25;

function test() {
        k  += 11     // k = k + 11
}
```

```
alert( k ); //  36
```

ფუნქციას გააჩნია ასევე ლოკალური ტიპის ცვლადები. რომელიც არ სცდება არსებული ფუნქციის ხილვადობის არეალს - მის ტანს. არსებული ცვლადის დეკლარირება ხდება უშუალოდ ფუნქციის ტანში და ის ხილვადია არსებული ფუნქციის შიგთავსში ასევე სხვა ფუნქციებში რომლებიც გამოძახებულია მიმდინარე ფუნქციის ბლოკში {...}, მაგრამ არამც და არამც ძირითადი ფუნქციის გარეთ.

```
function test() {
          var k = 25;
         k  += 11     // k = k + 11
}

alert( k ); //  k is not defined
```

შემთხვევა როცა ერთი და იგივე სახელით დეკლარირებულია ცვლადი ფუნქციის გარეთ და ფუნქციის შიგნით განიხილება შემდეგნაირად. გლობალური ცვლადი k რჩება გლობალურად და ის ფუნქციიდან არ განიცდის ცვლილებას გამომდინარე იქიდან, რომ არსებულ ფუნქციას თავისი ლოკალური k ვლადი გააჩნია, სწორედ მასი ცვლილება ხდება ფუნქციის შიდა გამოთვლის დროს.

```
var k = 25;

function test() {
          var k = 11;
         k  += 11     // k = k + 11

alert( k );
}
test()                  //  22

alert( k ); //  25
```

არსებობს ფუნქციის შექმნის კიდევ ერთი გზა, რომელიც ძალიან იშვიათად გამოიყენება მაგრამ ფუნქციის დეკლალირების სრულყოფილი წარმოდგენისთვის მასაც შევეხოთ.

```
var showMessage =  new Function()
```

არსებული სახით ფუნქცია იღებს და ამუშავებს მონაცემებს string_ის სახით, ძირითადად გამოიყენება მოკლე კოდისთვის

```
var sum = new Function('a,b', ' return a+b; ');

