

HTML5 შესაძლებლობები ლაშა იაშვილი



2023

ლაშა იაშვილი

HTML 5-ის შესაძლებლობები

საგამომცემლო სახლი

„ტექნიკური უნივერსიტეტი“

საქართველოს ტექნიკური უნივერსიტეტი

ლაშა იაშვილი

HTML 5-ის შესაძლებლობები



რეკომენდებულია საქართველოს
ტექნიკური უნივერსიტეტის
სარედაქციო-საგამომცემლო საბჭოს
მიერ. 07.04.2023, ოქმი №1

თბილისი

2023

წიგნში განხილულია HTML5 WEB ტექნოლოგიების ყველა საჭირო ელემენტი და ატრიბუტი. ახალი HTML5 ვერსია წინა ვერსიისგან განსხვავდება, იგი მეორე გამოცემაა, სადაც ჩაწერილია თანამედროვე კოდები, განახლებულია ბმულები და, წინა ვერსიასთან ერთად, მოიცავს ვებტექნოლოგიების ისეთ ენებს, როგორიცაა CSS და JAVASCRIPT. მეორე გამოცემაში კვლავ დარჩა WEB ტექნოლოგიების SVG და CANVAS გრაფიკული ელემენტები, ასევე HTML API ტექნოლოგიები.

წიგნი განკუთვნილია ყველა აუდიტორიის წარმომადგენლისათვის, ვისაც WEB დაპროგრამება აინტერესებს.

რეცენზენტები: საქართველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის პროფესორი ნონა ოთხოზორია,

საქართველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის ასოცირებული პროფესორი ლევან ჯიქიძე

© საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“, 2023

ISBN 978-9941-28-966-8

<http://www.gtu.ge>



ყველა უფლება დაცულია. ამ წიგნის არც ერთი ნაწილის (იქნება ეს ტექსტი, ფოტო, ილუსტრაცია თუ სხვა) გამოყენება არანაირი ფორმით და საშუალებით (იქნება ეს ელექტრონული თუ მექანიკური) არ შეიძლება გამომცემლის წერილობითი ნებართვის გარეშე.

საავტორო უფლებების დარღვევა ისჯება კანონით.

წიგნში მოყვანილი ფაქტების სიზუსტეზე პასუხისმგებელია ავტორი/ავტორები.

ავტორის/ავტორთა პოზიციას შეიძლება არ ემთხვეოდეს საგამომცემლო სახლის პოზიციას.

სარჩევი

შესავალი რა არის HTML?	9
თავი I ტეგები და ატრიბუტები	12
<a> 	13
href.....	14
<abbr>.....	14
.....	15
<head>.....	15
<meta>	15
<style>	18
<base>	18
<div>.....	19
<nav>.....	29
<address>	31
<table>	33
<cite>.....	35
.....	36
ტექსტის ფორმატირება	37
<blockquote>.....	38
<bdo>.....	40
<time>	41
<area>	41
shape.....	42
coords.....	42
download.....	45
hreflang	45
rel	46
<link>.....	47
გლობალური ატრიბუტები	47
რა არის DOM?	48
Accesskey.....	49
data-*	53
მოვლენები HTML 5-ში	56
lang.....	58
spellcheck	60
style	61
tabindex.....	62

<i>title</i>	63
<i>Translate</i>	63
<i>HTML ფორმები</i>	63
<i>action</i> ატრიბუტი.....	67
<i>method</i> ატრიბუტი	68
<i>name</i> ატრიბუტი.....	68
<i>enctype</i> ატრიბუტი.....	69
<i>autofocus</i>	75
<i>disabled</i>	77
<i>form</i>	78
<i>maxlength</i>	79
<i>placeholder</i>	80
<i>readonly</i>	81
<i>required</i>	82
<i>wrap</i>	83
<i><option></i>	84
<i><button></i>	86
<i><keygen></i>	87
<i><output></i>	89
<i>ფაილების მონიშვნა</i>	94
<i>MIME ტიპი</i>	96
<i><progress></i>	104
<i><meter></i>	104
<i>DISABLED და READ-ONLY</i>	105
<i>HTML 5 მულტიმედია</i>	106
<i><video></i>	107
<i><source></i>	108
<i><track></i>	111
<i>default</i>	112
<i>kind</i>	112
<i>srclang</i>	112
<i><audio></i>	112
<i><object></i>	113
<i><param></i>	113
<i><iframe></i>	114
<i>თავი II CSS მიმოხილვა</i>	117
<i>CSS სინტაქსი</i>	119

ტიპობრივი ან ელემენტის სელექტორი.....	119
უნივერსალური სელექტორი.....	119
დესცენდენტის სელექტორი.....	121
კლასის სელექტორი.....	122
ID სელექტორი.....	124
შვილობილი სელექტორი.....	125
ატრიბუტის სელექტორები.....	126
დაჯგუფებული სელექტორები.....	127
CSS შეტანის მეთოდები	128
CSS საზომი ერთეულები	128
Box model.....	130
სხვა სიდიდეები	132
backgrounds.....	132
background-repeat.....	135
background-attachment	136
background-position	136
background-blend-mode.....	137
ბრაუზერების სპეციფიკა	138
border.....	139
border-radius.....	142
border-image.....	144
Border-collapse	145
Box-shadow	145
text-shadow.....	146
Outline.....	147
Overflow	148
column.....	149
გაფართოებები	151
align-content.....	152
align-items.....	152
flex.....	152
flex-basis.....	154
flex-direcion.....	155
flex-flow.....	155
flex-grow	156
Flex-shrink	157
Flex-wrap.....	157

<i>font</i>	157
<i>font-family</i>	159
<i>font-size</i>	159
<i>font-size-adjust</i>	159
<i>font-stretch</i>	160
<i>font-style</i>	160
<i>font-variant</i>	161
<i>font-weight</i>	161
<i>@font-face</i>	161
<i>text</i>	162
<i>Margin</i>	168
<i>Padding</i>	169
<i>position</i>	170
<i>clip</i>	171
<i>cursor</i>	172
<i>z-index</i>	173
<i>Links</i>	174
<i>list</i>	174
<i>Gradients</i>	180
<i>display</i>	185
კომბინატორები	187
ფსევდო კლასები / Pseudo-classes	188
ფსევდო ელემენტები.....	191
ნავიგაცია	193
<i>transforms</i>	199
<i>transform-origin</i>	202
<i>transform-style</i>	204
<i>backface-visibility</i>	208
<i>perspective</i>	209
3D Transform მეთოდები	210
წესები/Rules	211
<i>transitions</i>	216
<i>transition-delay</i>	218
<i>transition-duration</i>	218
<i>inline-block</i>	218
<i>transition-property</i>	219
<i>transition-timing-function</i>	219

animations.....	225
თავი III ჯავასკრიპტი	228
ოპერატორები.....	229
შედარების ოპერატორები.....	229
ლოგიკური ოპერატორები	230
მინიჭების ოპერატორები	230
სტრიქონული ოპერატორი	230
კომენტარები.....	230
მმართველი სიმბოლოები	231
ჯავასკრიპტის საკვანძო სიტყვები.....	231
გამოტანა.....	232
ცვლადები	234
ფუნქცია /function	235
სინტაქსი.....	235
მმართველი ოპერატორები.....	236
if ოპერატორი.....	237
switch ოპერატორი.....	238
ციკლები	239
ობიექტები.....	241
data	243
მათემატიკა	246
მასივები.....	247
Javascript გლობალური თვისებები და ფუნქციები	251
ფუნქციები	252
DOM მიმოხილვა.....	253
HTML5 API თავი IV	259
გრაფიკული ელემენტები SVG	259
SVG	260
<rect>	262
<line>.....	263
Polylines	264
<path>	278
<text>	280
tspan	281
tref.....	281
ფილტრები.....	281
<CANVAS>	286

<i>Drag და Drop (ჩაგდება და გადაადგილება)</i>	293
<i>DataTransfer</i>	295
<i>DataTransfer.dropEffect</i>	295
<i>DataTransfer.effectAllowed</i>	295
<i>DataTransfer.files</i>	295
<i>DataTransfer.items</i> (წაკითხვის რეჟიმი)	295
<i>DataTransfer.types</i> (წაკითხვის რეჟიმი).....	295
<i>dataTransfer.setDragImage(element, x, y)</i>	295
<i>dataTransfer.getData</i> (ფორმატი).....	295
<i>dataTransfer.setData(format, data)</i>	295
<i>Geolocation</i> (ადგილმდებარეობა).....	296
<i>Local Storage</i>	300
<i>Application Cache</i>	302
<i>WEB WORK</i>	304
HTML SSE	306
EventSource ობიექტები.....	307
ლიტერატურა	308

შესავალი რა არის HTML?



HTML5 არის ტექნოლოგია, რომელიც გამოიყენება იმისათვის, რომ აიგოს და წარმოდგენილი იქნას საიტები. ახალი ვერსიის განახლება ოქტომბრის 2014 წელს მოხდა. ეს არის ჯერჯერობით საბოლოო და სრული მეხუთე ვერსია HTML-სა, რომელიც მსოფლიო ქსელის კონსორციუმის მიერ არის შემუშავებული (*World Wide Web Consortium - W3C*).

HTML	1991
HTML 2.0	1995
HTML3.2	1997
HTML4.01	1999
XHTML	2000
HTML5	2014
HTML5.1	2016
HTML5.2	2017
HTML5.3	DRAFT

დავიწყოთ მარტივად, წიგნთან მუშაობისას და ენის შესწავლისას, მკითხველს ვურჩევ გამოიყენოს პროგრამები როგორებიცაა:

✓ *Sublime Text*

✓ *Visual Studio Code*

სწავლისათვის გირჩევთ უბრალოდ *notepad*-ში აკრიფოთ სკრიპტები.

(კოდებს, რომელსაც ვწერეთ WEB-ტექნოლოგიებში მას სკრიპტები ჰქვია)

Scripts -(სკრიპტები) ეს არის კოდების სცენარი, რის მეშვეობითაც შესაძლებელია დაიწეროს ან შედგეს, ვებსაიტები სხვადასხვა ვებ დაპროგრამირების ენებით.

Notepad-ნებისმიერ *Windows*-ს მოყვება, როგორც წინა ვერსიებს ასევე *windows 10*-ს და *windows 11*-ს ხოლო მისი გამოძახება ძალიან მარტივია ვაწვებით *Start Screen*, კლავიატურაზე ანუ ვინდოუსის ლოგოს და ძეგნის ველში ვწერთ-*notepad*-ში, როდესაც სკრიპტს ავკრეფთ შესაბამის რედაქტორში, არ აქვს მნიშვნელობა *notepad*-ი იქნება თუ *Dreamweaver*-ი, მას შენახვის დროს მას გაფართოება უნდა მიუთითოთ *.html*.

HTML რომელიც ითარგმნება, როგორც **ჰიპერტექსტის მარკირების ენა** (*Hyper Text Markup Language*). ჰიპერტექსტი განიმარტება როგორც ჩვეულებრივ ტექსტზე უფრო მეტი ინფორმაციული და ფუნქციური მონაცემების შემცველი დოკუმენტი.