var result = sum(1, 2);
alert( result );          // 3
```

## Javascript Events

ვებ გვერდების დინამიურობა ძირითადად რაიმე ტიპი ხდომილებებზე/ქმედებაზეა დამოკიდებული. ხშირია, როდესაც დოკუმენტი ან მისი რომელიმე ელემენტი გარკვეულ ქმედებაზე განიცდის რაღაცა სახის ცვლილებას. მაგალითისათვის ვებ ბრაუზერი ახორციელებდეს ქმედებას, როდესაც ტვირთავს დოკუმენტს, მომხმარებელი მიერ მაუსის მიტანისას მოქმედებს ჰიპერბმულზე, ასევე ფორმის ღილაკზე ახორციელებს კლიკს. Javascript_ით შესაძლებელია წინასწარ განისაზღვროს კონკრეტული ქმედება, რაიმე ფუნქციის ან სკრიპტის ფრაგმენტის მეშვეობით, რომელიც ამუშავდება მხოლოდ მომხმარებლის მიერ გამოხატული აქტივობის შემდეგ.

ზოგადად სხვადასხვა ტიპის მოქმედება უზრუნველყოფს სხვადასხვა ტიპის ქმედების/ხდომილების წარმოქმნას. მაგ.: ბმულზე ხელის მიტანა და მასზე უშუალო ზემოქმედება (კლიკი) ერთმანეთისაგან განსხვავებული პროცესებია, ისევე როგორც თუნდაც ფორმის ორი ერთნაირი ღილაკი Submit და Reset აზრობრივად ერთმანეთისაგან რადიკალურად განსხვავდება.

ბრაუზერების დახვეწამ დროთა განმავლობაში განავრცო Javascript ხდომილებები და დღეს დღეისობით დეველოპერებს საკმაოდ დიდი არჩევანი აქვთ . ცხრილ XIV მოცემულია იმ ძირითადი ხდომილებების ჩამონათვალი, რომელიც აქტიურად გამოყენებადია. ესენია მაუსის, კლავიატურის, ფორმის და ობიექტის ქმედებები. სრული სია შეგიძლიათ იხილოთ მითითებულ ბმულზე -

http://www.w3schools.com/jsref/dom_obj_event.asp

ცხრილი XIV

| ხდომილება | აღწერა | | ტეგები, რომელსაც მიესადაგება |
|---|---|---|---|
| onclick | მაუსი | ერთჯერადი კლიკი ( დაჭერა - აშვების პროცესი) | ფაქტობრივად ყველა html ელემენტი |
| ondblclick | | ორმაგი კლიკი | ფაქტობრივად ყველა html ელემენტი |
| onmousedown | | მაუსის ღილაკზე მხოლოდ დაწოლა ( აშვება არა ) | ფაქტობრივად ყველა html ელემენტი |

| | | | |
|---|---|---|---|
| onmouseup | | მაუსის ღილაკზე ხელის აშვება | ფაქტობრივად ყველა html ელემენტი |
| onmouseout | | მაუსის კურსორი ცდება ელემენტის საზღვრებს | ფაქტობრივად ყველა html ელემენტი |
| onmouseover | | მაუსის კურსორი შედის ელემენტის საზღვრებში | ფაქტობრივად ყველა html ელემენტი |
| onmousemove | | მაუსის კურსორი გადაადგილდება ტეგის არეალში | ფაქტობრივად ყველა html ელემენტი |
| onkeydown | **კლავიატურა** | კლავიატურის კლავიშზე ხელის დაწოლა | ფაქტობრივად ყველა html ელემენტი |
| onkeyup | | კლავიატურის კლავიშზე ხელის აღება | ფაქტობრივად ყველა html ელემენტი |
| onkeypress | | დაწოლილი და აშვებულია კლავიშზე ხელი | ფაქტობრივად ყველა html ელემენტი |
| onload | **ფრეიმები / ობიექტები** | დოკუმენტის ჩატვირთვა დასრულებულია | BODY , IMG |
| onerror | | შეფერხება წარმოქმნა სცენარის შესრულებისას | IMG |
| onabort | | სურათის ჩატვირთვის შეფერხება | IMG |
| onresize | | დოკუმენტის ზომის ცვლილება | WINDOW |
| onscroll | | ელემენტის დასკროლვისას | უმრავლესობა ტეგებზე |
| onunload | | ელემენტი ჩატვირთvის პროცესშია | ძირითადად BODY |
| onfocus | **ფორმები** | ფორმის მონიშვნა / კურსორის ჩაყენება | BUTTON, INPUT, LABEL, SELECT, TEXTAREA |
| onblur | | ფორმის ელემენტიდან ამოსვლა | |
| onchange | | ფორმის ველი არჩეულია | SELECT |
| onreset | | ფორმის განულება / ინფორმაციის ჩამოყრა | FORM |
| onsubmit | | ფორმის მონაცემების გაგზავნა | FORM |

| | | | |
|---|---|---|---|
| onselect | | ტექსტის არჩევა | INPUT,<br>SELECT |

ხდომილების გამოყენება შესაძლებელია  უშუალოდ ელემენტში მისი გაწერით ან სკრიპტიდან ელემენტზე მიმართვიანობით.

**დავალება**

ღილაკზე მაუსის დაკლიკვით ბლოკ id="txt" შეეცვალოს ფონი და გახდეს ის წითელი

```
 <div id = "txt"> ტექსტი  1  </div>
 <button onclick = "setBg()" > ღილაკი  </button>
<script>
function setBg(){
            document.getElementById("txt").style.backgroundColor="red";
}
</script>
```

შესაძლებელი იყო არსებული ფუნქცია ცალკე გატანის გარეშე უშუალოდ მეთოდ onclick_ში გაგვეწერა. ეს გზა შედარებით არაკომფორტულია დიდ სკრიპტთან მიმართებაში და ნაკლებად გამოიყენება, მაგრამ მისი უგულველყოფა არ იქნება სამართლიანი, რადგანაც მცირე კოდებში საკმაო გამოყენება აქვს.  კოდის მინიმიზაცია თვალსაჩინოა.