მაგალითი:

```
<!DOCTYPE html>
<html>
<body>
</body>
```

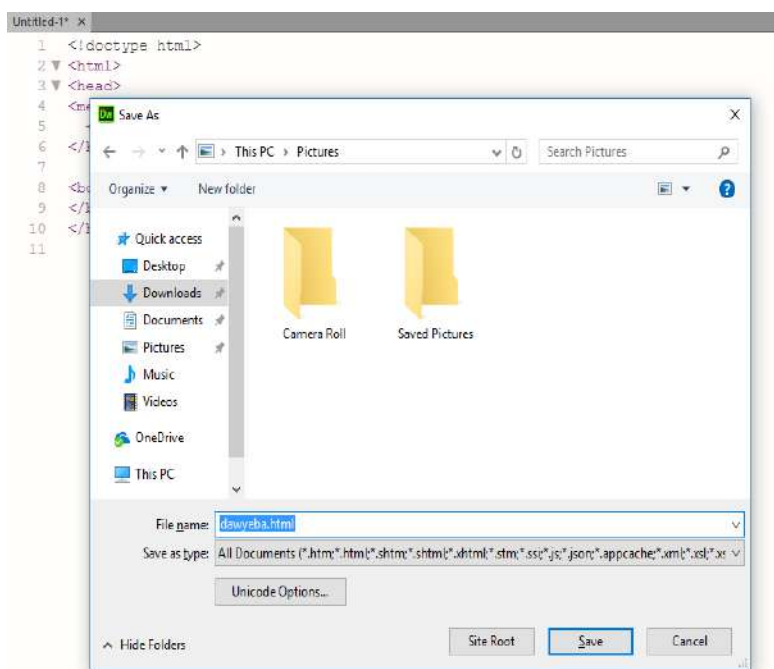
</html>

შევინახოთ რაიმე სახელით (მაგალითი თუ გსურთ, თქვენი სახელი დაარქვით,) მე პირობითად ავიღებ *dawyeba.html*-ს, აუცილებლად სახელს, რომ მივუწეროთ *.html* არ დაგავიწყდეთ.

ქართულად ასოებით აკრეფის დროს დეკოდირება უნდა მივუთითოთ:

UTF-8 ხოლო *Dreamweaver*-ში კი: **Modify > Page properties > Title/encoding > Encoding -> UTF-8** და თუ არ ქნა **Edit>Preferences>Fonts > Code View> to Sylfaen> Apply**-და ქართულად დაწერს ნებისმიერი ტექსტი, რომელიც <აქ> მოთავსებულია, ეწოდება ტეგი.

რედაქტორები არსებობს, რომელებსაც არ სჭირდება ენკოდირების მითითება და ავტომატურად ინახავს ქართული უნიკოდით, მაგალითად *Visual Studio Code*, რომლის გადმოწერა *microsoft* საიტიდან თავისუფლადაა შესაძლებელი, ნებისმიერი ოპერაციული სისტემისათვის.



<!DOCTYPE html>

<html>

<head>

<title>პირველი გვერდი</title>

</head>

<body>

<h1>ეს არის სათაური</h1>

<h2>ეს არის სათაური</h2>

<h3>ეს არის სათაური</h3>

<h4>ეს არის სათაური</h4>

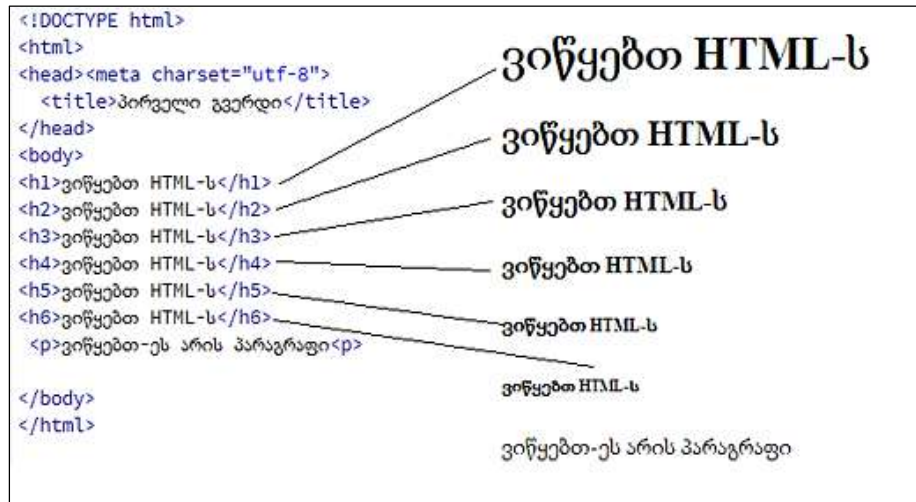
<h5>ეს არის სათაური</h5>

<h6>ეს არის სათაური </h6>

<p>ვიწყებთ-ეს არის პარაგრაფი<p>

</body>

</html>



სურ. 1.2

DOCTYPE განსაზღვრავს HTML დოკუმენტის ტიპს, მისი წერის სტრუქტურა ნაჩვენებია სურ.1.2

- ✎ ტეგის ტიპი <html> და </html> აღიწერება HTML დოკუმენტს.
- ✎ ტეგები <head> და </head> დოკუმენტის შესახებ გვაწვდის ინფორმაციას.
- ✎ <title> და </title> გვაწვდის ინფორმაციას სათაურის შესახებ.
- ✎ <body> და </body> შინაარსობრივად აღწერს გვერდს.
- ✎ <h1> დან <h6> ამდე განისაზღვრება სათაურები, <h1>-ი არის ყველაზე დიდი ზომის, ხოლო <h6> ანალოგიურად ყველაზე პატარა ზომის სათაური.

HTML 5 ში არის სემანტიკური ახალი ტეგები, როგორიცაა მაგალითად <header>, <footer>, <article> და <section> რომელმაც გაამარტივა სკრიპტის წერა და საბოლოოდ უფრო მოქნილი გახადა ვებგვერდები, ვიდრე 4.01-ის შემთხვევაში, ასევე უნდა აღინიშნოს რომ დაემატა გრაფიკული ელემენტები <svg> და <canvas>, ასევე მულტიმედიის ელემენტები, როგორიცაა <audio> და <video> ტეგები.

HTML 5-ში დიდ ინტერესს იწვევს API-ები (**Application Programming Interfaces**- აპლიკაციის პროგრამული ინტერეფეისი), საკითხები, რომლებსაც შევხებით:

HTML GeoLocation
HTML Drag and
Drop HTML Local
Storage
HTML Application Cache
HTML Web Workers
HTML SSE

თავი I ტეგები და ატრიბუტები

HTML-ში დოკუმენტის შექმნა იწყება ტეგით `<!DOCTYPE html>` ამით ვუბნებით ბრაუზერს, რომ დეკლარირება ხდება HTML-ი დოკუმენტის:

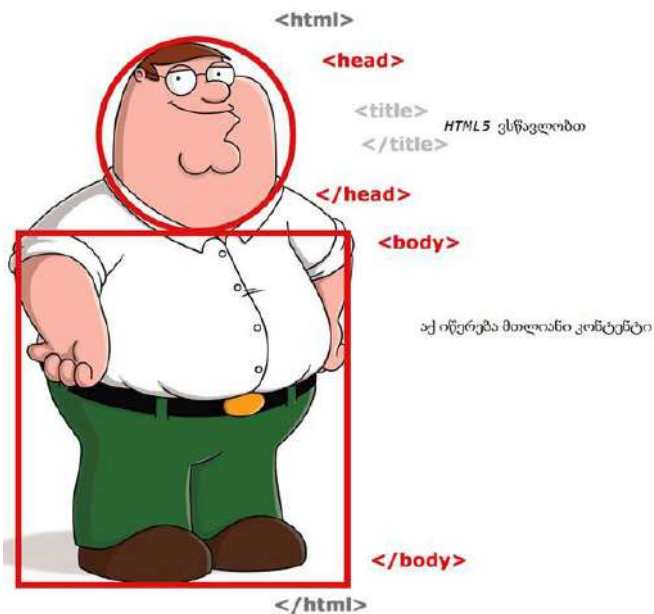
```
<!DOCTYPE html>  
<html>  
<body>
```

მოთავსებულ კონტენტს შეუძლია javascripts, css და სხვა.

```
</body>  
</html>
```

მაგრამ ამის შესახებ ქვემოთ ვისუბრებთ:

```
<!DOCTYPE html>  
<html>  
<head>  
<title>  
ვიწყებთ  
</title>  
</head>  
<meta charset="UTF-8">  
<body>  
ტექსტი  
</body>  
</html>
```



<!DOCTYPE html>

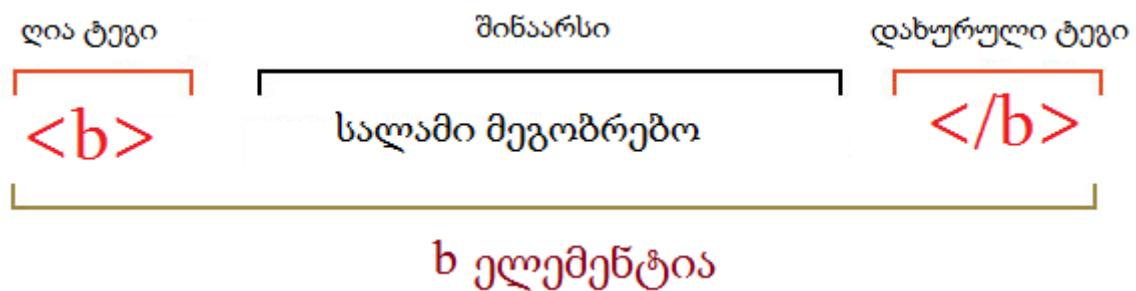
ეხმარება ბრაუზერს, რომ სწორად წარმოადგინოს ეკრანზე ინფორმაცია. არსებობს სხვადასხვა ტიპის აღწერა, რომელიც ქვემოთაა მოცემული:

- ✓ <!DOCTYPE html>
- ✓ <!DOCTYPE HTML>
- ✓ <!doctype html>
- ✓ <!Doctype Html>

ყველა ზემოთ მოყვანილი მათგანი დასაშვებია და არ წარმოადგენს ბრაუზერისთვის ან უშუალოდ სკრიპტისათვის შეცდომას.

გადავიდეთ უშუალოდ ტეგებზე, როგორც ზემოთ აღვნიშნე, ამათ შორის მოთავსებულ < სიტყვას> ტეგი ეწოდება.

მაინც რა არის ელემენტი? ელემენტი არის ის, რაც იწერება გახსნილ და დახურულ ტეგში, მაგალითი:



**<a> **

ამ ტეგის მეშვეობით შესაძლებელია გადასვლა ერთი ბმულიდან (ლინკიდან) მეორე ბმულზე (ლინკზე)

მაგალითი:

<a href="<http://gtu.ge>"> დაკლიკება გამოჩნდება „დაკლიკება“ დაჭერის შემდეგ გადავა მითითებულ საიტზე.

სანამ ტეგი დაიხურება, რაიმე ტექსტი უნდა ჩავწეროთ, რომ გავიგოთ, ლინკი სად იმყოფება

მაგალითი 1.1

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> სალამი ყველას </title>
</head>
<body>
```

```
<a href="http://gtu.ge">საიტი</a>
</body>
</html>
```

href

ეს გახლავთ დამხმარე ატრიბუტი `<a>` ტეგის, რომელშიც აღიწერება ლინკები, ხოლო მისი სინტაქსი ასე გამოიყურება `<a href="ლინკი">`

<abbr>

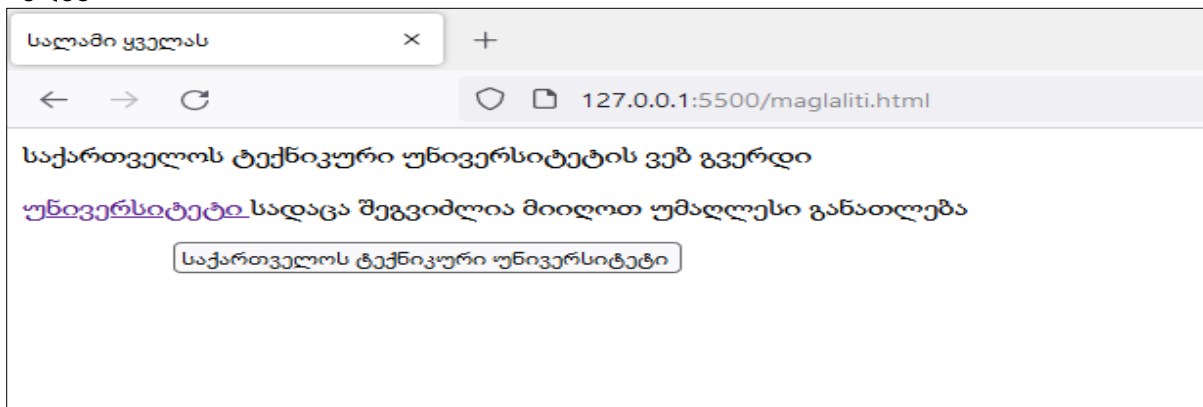
მოცემული ტეგი განსაზღვრავს აბრევიატურას ინფორმაციას რომელსაც საიტზე განვთავსებთ, ან სტატიაში, ავიღოთ მაგალითი :

მაგალითი 1.2

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>სალამი ყველას</title>
</head>
<body>
საქართველოს ტექნიკური უნივერსიტეტის ვებგვერდი
<p>
<a href="http://gtu.ge">
<abbr title="საქართველოს ტექნიკური უნივერსიტეტი " > უნივერსიტეტი
</abbr> </a>
სადაცა შეგვიძლია მიიღოთ უმაღლესი განათლება </p>
</body>
</html>
```

როდესაც ლინკთან მიიტანთ კურსორს, გამოჩნდება წარწერა „საქართველოს ტექნიკური უნივერსიტეტი“

შედეგი



სურათის ტეგი, სადაც შესაძლებელია სურათების ჩასმა. მასში აუცილებლად მოთავსებული არის ატრიბუტი *src*, რომელიც აღწერს, სად იმყოფება სურათი, ის ერთგვარად წარმოადგენს სურათის ლინკს და ზუსტად ეს გახლავთ ატრიბუტი. ქვემოთ მოცემულ *img* ტეგში ასევე გვხვდება სურათის სიგანის-*width* და სიმაღლის-*height* ატრიბუტები.

მაგალითი:

```

```

ალბათ, გასაგებია, რომ 140 და 342 მისი ზომებია, საერთოდ კი ატრიბუტის სინტაქსი ასე გამოიყურება:

<ტეგი სახელი="ცვლადი">

მოცემულ ტეგს აქვს რამდენიმე ატრიბუტი, რომელსაც HTML5 ის მხარდაჭერა აქვს, HTML4.01-ისგან განსხვავებით, HTML4.01-ში იყო და: *align*, *border*, *hspace*, *vspace* ხოლო დარჩენილი ატრიბუტები გახლავთ:

➤ *alt*- ამ ატრიბუტში იწერება ტექსტი, რომელიც შემდეგ არ ჩნდება საიტზე:

➤ **

მის აზრი ის არის რომ ეს ატრიბუტი აღწერს ინფორმაციას სურათის შესახებ, ჩვენ შეგვიძლია საერთოდ არ მიუთითოთ ეს ატრიბუტი და ისე დავეწეროთ ** ტეგი

➤ *height* და *width*- აღწერს სურათის სიმაღლეს და სიგანეს, *height="342"* ხოლო 342 არის პიქსელი, ე.ი. სიმაღლისა და სიგანის ზომის ერთეულად აღებულია პიქსელები.

➤ *ismap*-ამ ატრიბუტის გამოყენება ხდება იშვიათად, ძირითადად სურათზე დაკლიკების შედეგად წარმოადგენს ამ სურათის კოორდინატებს.

➤ *Usemap*-ატრიბუტი მონათასავე ატრიბუტია *<map>* ელემენტის, რომელიც უკავშირდება # სიმბოლოთი, ის მისი სინტაქსი ასე გამოიყურება:

```
<img usemap="#saxeli">
```

<head>

head ელემენტი მოიცავს ინფორმაციას დოკუმენტის შესახებ და მასში შესაძლებელია აღიწეროს *meta* ინფორმაცია, *<style>*, *<base>*, *<link>*, *<script>*, *<noscript>* და *<title>*.

<meta>

რაც შეეხება *meta* ტეგს, იგი შეიცავს ინფორმაციას HTML დოკუმენტის შესახებ და არსად საიტზე არ ჩნდება, მისი მიზანი, აღწეროს საიტი, მიუთითოს დეკოდირება, თუ საკვანძო სიტყვები, მაგალითად, რა განსხვავება დეკოდირებას შორის *html 4.01* თან შედარებით:

HTML 4.01:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

HTML 5:

```
<meta charset="UTF-8">
```

ასე რომ, *HTML 5* უფრო მეტად თანამედროვე და დახვეწილია და მარტივად აღწერს *HTML* - ის დოკუმენტს დღეისობით არსებობს *HTML 5.3* რომელიც ჯერჯერობით არ ჩაშვებულა.

ასევე *<meta>* ტეგში შესაძლებელია აღიწეროს საიტი, საკვანძო სიტყვები, ავტორის მითითება. მისი არსი ის არის, რომ საძიებო სისტემა მარტივად მოახდინოს საიტის სკანირება, როგორც შინაარსობრივად, ისე საკვანძო სიტყვებითაც, რომელსაც ვებ სამყაროში **SEO** ჰქვია.



SEO არის „*Search Engine Optimization*“, რომელის მეშვეობითაც შესაძლებელია საძიებო სისტემაში გამოჩნდეს ვებგვერდი. რაც შეეხება SEO-ს, მოვახდინოთ მოკლე გადახვევა, როდესაც ჩავწერთ რაიმე ტექსტს ძებნის ველში და დავაჭერთ „ენტერ“-ს, მივიღებ *WEB* შედეგების სიას, რითაც შეიცავენ დასმულ კითხვაზე პასუხს ან მიახლოებულია მასთან თემატურად. მომხმარებლები ჩვეულებრივ სტუმრობენ იმ საიტებს, რომლებიც ამ სიის თავში არის, რადგან მათ უფრო შესაბამება და დაისმის კითხვა: რატომაა ზოგი საიტის რეიტინგი უკეთესი ვიდრე სხვების? ეს ხდება ძლიერი მარკეტინგული ტექნოლოგიის მეშვეობით ანუ „საძიებო ოპტიმიზების“ მეშვეობით- (*SEO*). სეო არის ტექნიკა, რომელიც ეხმარება საძიებო სისტემას, მოძებნოს და წამოწიოს თქვენთვის საინტერესო საიტი საიტების სიაში სხვა საიტებზე წინ.

პირველ რიგში, საძიებო სისტემა ასკანირებს საიტს, რათა ნახოს, რა არის იქ. ეს ამოცანა სრულდება კომპიუტერული პროგრამის მიერ, რომელსაც ჰქვია „*crawler*“, ან „*spider*“ (ან „*Googlebot*“, როგორცაა გუგლის შემთხვევაში). *Spider* მიჰყვება ლინკებს ერთი გვერდიდან მეორეზე და აფიქსირებს ყველაფერს, რასაც პოულობს ძებნის პროცესში და პოულობს მრავალი გვერდს (მიახლოებით 20 მილიარდამდე).

მისთვის შეუძლებელია ესტუმროს საიტს ყოველდღიურად და ნახოს, რა ცვლილებები მოხდა ან უკვე არსებულმა გვერდმა ხომ არ შეიცვალა სახე. ზოგჯერ *crawler*-ს არ შეუძლია ესტუმროს თქვენს საიტს მთელი თვის განმავლობაში, როგორც უკვე ვახსენეთ *crawler*-ი არ არის ადამიანი და საძიებო სისტემები, რომლებიც ვერ ხედავენ გამოსახულებებს,

ფილმებს, ჩარჩოებს, პაროლით დაცულ გვერდებს და კატალოგებს. მას მერე, რაც გვერდი დასკანერდება, შემდეგი ნაბიჯია მისი შემადგენლობის ინდექსირება. ინდექსირებული გვერდი ინახება მონაცემების ბაზაში, საიდანაც მოგვიანებით შესაძლებელია მისი პოვნა. ინდექსაციის პროცესი არის სიტყვებისა და გამონათქვამების განსაზღვრა, რომლებიც ყველაზე კარგად აღწერს გვერდების მნიშვნელობას. ადამიანისათვის შეუძლებელია დიდი ინფორმაციის გადამუშავება, მაგრამ საძიებო სისტემა კარგად ართმევს ამ ამოცანას თავს. ზოგჯერ ისინი შეიძლება არ იგებენ გვერდის სწორ მნიშვნელობას, მაგრამ თუ დაეხმარები (*meta* ტეგის გამოყენებით) მისი ოპტიმიზებით, უფრო მარტივი იქნება, თქვენი გვერდის კლასიფიცირება მოხდეს სწორედ, რათა მიიღოს უფრო მაღალი რეიტინგი. როდესაც საძიებო მოთხოვნა მოდის, საძიებო სისტემა ამუშავებს მას, ის ადარებს საძიებო მოთხოვნაში ინდექსირებული გვერდებით მონაცემთა ბაზაში, თუ ერთზე მეტი გვერდი შეიცავს საძიებო სიას, საძიებო სისტემა იწყებს თითოეული გვერდის შესაბამისობის თვლას მის ინდექსში. არსებობს მრავალი შესაბამისობის გამომთვლელი ალგორითმი. თითოეულ ამ ალგორითმს აქვს განსხვავებული შეფარდებითი წონა ყველა საერთო ფაქტორისათვის, როგორც საკვანძო სიტყვისათვის, ბმულისათვის. ამიტომ სხვადასხვა საძიებო სისტემები იძლევა სხვადასხვა შედეგს, ერთი და იმავე საძიებო სიტყვაზე.

საიტი უფრო სწრაფად ახდენს ინდექსაციას, თუ გავაკეთებთ *sitemap*-ს, გავუწერთ მოკლე აღწერას თუ საკვანძო სიტყვებს *meta* ტეგებში და ასევე საიტს დავარეგისტრირებთ *google webmaster tool*-ში ან *bing webmaster tool*-ში ან სხვა საძიებო სისტემებში მარტივია მოხდეს ინფორმაციას საიტის შესახებ. რაც შეეხება საძიებო სისტემის მუშაობის პრინციპს, უნდა ითქვას, რომ ბოლო ნაბიჯი საძიებო სისტემის აქტივიზაციაა არის შედეგის პოვნა. ეს პრინციპულად სხვა არაფერია, თუ არაა ბრაუზერზე მისი ჩვენება.