```
<div id = "txt"> ტექსტი  1  </div>
<button onclick = "document.getElementById('txt').style.backgroundColor='red' " > ღილაკი  </button>
```

არსებობს კიდევ ერთი გზა, როცა event_ის მიბმა უშუალოდ სკრიპტიდან ხორციელდება

```
<div id = "txt"> ტექსტი  1  </div>
 <button> ღილაკი  </button>
<script>
document.getElementsByTagName("button")[0].onclick = function(){
            document.getElementById("txt").style.backgroundColor="red";
}
</script>
```

ქმედებებთან / ხდომილებებთან ურთიერთბისას მნიშვნელოვანია this მნიშვნელობის არსის ცოდნა. ის გამოიყენება მიმდინარე ელემენტის გამოსახატად სა კომფორტულია ბევრ შემთხვევაში. რამდენიმე ელემენტის შემთხვევაში სათითაოდ მოგვიწევს მათი წვდომის (document.getElement…..) განსაზღვრა, თავის მარტივად დასაძვრენად გამოვიყენოთ this

**დავალება:**

ბლოკებზე ხელის დაწოლისას გამოსახოს ეკრანზე არსებულ ბლოკში არსებული ინფორმაცია

```
<div onclick = "getTxt(this)" > ტექსტი  1  </div>
<div onclick = "getTxt(this)" > ტექსტი 2  </div>
<div onclick = "getTxt(this)" > ტექსტი  3  </div>

<script>
function getTxt(t){
        alert(t.innerHTML);
}
</script>
```

მოცემულ კოდში this არგუმენტის არცოდნის შემთხვევაში მოგწევდათ კოდის შემდეგნაირად გაშლა

```
<div onclick = "getTxt1()"  id = "dv1"> ტექსტი  1  </div>
<div onclick = "getTxt2()" id = "dv2"> ტექსტი 2  </div>
<div onclick = "getTxt3()" id = "dv3" > ტექსტი  3  </div>

<script>
function getTxt1(){
        alert(document.getElementById("dv1").innerHTML);
}
function getTxt(2){
                alert(document.getElementById("dv2").innerHTML);

}
function getTxt(3){
                alert(document.getElementById("dv3").innerHTML);
}
</script>
```

ან შესაძლოა სხვა ინტერპრეტაცია ჰქონოდა სკრიპტის ვრცელ ვარიანტს

კონკრეტული ხდომილების მიბმა უშუალოდ დოკუმენტზე შესაძლებელია html კოდი ხელშეუხებლად, უშუალოდ სკრიპტიდან შემდეგი სახით:

მიმართავთ კონკრეტულ ელემენტს და ახდენთ მასზე მოქმედების აქტივაციას. შესაძლებელია მიენიჭოს, როგორც ანონიმური ტიპის ფუნქცია ( გზა I ), ასევე მოხდეს ფუნქციის დეკლალირება და მხოლოდ სურვილისამებრ გამოყენება( გზა II ). გასათვალისწინებელია - ფუნქცია, რომელიც აღიწერება სტანდარტულად მისი მინიჭება უნდა მოხდეს ფრჩხილების გარეშე. მაგ.: `getTxt` და არა `getTxt()` , რადგანაც ფუნქციის გაწერა ფრჩხილებთან ერთად მის ავტომატურ გამოძახებას ნიშნავს, ხოლო მოცემულ მომენტში ფუნქცია უნდა გააქტიურდეს რაიმე ელემენტზე მოქმედების შედეგად.

მაგალითისათვის იხილეთ მოცემული სკრიპტი:

```
<div id = "dv1"> ტექსტი 1 </div>
<div id = "dv2"> ტექსტი 2 </div>
<div id = "dv3" > ტექსტი 3 </div>

<script>
გზა I

document.getElementById("dv1").onclick = function(){
                        alert(document.getElementById("dv2").innerHTML);
}

გზა II

document.getElementById("dv1").onclick = getTxt;  // მოცემულ შემთხვევაში ფუნქციას ფრჩხილები () არ ჭირდება;

function getTxt(){
                        alert(document.getElementById("dv2").innerHTML);

}
    </script>
```

*კითხვები და პრაქტიკული დავალებები თვითშეფასებისათვის*

1. ჩამოთვალეთ javascript_ის გამოტანის ოპერატორები;
2. გაიხსენეთ ცვლადის ტიპები .
3. javascript_ის რეზერვირებული სიტყვები და მათი მიმართება ცვლადის სახელებთან;
4. ჩამოაყალიბეთ ცვლადის გამოცხადების გზები;
5. არითმეტიკული მოქმედები javascript_ში;
6. ინკრემენტისა და დეკრიმენტის დანიშნულება;
7. ჩამოთვალეთ შედარების ოპერატორები;
8. switch-case კონსტრუქციის დანიშნულება;
9. რა არის ციკლი და რა სტილით მუშაობს იგი?
10. ციკლის მეშვეობით ეკრანზე გამოსახეთ 1 დან 20 მდე ყველა ლუწი რიცხვი.
11. ციკლის მეშვეობით გამოსახეთ ტექსტში არსებული ხმოვნების რაოდენობა;
12. შექმენით რიცხვითი მასივი და მოახდინეთ მისი რიცხვთა ჯამის ეკრანზე გამოტანა
13. იპოვეთ რიცხვითი მასივის მაქსიმალური და მინიმალური მნიშვნელობები;
14. მოახდინეთ ტექტური ინფორმაციით შედგენილი მასივის სორტირება;
15. ტაიმერის ფუნქციების გამოყენებით მოახდინეთ საიტზე საათის ასახვა;
16. შექმენით საიტზე ტაიმერი რომლის დროის გასვლის შემდგომ საიტი გადამისამართდება ავტომატურ რეჟიმში სხვა საიტზე - მაგ. http://vet.ge;
17. შექმენით ფუნქცია რომელიც გამოსახავს მისი არგუმენტის კუბურ ფესვს;
18. შექმენით ფორმის 2 ველი და ღილაკზე მაუსის დაწკაპებისას მოახდინეთ მათში შეყვენილი მნიშვნელობების დაჯამება;
19. შექმენით ფერადი ბლოკი რომელზე დაწკაპებისასაც ის შეიცვლის ფერს;
20. კონკრეტულ ბლოკზე ხელის მიტანისას ის უნდა გაქრეს და 5 წამში ისევ გამოჩნდეს;

## გამოყენებული ლიტერატურა


1. ე. ჩიკაშუა. კომპიუტერული გრაფიკული სისტემა Adobe Photoshop. საქართველოს უნივერსიტეტი. თბილისი, 2013.
2. Кэтрин Айсманн. МАСКИ И КОМПОЗИЦИЯ в PHOTOSHOP. Издательский дом "Вильямс", 2007.
3. (Виктор Родионов) Родионов В. И. — Подготовка электронных публикаций в InDesign CS6. 2013
4. http://fototips.ru/obrabotka/sloi-v-fotoshope-i-rezhimy-nalozheniya/
5. http://rugraphics.ru/photoshop/color-lookup-v-photoshop-cs6
6. https://helpx.adobe.com/illustrator/topics-cs6.html. Learn & Support / InDesign Help (ილუსტრატორის მხარდამჭერი ოფიციალური საიტი). Copyright © 2015 Adobe Systems Incorporated. All rights reserved.
7. http://hronofag.ru/2012/01/pathfinder-illustrator/