დაუსრულებელი რაოდენობის ძიების რეზულტატები ხარისხდება უფრო შესაბამისი საიტებიდან ნაკლებად შესაბამის საიტებამდე, თუმცა ყველა საძიებო სისტემის ძირითადი პრინციპების მუშაობისა ერთი და იგივეა, სხვადასხვა საძიებო სისტემებისათვის სხვადასხვა ფაქტორებია მნიშვნელოვანი, იყო დრო, როდესაც *SEO* -ს სპეციალისტები ხუმრობდნენ, რომ *Bing*-ის ალგორითმი შექმნილია *Google*-ის საპირისპიროდ, თუმცა ამაში შეიძლება იყოს ჭეშმარიტები მარცვალიც, თუ შენ (თქვენ) გეგმავ დაიპყრო ერთერთი მათგანი, აუცილებელია ფრთხილად მოახდინო ოპტიმიზება.

საიტი უფრო სწრაფად ახდენს ინდექსაციას, თუ დავურთავთ *sitemap*-ს, ან მოკლე აღწერას თუ საკვანძო სიტყვებს *meta* ტეგებში და ასევე საიტს დავარეგისტრირებთ *google webmaster tool*-ში ან *bing webmaster tool*-ში ან სხვა საძიებო სისტემებში მარტივია მოხდეს ინფორმაციას საიტის შესახებ. რაც შეეხება საძიებო სისტემის მუშაობის პრინციპს, უნდა ითქვას, რომ ბოლო ნაბიჯი საძიებო სისტემის აქტივიზაციაა არის შედეგის პოვნა. ეს პრინციპულად სხვა არაფერია, თუ არაა ბრაუზერზე მისი ჩვენება.

დაუსრულებელი რაოდენობის ძიების რეზულტატები ხარისხდება უფრო შესაბამისი საიტებიდან ნაკლებად შესაბამის საიტებამდე, თუმცა ყველა საძიებო სისტემის ძირითადი პრინციპების მუშაობისა ერთი და იგივეა, სხვადასხვა საძიებო სისტემებისათვის სხვადასხვა ფაქტორია მნიშვნელოვანი, იყო დრო, როდესაც *SEO* -ს სპეციალისტები ხუმრობდნენ, რომ *Bing*-ის ალგორითმი შექმნილია *Google*-ის საპირისპიროდ, თუმცა ამაში შეიძლება იყოს ჭეშმარიტები მარცვალიც, თუ შენ (თქვენ) გეგმავ დაიპყრო ერთ ერთი მათგანი, აუცილებელია ფრთხილად მოახდინო ოპტიმიზება. არსებობს მრავალი მაგალითი საძიებო სისტემის განსხვავებისა. მაგ: *Yahoo!* და *Bing*,

საძიებო სიტყვას აქვს დიდი მნიშვნელობა, ხოლო Google-ისთვის ბმულებია ძალიან მნიშვნელოვანი. ასევე Google-ისთვის საიტები არის ღვინოსავით-რაც უფრო ძველია მით უკეთესი, იმ დროს როცა Yahoo-მ საერთოდ არ მიაჩნია უპირატესობა საიტებს. ამასთან უფრო მეტი დროა საჭირო, რომ საიტი შევიდეს Google-ის ტოპ საიტებში, ვიდრე Yahoo-ში.

<!-- კომენტარი--> ეს არის კომენტარი, რომელიც სკრიპტისათვის დამალულია.

<style>

აღწერს დოკუმენტის სტილს. ძირითადად სტილი სტატიკური თუ დინამიკური საიტებისა იწერება კასკადურ ენაში, რომელსაც **CSS** ჰქვია (**Cascading Style Sheets**), და მისი გაფართოება არის **.css**, სკრიპტშივე **head** ტეგში შიგნით მოვათავსოთ **<style>** ლინკი, სადაც აღიწერება საიტის დიზაინი, შიგნით შესაძლებელია ასევე ცალკე ფაილის შექმნა და **<link>**-ით დაკავაშირება შესაბამის მისამართზე, მაგალითი **<Link>** თვითონ გვეუბნება, რომ გადამისამართება ხდება.

CSS -ს ჩვენი კიდევ დავუბრუნდებით, ჯობს გადავიდეთ შემდეგ ტეგებზე, რაც შეეხება **rel** ატრიბუტს, ის დამხმარე ატრიბუტია, რომელიც ურთიერთქმედებს არსებულ დოკუმენტსა და მიბმულ დოკუმენტს შორის.

<base>

HTML არის ერთ-ერთი ტეგი, რომელსაც არ აქვს დამხურავი ტეგი, მისი ატრიბუტებში შედის **href** და **_target**, თვითონ ტეგის მიზანიაა, რომ განთავსდეს ლინკი, რომელიც შეიცვას სხვა ლინკებს, ის შეიძლება მხოლოდ ერთხელ განთავსდეს დოკუმენტში.

ელემენტები	სინტაქსი
_blank იხსნება ახალ ფანჯარაში	<base target="_blank">
_parent იხსნება იმავე გვერდზე	<base target="_parent">
_self იხსნება მშობლიურ გვერდზე	<base target="_self">
_top იხსნება სრულიად ახალ გვერდზე	<base target="_top">

<script> და **<noscript>** ეს ორი ტეგი ორიენტირებულია **javascript** ფაილების წასაკითხავად, ხოლო **<noscript>** თუ ბრაუზერს არ აქვს მისი მხარდაჭერა შეტყობინების გამოსატანად, მაგრამ თანამედროვე ყველა ბრაუზერს აქვს მისი მხარდაჭერა, ამიტომაც ხშირად აქტიურად არ გამოიყენება.

მაგალითი 1.3

```
<!DOCTYPE HTML>
<html>
<head>
<script>
```

```

    document.write("სალამი ყველას")
</script>
<- script>თქვენს ბრაუზერს არ აქვს JavaScript-ის მხარდაჭერა
</- script>
<p> ბრაუზერს, რომელსაც არ აქვს მხარდაჭერა - script ელემენტის
გამოჩნდება ის ტექსტი, რაც წერია ტეგებს შორის.</p>
</head>
<body>
</body>
</html>

```

ახლა უბრალო ვახდენთ ტეგის დემონსტრირებას, რაც შეეხება ტეგში ჩაწერილი სტრიქონს `document.write()`, მას მოგვიანებით განვიხილავთ.

<div>

`div` ტეგი HTML 4.01 -ის გან განსხვავებით *HTML5*-ში გახდა ერთ- ერთი ძირითადი მთავარი ტეგი, მასში აღიწერება ლინკები, მენიუს ტეგები, ინფორმაცია სხვადასხვა ობიექტზე, ასევე `div` ტეგი გამოიყენება სტილის აღსაწერად, რაც ბლოკების ანუ განლაგების ე.წ. „ლეიაუტების“ (*Layouts*) და ასევე კლასების აღსაწერად, მოკლედ ძალიან დიდი და ფართო დანიშნულება აქვს `div` ტეგს.

დავიწყოთ ძალიან მარტივი მაგალითით, შევქმნათ რაიმე ტექსტი, რომელზეც გამოყოფილია და გაფერადებულია სხვადასხვა სიტყვა, ასევე დამატებით გავეცნოთ ახალ ტეგებს, რომლებიც `div` ტეგის გარეშეც მშვენივრად მუშაობს, მაგალითის საფუძველზე ვნახოთ, `<p>` და `<div>` -ს შორის რა განსხვავებაა.

გამოვიყენოთ `<div>` და პრობლემაც მოგვარებულია, მაგრამ, როგორც ზემოთ აღვნიშნე,

`<div>`-ს გარდა, მასა დაემატა სხვა ფუნქციებიც, მოდი, შევქმნათ სტატიკური საიტი რომელშიც იქნება ქალაქები აღწერილი; თბილისი, თელავი და ბათუმი, ამისათვის დაგვჭირდება `<div class="">` ები, შემდეგ `<style>` ტეგებში მისი დიზაინის აღწერით.

სანამ დავუბრუნდებით CSS საფუძვლებს, სკრიპტში არსებულ ტეგებს ავხსნით:

სტატიკური არის საიტი, რომელიც მონაცემთა ბაზას არ იყენებს და ხელით ხდება მასში ინფორმაციის გათავსება, ხოლო **დინამიკური** საიტები იყენებს მონაცემთა ბაზას -*MySQL*

სანამ `div` ტეგებს დავუბრუნდებოდეთ კიდევ რაც უნდა ვიცოდეთ:

- ➔ `<b>` გასქელებული ტექსტი `</b>`
- ➔ `<i>` დახრილი ტექსტი `</i>`
- ➔ `<tt>`ნაბეჭდი ტექსტი`</tt>`
- ➔ `<u>` გახაზული ტექსტი `</u>`
- ➔ `<strike>`გადახაზული ტექსტი`</strike>`

- <s>გადახაზული ტექსტი</s>
- განსაზღვრავს სიას
- გადანომრის დროს

მაგალითი 1.4

```
<html>
<body>
    <b> გასქელებული ტექსტი </b>
    <i> დახრილი ტექსტი </i>
    <tt>ნაბეჭდი ტექსტი</tt>
    <u> გახაზული ტექსტი </u>
    <strike>გადახაზული ტექსტი</strike>
    <s>გადახაზული ტექსტი</s>
    <li> განსაზღვრავს სიას</li>
    <ol> გადანომრის დროს<ol>

<!-- <ol> გამოიყენება ნუმერაციის დროს, ხოლო <ul> ნუმერაციის გარეშე -->
<ol>

<li> </li>
<li> ჯავახიშვილი </li>
<li> ალასანია </li>
<ol>
<p> დაულაგებელია სია:</p>
<ul>საქართველოს ტექნიკური უნივერსიტეტი </ul>
<ul> ჯავახიშვილი </ul>
<ul> ალასანია </ul>
<li>
</ul>
</ol>
```

პარაგრაფი ანუ <p> მასში მოთავსებულ ტექსტებზეა პასუხისმგებელი და ამავე ტექსტის რეგულირება შესაძლებელია მასშივე მითითებული პოზიციით, რომელიც ასე გამოიყურება *Visual Studio Code*-ში.

- ✓ **left** ტექსტს მარცხენა კიდეზე ასწორებს.
- ✓ **right** ტექსტს მარჯვენა კიდეზე ასწორებს.
- ✓ **center** ტექსტს გვერდის ვერტიკალური შუახაზის სიმეტრიულად ასწორებს (გამოიყენება სათაურების კეთებისას).
- ✓ **justify** იგივეა, რაც **Left+right**, ანუ ტექსტს ორივე კიდეზე ასწორებს.

<body text="#1E90FF" bgColor="#CFCFCF"> -მოდის გავშიფროთ აქ, რაც წერია, ამით ვამბობთ, რომ **body text**-ი უნდა იყოს #1E90FF-ფერის ანუ მოცისფრო, ამ კოდს რომელიც დავწერე **hex color-ი ჰქვია** და ამის თაობაზე, ორიოდ სიტყვით, ცოტა ქვემოთ განვმარტავთ, რაც შეეხება **bgColor=""** ეს იგივეა, რაც **background color**-ანუ

უკანა ფონის ფერი-ნაცრისფერი, რა თქმა უნდა, თქვენ სხვა ფერების მითითებაც შეგიძლიათ და ჩაწეროთ სკრიპტში სიტყვიერად მსგავსი ფერი, (არა მარტო ფერები) საკითხი CSS-სის მეშვეობით წყდება.

`<body text="blue" bgcolor="grey">` მაგრამ ასე უფრო მუქ ფერებს მივიღებთ, მხოლოდ სკრიპტის მიხედვით „მოგესალმებით“ არის ლურჯი, ამიტომაც სკრიპტი მარტო მას გააღურჯებს, რაც შეეხება `<font color>`, აქ გასაგებია, რომელ ტექსტზეა ლაპარაკი „ჩემი პირველი ნაბიჯები HTML 5-ში“ იქნება წითლად.

`<p><font size="3" color="red">ტექსტი წითლად 3 ზომის</font></p>`

`<p><font size="2" color="blue">ტექსტი ლურჯად 2 ზომის</font></p>`

`<p><font face="acadnux" color="green">text mwvaned </font></p>`

რაც შეეხება ჩვენს `div` ტეგს, მისი მეშვეობითაც შესაძლებელია ფონტის რეგულირება, მაგ:

✦ `<div align="center">...</div>`

✦ `<div align="left">...</div>`

✦ `<div align="right">...</div>`

✦ `<div align="justify">...</div>`

შეიძლება შევამჩნიოთ კიდეც, რომ ტექსტის განლაგება *default*-ად *Left*-ზეა დაყენებული. ასევე საგულისხმოა, რომ დამხურავი ტეგი `</p>`-ს შემდეგ ავტომატურად გადადის ხაზზე, ანუ `<br>`-ს აკეთებს, მაშინ დაისმის კითხვა, ეს ფუნქცია ხელს, თუ გვიშლის?

br -იგივეა *break* ანუ გამოტოვება, საჭიროდ არ ჩავთვალოთ, ამისათვის დიდი ყურადღება დამეთმო, ერთი სტრიქონით გამოტოვება არის და მეტი არაფერი

გამოვიყენოთ `<div>` და პრობლემაც მოგვარებულია, მაგრამ, როგორც ზემოთ აღვნიშნე, `<div>`-ს გარდა, მასა დაემატა სხვა ფუნქციებიც, მოდი, შევქმნათ სტატიკური საიტი, რომელშიც იქნება ქალაქები აღწერილი; თბილისი, თელავი და ბათუმი, ამისათვის დაგვჭირდება `<div class="">` ები, შემდეგ `<style>` ტეგში მისი დიზაინის აღწერით. სანამ დავუბრუნდებით CSS საფუძვლებს, სკრიპტში არსებულ ტეგებს ავხსნით:

სტატიკური არის საიტი, რომელიც მონაცემთა ბაზას არ იყენებს და ხელით ხდება მაშინ ინფორმაციის განთავსება, ხოლო დინამიკური საიტები იყენებს მონაცემთა ბაზას -MySQL ან სხვ.

- `<i>` დახრილი ტექსტი `</i>`
- `<tt>`ნაბეჭდი ტექსტი`</tt>`
- `<u>` გახაზული ტექსტი `</u>`
- `<strike>`გადახაზული ტექსტი`</strike>`
- `<s>`გადახაზული ტექსტი`</s>`
- `<li>` განსაზღვრავს სიას`</li>`
- `<ol>` გადანომრის დროს`</ol>`

მაგალითი 1.4

```
<html>
<body>
  <b> გასქელებული ტექსტი </b>
  <i> დახრილი ტექსტი </i>
  <tt>ნაბეჭდი ტექსტი</tt>
  <u> გახაზული ტექსტი </u>
  <strike>გადახაზული ტექსტი</strike>
  <s>გადახაზული ტექსტი</s>
  <li> განსაზღვრავს სიას</li>
  <ol> გადანომრის დროს&ltol>
<!-- <ol> გამოიყენება ნუმერაციის დროს, ხოლო <ul> ნომერაციის გარეშე -->
<ol>

<li> </li>
<li> ჯავახიშვილი</li>
<li> ალასანია </li>
<ol>
<p> დაულაგებელია სია:</p>
<ul>საქართველოს ტექნიკური უნივერსიტეტი</ul>
<ul> ჯავახიშვილი</ul>
<ul> ალასანია</ul>
<li>
</ul>
</ol>
</body>
</html>
```

მაგალითი 1.5

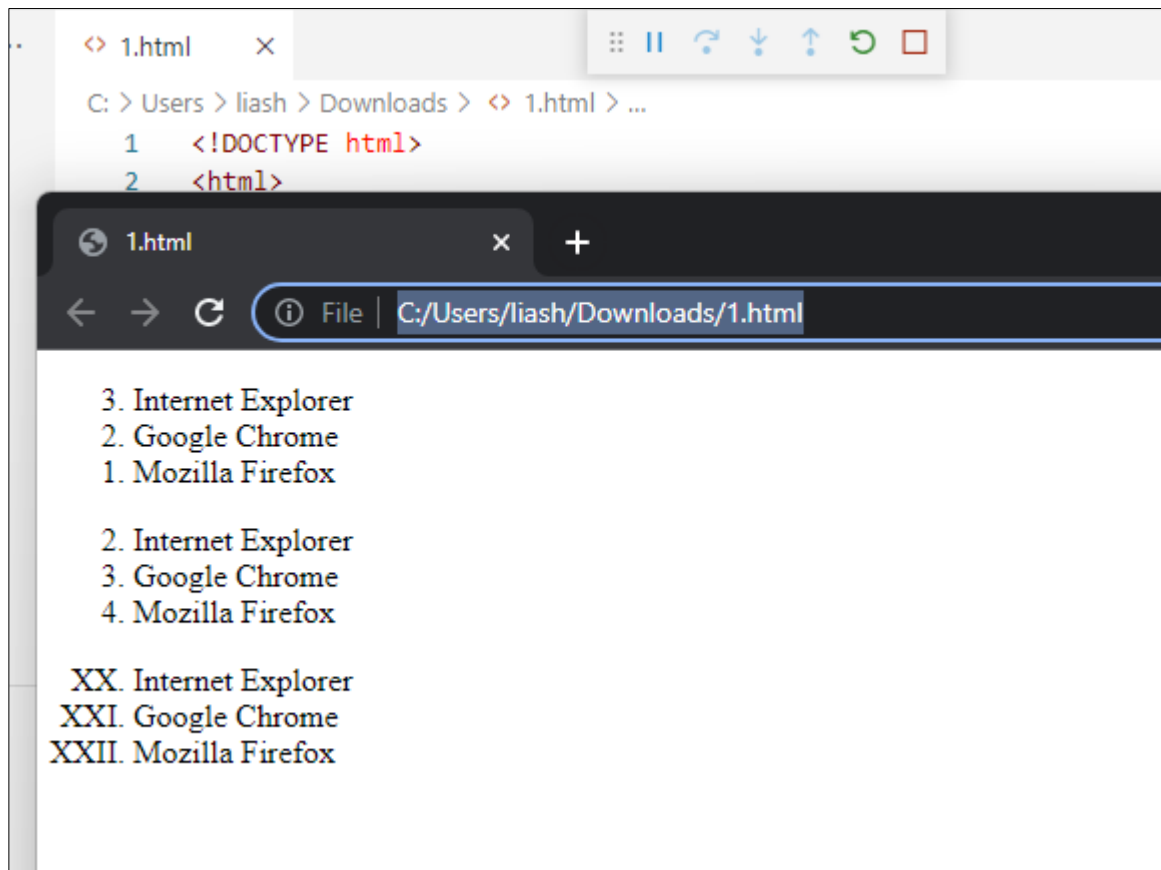
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<!--მისი სინტაქსია <ol reversed>, რომლის მხარდაჭერა მხოლოდ Internet
Explorer-ის ბრაუზერს 11-ის ჩათვლით არა აქვს-->
<ol reversed>
<li> Internet Explorer </li>
<li> Google Chrome </li>
<li> Mozilla Firefox </li>
</ol>
<!--სინტაქსი <ol start="number">
და იწყება მაგალითისათვის 2-დან და ატრიბუტია start="" -->
<ol start="2">
<li>Internet Explorer</li>
<li>Google Chrome</li>
```



```

<li>Mozilla Firefox</li>
</ol>
<!-- რომელიც ციფრებით გადანომვრა იწყება ნუმერაცია 20-დან გადანომრავს 20
დან და მისი ატრიბუტია type="" -->
<ol type="I" start="20">
  <li>Internet Explorer</li>
  <li>Google Chrome</li>
  <li>Mozilla Firefox</li>
</ol>
</body>
</html>

```



`div` ტეგის ატრიბუტია ერთ `class=""`, სადაც აღიწერება ჩვენთვის სასურველი სახელი, ასევე `<div id="">` აქვს კიდევ ერთი `id` ატრიბუტი, რომელიც `class`-სგან განსხვავებით უნიკალურია, შეიძლება გვქონდეს ერთი კლასი მრავალი ელემენტით და ასევე ბევრი ელემენტი ერთი კლასით, ხოლო თითოეულ ელემენტს შეიძლება ჰქონდეს ერთი `id` და თითოეულ გვერდს შეიძლება ჰქონდეს ერთი ელემენტი `id`-ით ხოლო რაც შეეხება `CSS`, მაში ჩაწერის ფორმა სხვადასხვანაირია. მაგალითი

```
<div id="header_id" class="header_class">ტექსტი</div>
```

```

#header_id {color:#fff}
.header_class {color:#000}

```

როგორც ვხედავთ, პრეფიქსი # გამოყენება id-ის, ხოლო წერტილი-კლასის შემთხვევაში, დაუბრუნდეთ ჩვენი საიტის შექმნას.

მისი სკრიპტი ასე გამოყურება:

მაგალითი 1.6

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
div.Tbilisi
{
    background-color: #e5e5e5;
    color:#000000;
    margin:20px;
    padding:20px;
text-align: justify;
}
div.Telavi
{
    background-color: #e5e5e5;
    color:#CC0000;
    margin:20px;
    padding:
        text-align: justify;
}
p{
color:blue;
    text-align: center;
}
h2
{
color:green;
}
</style>
</head>
<body>
<div class="Tbilisi">
<h2>თბილისი
</h2>
<p> თბილისი – საქართველოს დედაქალაქი; 1922–1936 წლებში ერთდროულად იყო
ამიერკავკასიის სფსრ დედაქალაქი;
</p>
</div>
<div class="Telavi">
<h2>კახეთი
</h2>
```

<p> თელავი—ქალაქი აღმოსავლეთ საქართველოში, გომბორის ქედის ჩრდილო-აღმოსავლეთი კალთის ძირას, ალაზნის ვაკეზე.

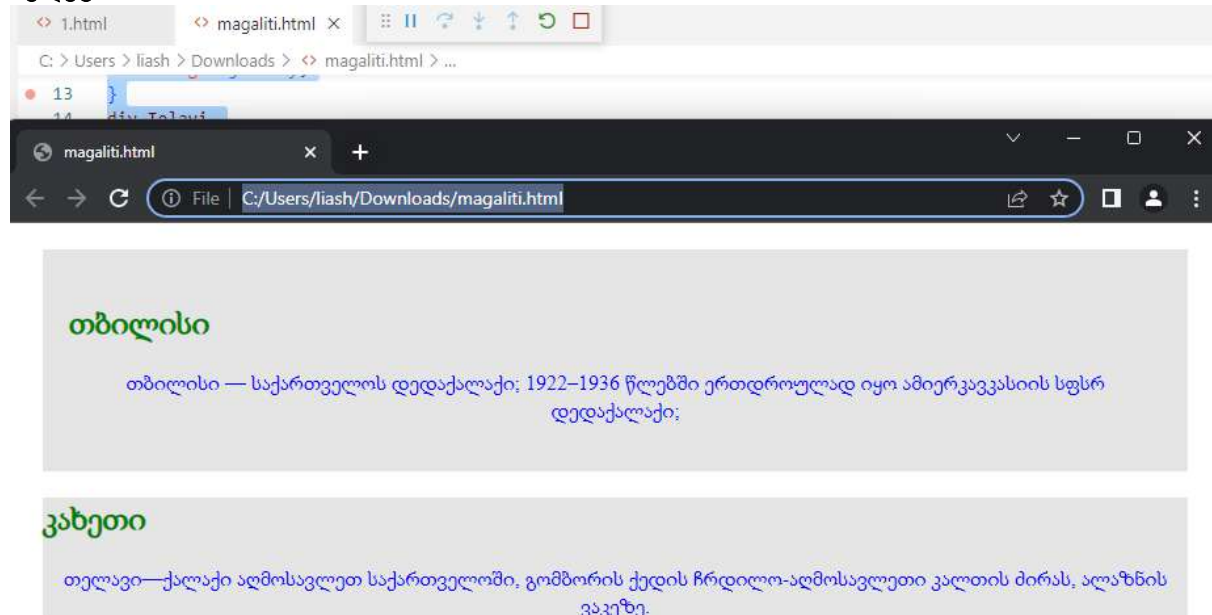
</p>

</div>

</body>

</html>

შედეგი



სტილსა და სემანტიკაში შედის შემდეგი სახის ტეგები <style>, <div>, , <header>, <footer>, <main>, <section>, <article>, <aside>, <details>, <dialog>, <summary> ამ ტეგებს ეტაპობრივად განვიხილავთ, რაც შეეხება სემანტიკას. სემანტიკა სწავლობს სიტყვებს და ფრაზებს ენაში, ზოგი ლიტერატორი და პროგრამისტის მტკიცებით <div> და , ტეგები არ აღწერენ არაინაირ კონტენტს, ამიტომაც არასემანტიკური არიან, ზოგნი კი პირიქით ამტკიცებს, ჩვენ არ დავიწყებთ ამ ტეგების ერთი ან მეორე მხარის

მტკიცებას, თუმცა, ჩვენი აზრით, ორივე ტეგი აღწერს დოკუმენტის კონტენტს და ამიტომაც შევიყვანეთ სემანტიკურობაში.

რაც შეეხება ფერებს, რომელიც ასე აქტიურად გამოიყენება **hex color**-ები ჰქვია, მაინც რა არის **hex color**-ი?

დავიწყოთ იქიდან, რომ ფერები ბუნებაშიც და ვებსადავროშიც მიიღება სამი ძირითადი ფერის ერთმანეთთან შერევით: „წითელი“, „მწვანე“, „ლურჯი“, ინგლისურად (**RGB**). ეს ის ძირითადი ფერებია, რის შერევითაც მილიონობითა და მილიარდობით ფერის მიღებაა შესაძლებელი. ვებსივრცეში ყოველ ფერს თავისი **HEX** კოდი აქვს, ისინი ძირითადად წარმოდგება „თექვსმეტობით“ მანქანურ კოდებში, თვალსაჩინოებისთვის ერთ სურათს გაჩვენებთ ბინარულ (**binary**), ათობით (**Decimal**) და თექვსმეტობის (**HEXdecimal**), რომ წარმოდგენა შეგეგმნათ, თუ რა არის **HEX** კოდი.

🔗 **HTML4** ჩაიწერებოდა:

HTML5 უნდა გამოვიყენოთ:

```
<p style="color: #800080">
```

ხოლო CSS ში უკვე ასევე შეგვიძლია გამოვიყენოთ RGB მაგალითი:

```
P
```

```
{ color: rgb(128,0,128);
```

```
}
```

```
ან P { color: #800080; }
```

Binary	Decimal	Hexadecimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10
10001	17	11
etc	etc	etc

ახლა კი უფრო ჩავუღრმავდეთ:

- ყველაზე დაბალი მნიშვნელობა, რაც კი შეიძლება ფერმა მიიღოს რიცხობრივად, ესაა 0, ანუ HEX- ში იგივეა, რაც 00 , ხოლო ყველაზე მაღალია 255, ანუ HEX- ში იგივე, რაც FF.
- HEX - ფერებისათვის დამახასიათებელია სამი წყვილი მთელი რიცხვი, რომელთა საწყისი გახლავთ #, ან შეიძლება შეგხვდეთ ისე, რომ დიეზი არ ჰქონდეს, მაგრამ არც ესაა შეცდომა, მთავარია, მთლიანობაში შედგებოდეს 6 სიმბოლოსაგან.
- HEX-ი თეთრი ფერის განმსაზღვრელია FFFFFFFF, ხოლო შავი ფერისა კი 000000 ფერებში ადვილად რომ გავერკვიოთ და მომავალში დაგვჭირდეს ვებგვერდების შექმნაში ან რედაქტირებაში მარტივად მიაკითხეთ სამიეზო სისტემას *google-* ში ჩაწერეთ *hex-color* და ნებისმიერი მათგანის გამოყენება შეგვიძლია, ან *HTML* ედიტორებს მოჰყვება თითქმის ყველა.

რაც შეეხება, თუ როგორ გამოყურება, ჩვენი სკრიპტი, CSS-ში აუცილებელია, რომ ცვლადი { ფრჩხილები იყოს } შესაბამისად ბლოკებში აღწერილი კოდი ფერი, „ცვლადს“

ეკუთვნის, ასევე აუცილებელია წერტილში ; რადგან ახალ სტრიქონზე გადასვლა მოხერხდეს.

```
div.Tbilisi {
  background-color: #e5e5e5;  — უკანა ფონი
  color:#000000; — ფონტის ფერი
  margin:20px; — საზღვრები
  padding:20px; —
  text-align: justify; — ტექსტის პოზიცია
}

div.Telavi {
  background-color: #e5e5e5;
  color:#CC0000;
  margin:20px;
  padding:
  text-align: justify;
}

p{
  color:blue;
  text-align: center;
}

h2{
  color:green;
}

</style>
```

ანალოგიურად აქაც იგივეა

ყველგან, სადაც <p> არის გამოიტანს
ლურჯად და ტექსტი შუაში იქნება

h2 აღწერილი ტექსტი იქნება მწვანე

განვიხილოთ HTML5-ის ახალი ტეგები <header>, არ შეგეშალოთ <head>-ში, <footer>, <nav>, <aside>, <section> მოცემულია სტრუქტურა HTML 5-ის.

head ტეგისაგან განსხვავებით, header ტეგი ხილულია და ის იწერება უკვე <body> </body> ტეგებს შორის

მაგალითი 1.7

```
<!DOCTYPE html>
<html Lang="ka">
<head>
<meta charset="utf-8">
<meta name="description" content="html 5-ს ვწავლობთ ">
<meta name="keywords" content="HTML5, CSS3, JavaScript">
<title>დოკუმენტის სათაური </title>
<link rel="stylesheet" href="mystyles.css">
</head>
<body>
<header>
<h1>ეს უკვე ხილული სათაურია ვებსაიტის </h1>
</header>
</body>
</html>
```

<header>

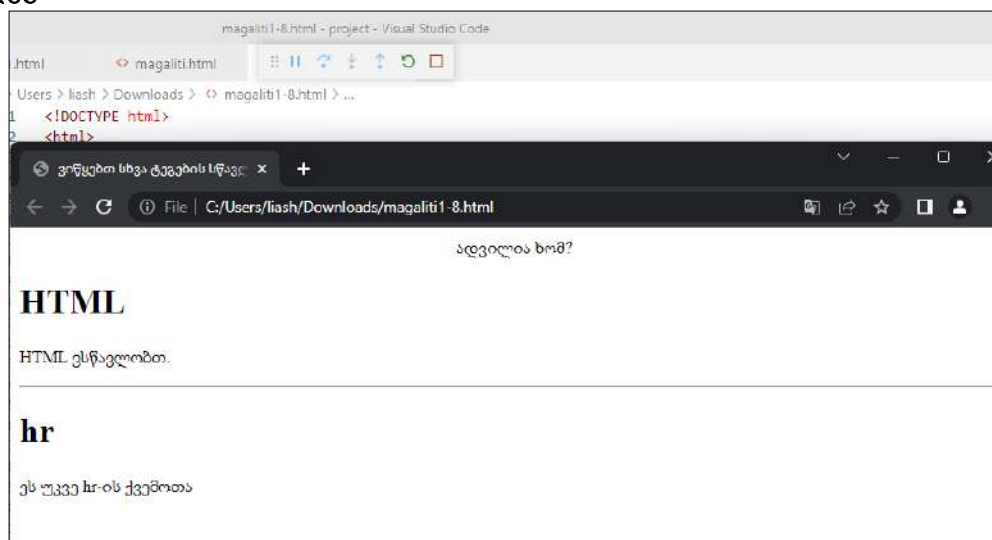
მიზანია, რომ წარმოადგინოს ნავიგაციის ლინკები და ლოგოები <h1> დან <h6> ამდენ, ყველა „ჰიდერის“ ელემენტი, რა თქმა უნდა, <header> ტეგი არ იმუშავებს ისეთ ტეგებთან როგორც <footer> ან <address>.

აქვე უნდა განვიხილოთ <hr> ტეგი, რომელიც HTML5-ში წარმოადგენს როგორც ჰორიზანტულ ხაზს, მას ვიზუალიზაციის დროს იყენებენ თემატიკუად, რათა წყვეტა მოხდეს.

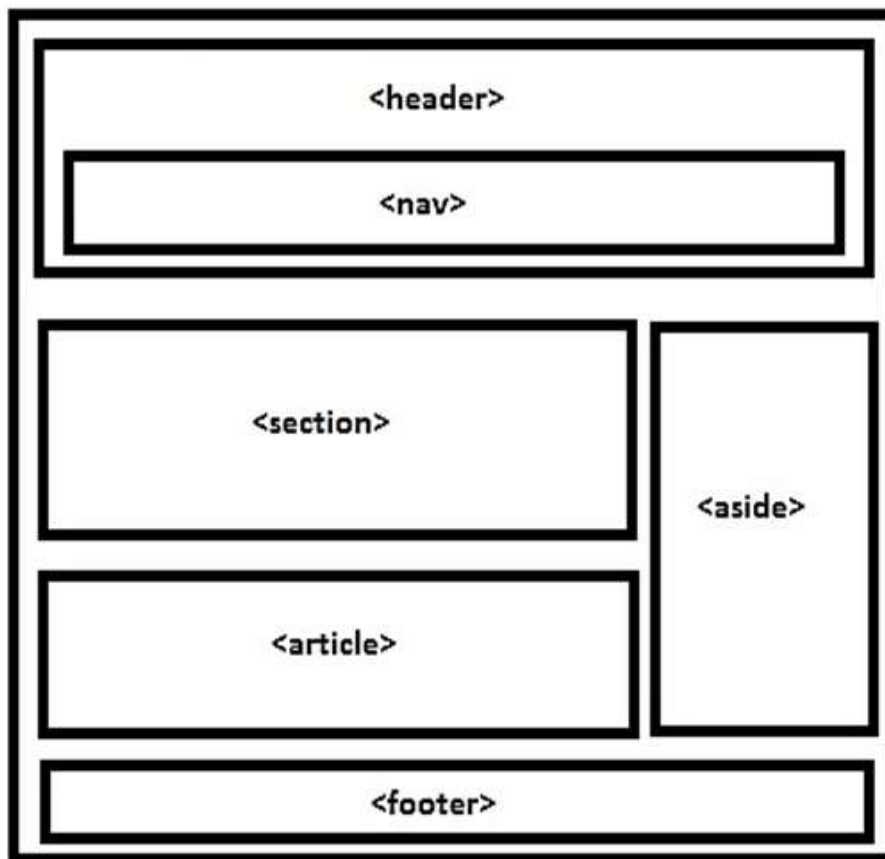
მაგალითი 1.8

```
<!DOCTYPE html>
<html>
<head>
<title> ვიწყებთ სხვა ტეგების სწავლას
</title>
<meta charset="utf-8">
</head>
<body>
<header>
<center>
ადვილია ხომ?
</center>
</header>
<h1>HTML</h1>
<p>HTML ესწავლობთ. </p>
<hr>
<h1>hr
</h1>
<p> ეს უკვე hr-ის ქვემოთა
<p>
</body>
</html>
```

შედეგი



სტრუქტურა, რომელიც ნაჩვენებია, სხვანაირიც შეიძლება იყოს, მთავარია, სკრიპტს როგორ დავწერთ. მოდი, განვიხილოთ თითოეული ტეგი, ახალი ტეგი, რომელიც მოცემული გვაქვს HTML5-ში არის `<header>`, `<nav>`, `<section>`, `<article>`, `<aside>` და `<footer>` ტეგი, რომელმაც უფრო გაამარტივა სინტაქსურად.



`<nav>`

მოვიყვანოთ ნავიგაციის მარტივი მაგალითი 1.9 რომელიც მოცემულია ქვემოთ სკრიპტი

მაგალითი 1.9

```
<!DOCTYPE html>
<html>
<meta charset="utf-8">
<body>
<nav>
  <a href="https://twitter.com/"> twitter </a> |
  <a href="http://gtu.ge">საქარ. ტექ. უნივერსიტეტი </a> |
  <a href="http://facebook.com">FACEBOOK </a> |
  <a href="http://ok.ru">Одноклассники </a>
</nav>
</body>
</html>
```

რაც შეეხება `<section>`, გამოყოფს სექციებს, ხოლო `<footer>` დოკუმენტის ქვედა კოლენტეტურია სტრუქტურულად, `<aside>` გამოყოფილი ფორმაა რომელიც

წარმოდგენილია როგორც გვერდითი ფორმა. სტრუქტურაზე მოცემული ტეგები წარმოვადგინოთ მთლიანობაში და დავეწეროთ შემდეგი სახით.

მაგალითი 1.10

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body bgcolor="#e5e5e5">
<header> სხვადასხვა ვებსაიტები </header>
<br>
<nav>
    <a href="http://gtu"> საქარ.ტექ.უნივერსიტეტი </a> |
    <a href="https://viciadmin.gtu.ge/#/login/login">ელექტრონული
სერვისების სისტემა </a> |
    <a href="http://facebook.com">FACEBOOK </a> |
    <a href="http://ok.ru">Одноклассники</a>
</nav>
<section>
<h1>ცხრილები </h1>
მოცემულ გვერდზე შეგიძლიათ მოიძიოთ ცხრილები .
ხოლო სურათი კი
<!-- ინტენეტში მოიძიეთ რომელიმე მისამართზე -->

</section>
<section>
<h1> საქართველოს ტექნიკური უნივერსიტეტი </h1>

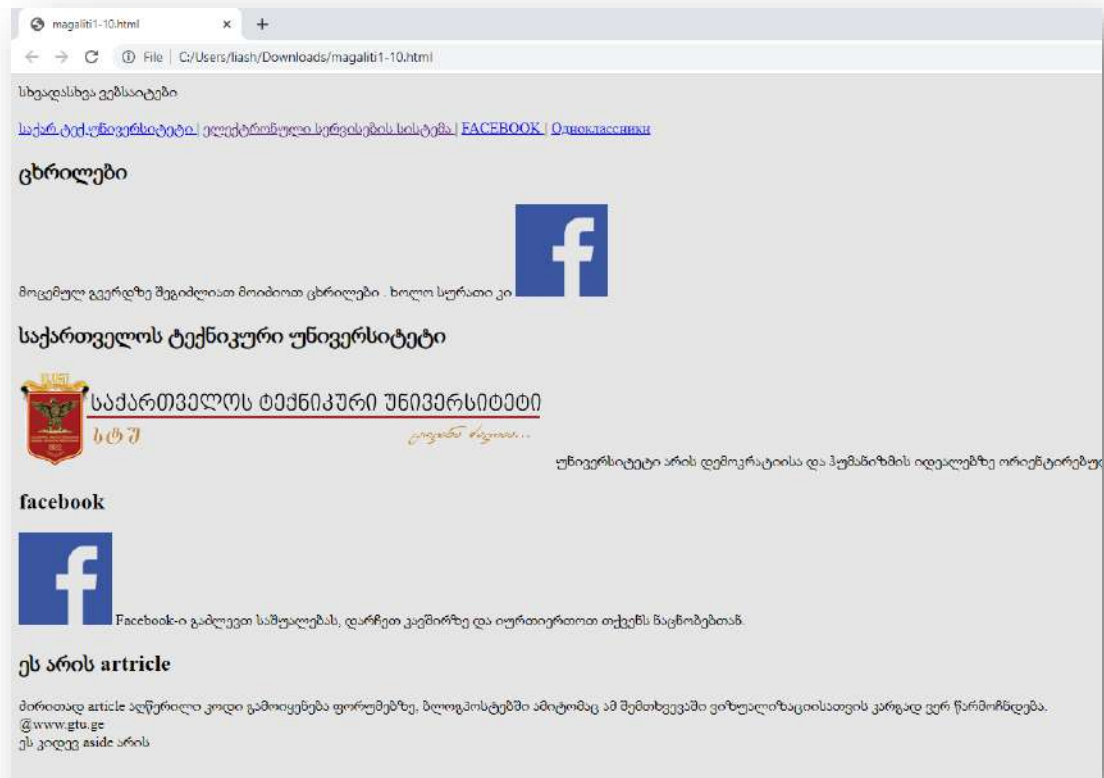
უნივერსიტეტი არის დემოკრატიისა და ჰუმანიზმის იდეალებზე ორიენტირებული
ინჟინერიის, ტექნოლოგიებისა და კულტურის განვითარების კერა.
</section>
<section>
<h1> facebook </h1>

Facebook-ი გაძლევთ საშუალებას, დარჩეთ კავშირზე და იურთიერთოთ თქვენს
ნაცნობებთან.
</section>
<article>
<h1> ეს არის article </h1>
ძირითად article აღწერილი კოდი გამოიყენება ფორუმებზე, ბლოგპოსტებში
ამიტომაც ამ შემთხვევაში ვიზუალიზაციისათვის კარგად ვერ წარმოჩნდება.
<footer> @www.gtu.ge</footer>
</article>
<aside> ეს კოდევ aside არის
</aside>
```


</body>

</html>

შედგენი



<address>

განსაზღვრავს მისამართს ან პიროვნების შესახებ ინფორმაციას.

<address>

მეილი:

ლამა იაშვილი



შეგახსენებთ: როცა სკრიპტს ავკრეფთ, გაფართოება სახელი .html უნდა მიეთითოს, ერთი მძიმის ან ტეგის ღიად დატოვებისას წარმოიქმნება სინტაქსური შეცდომა, და ჩვენი დაწერილი კოდი სრულყოფილი არ

ერთდროულად განვიხილოთ სამი ტეგი, რომლებიც HTML5 დანამატებია <details>, <summary> და <dialog> ამ უკანასკნელს აქვს თავისი open ატრიბუტი, რომელიც მიუთითებს, რომ დიალოგიური ფანჯარა აქტიურია, თვითონ ტეგები გვეუბნება, თუ რა რისი დანიშნულებაა:

მაგალითი 1.11:

<!DOCTYPE html>

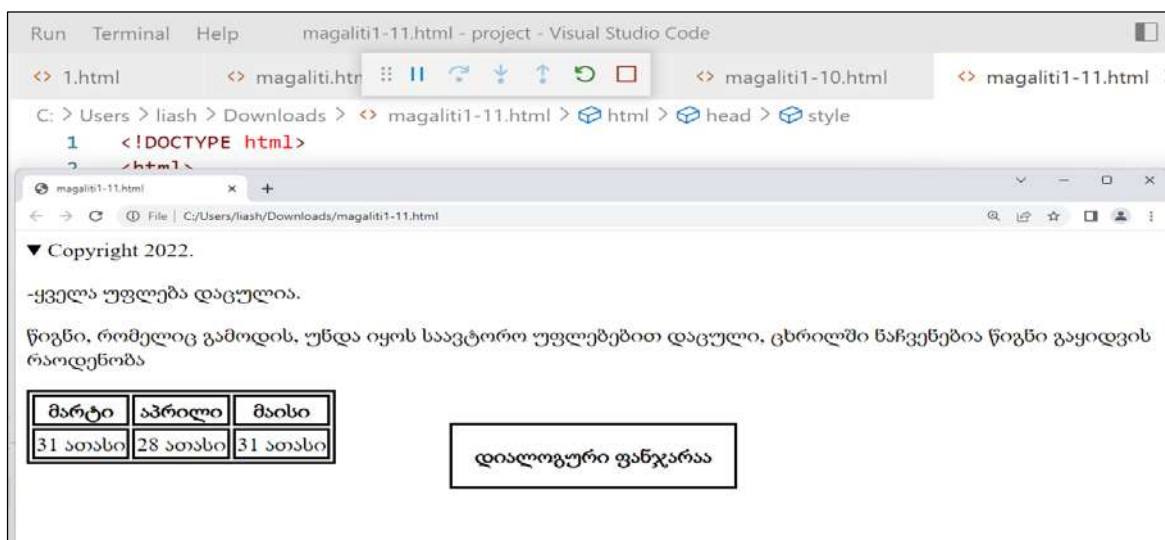
<html>

```

<head>
<style>
table, th, td
{
    border: 2px solid black;
}
</style>
</head>
<body>
<details>
    <summary> Copyright 2022.</summary>
    <p> -ყველა უფლება დაცულია.</p>
    <p>
        წიგნი, რომელიც გამოდის, უნდა იყოს საავტორო უფლებებით დაცული,
        ცხრილში ნაჩვენებია წიგნი გაყიდვის რაოდენობა
    </p>
</details>
<table>
    <tr>
        <th>მარტი <dialog open>დიალოგური ფანჯარა </dialog></th>
        <th>აპრილი </th>
        <th>მაისი </th>
    </tr>
    <tr>
        <td>31 ათასი </td>
        <td>28 ათასი </td>
        <td>31 ათასი </td>
    </tr>
</table>
</body>
</html>

```

შედეგი



CSS-ში სხვანაირად გამოჩნდება, როდესაც *Layout*-ებში გადავალთ.

<table>

<table> ტეგით ხდება ცხრილების შექმნა, ამ თეგთან მიმართებით არის კიდევ სხვა ტეგები, რომლის მეშვეობით შესაძლებელია რიგების შექმნა **<tr>**, ხოლო ცხრილების **<td>**, არის ცხრილების მონაცემები, სადაც შესაძლებელია მონაცემების შექმნა, ასევე აქვს **<th>** ანუ ცხრილების ჰიდერი.

<table> ცხრილს აქვს თავისი ატრიბუტი, როგორცაა **border=""**, რომელიც განსაზღვრავს ცხრილის საზღვრების სისქეს და **style=""**- სტილს.

„**align**“, „**bgcolor**“, „**cellpadding**“, „**cellspacing**“, „**frame**“, „**rules**“, „**summary**“, და „**width**“ HTML5-ში, **<table>** ტეგს აღარ აქვს ამ ატრიბუტების მხარდაჭერა.

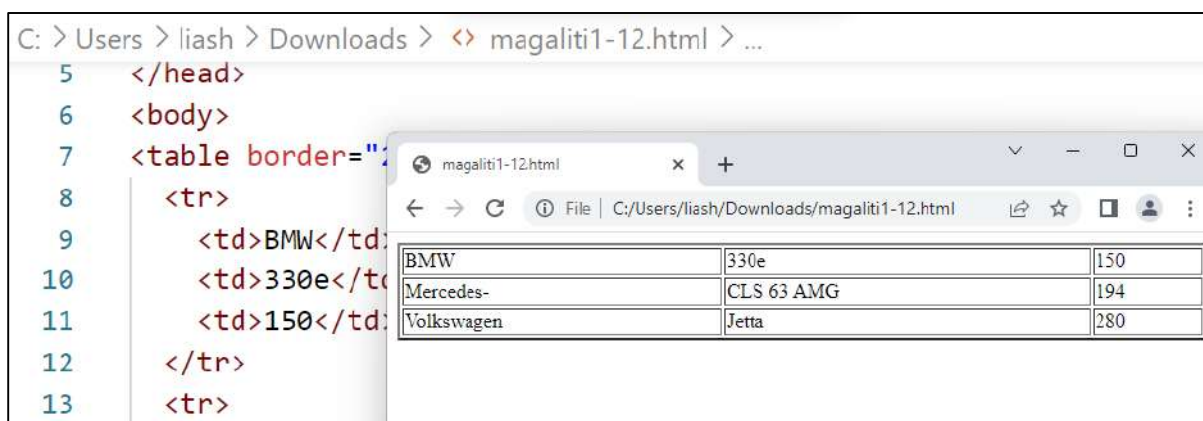
მაგალითი 1.12:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<table border="2" style="width:100%">
  <tr>
    <td>BMW</td>
    <td>330e</td>
    <td>150</td>
  </tr>
  <tr>
    <td>Mercedes-</td>
    <td>CLS 63 AMG</td>
    <td>194</td>
  </tr>
  <tr>
    <td>Volkswagen </td>
    <td>Jetta</td>
    <td>280</td>
  </tr>
</table>
</body>
</html>
```

ახლა დავუმატოთ ცხრილში ჰიდერები **<th>** და მივიღებთ შედეგს:

- ✓ **<table>** ცხრილის ტეგი.
- ✓ **<caption>** რომელიც განსაზღვრავს ცხრილში ტექსტის გატანას საზღვრებს გარეთ და მოქცეულია, როგორც სათაური.
- ✓ **<colgroup>** განსაზღვრავს კონკრეტული რიგის სხვადასხვა ფერში წარმოჩენას.
- ✓ **<col>** ტეგი ეხმარება **<colgroup>**-ს სვეტებისა და შესაბამისი რიგის სხვადასხვა ფერში წარმოჩენას.
- ✓ **<thead>** წარმოადგენს ცხრილის ჰიდერს.

- ✓ `<tbody>` არის ცხრილის „სხეული“.
- ✓ `<tfoot>` ცხრილის *footer* კონტეინერია.
- ✓ `colspan` და `rowspan` ელემენტი გვეხმარება რიგისა და სვეტის გაეთიანებაში.



მაგალითი 1.13

```

<!DOCTYPE html>
<html>
<head>
<style>
thead {color:green;}
tbody {color:blue;}
tfoot {color:red;}
table,th,td
{border:1px solid black;}
</style>
</head>
<body>
<table border="1" style="width:300px">
<thead>
<tr>
<th>სახელი </th>
<th>გვარი </th>
<th>ტელეფონი </th>
</tr>
</thead>
<tbody> <tr>
<td>ეკა</td>
<td>წერდიანი</td>
<td>555-5555</td>
</tr>
<tr>
<td>შოთა</td>
<td>გორგელიანი</td>
<td>555-5555</td>
</tr>

```

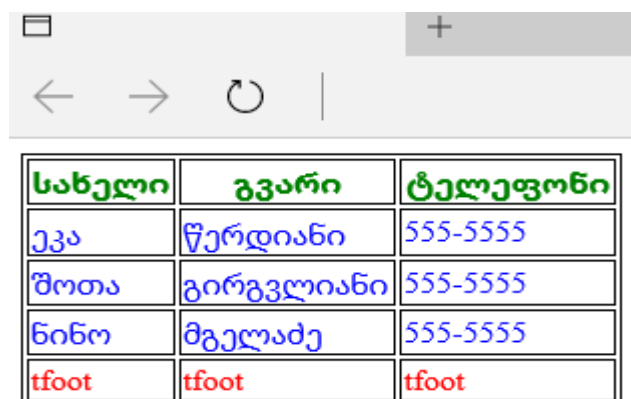
thead - მწვანე ფერისაა
tbody - ლურჯი ფერისა
tfoot - წითელი ფერისაა

```

<tr>
<td>ნინო </td>
<td>მგელაძე </td>
<td>555-5555 </td>
</tr>
<tbody>
<tfoot>
<tr>
<td>tfoot</td>
<td>tfoot</td>
<td>tfoot</td>
</tr> </tfoot>
</tbody>
</table>
</body>
</html>

```

შედეგი



სახელი	გვარი	ტელეფონი
ეკა	წერდიანი	555-5555
შოთა	გირგვლიანი	555-5555
ნინო	მგელაძე	555-5555
tfoot	tfoot	tfoot

<cite>

html5-ში <cite> ტეგი განსაზღვრავს რაიმე სათაურს სამუშაო ადგილის, წიგნის, მაღაზიის, სურათის თუ ა.შ. მაგალითისათვის სურათის ლინკი ავიღოთ *wikipedi*-დან და ზომები გაუწერეთ სიგანე და სიმაღლე 200px.

მაგალითი 1.14

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>cite ტეგი </title>
</head>
<body>

<p> <cite>ვსწავლობთ html 5-ს </cite></p>
</body>
</html>

```

შედეგი



ვსწავლობთ html 5-ს

span ტეგი გამოიყენება ტექსტის სტილის აღსაწერად, მაგალითი თუ გვინდა, რომ ტექსტი ან სიტყვა იყოს სხვა ფერში წარმოდგენილი, დავუშვათ მწვანედ, მაშინ სკრიპტი გვექნება:

მაგალითი 1.15

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>span</title>
```

```
</head>
```

```
<body>
```

```
<p>საყოველთაოდ ცნობილია, რომ
```

```
<span style="color:green; font-weight:bold">
```

```
ინტერნეტში </span>
```

განსათავსებელი ინფორმაციისათვის საჭირო სახის მისაცემად (ეს ინფორმაცია, ძირითადად, web-საიტების სახით არსებობს) ბაზისურ

```
<span style="color:darkolivegreen; font-weight:bold">
```

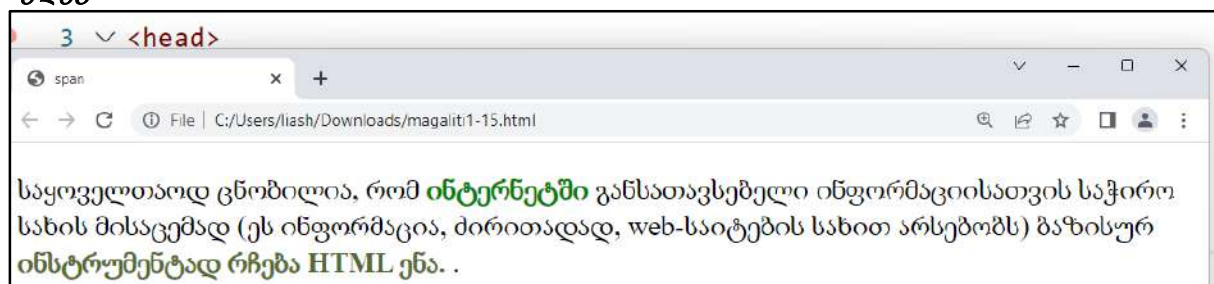
```
ინსტრუმენტად რჩება HTML ენა.
```

```
</span>. </p>
```

```
</body>
```

```
</html>
```

შედეგი



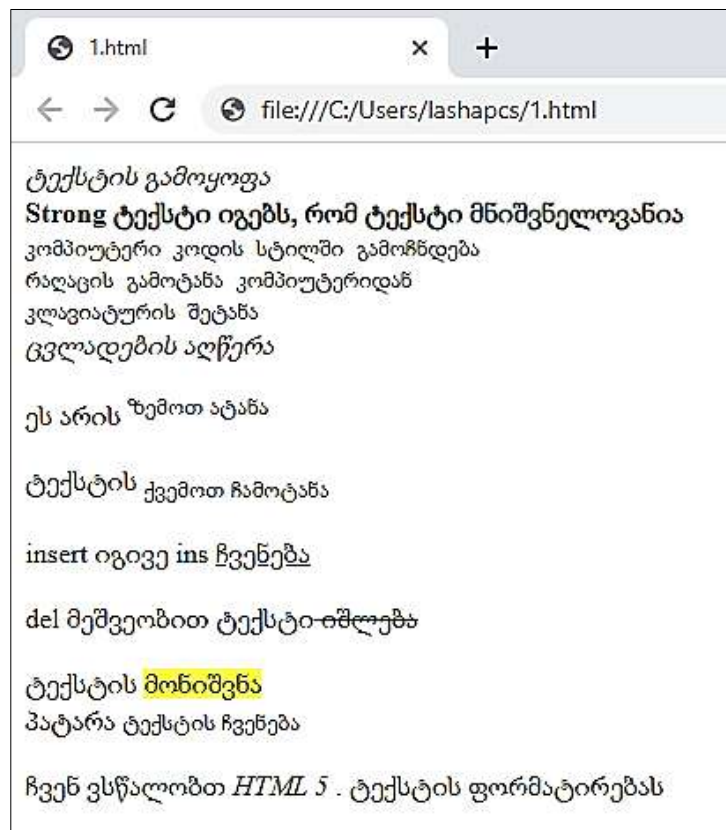
ტექსტის ფორმატირება

ტექსტის ფორმატირების სხვადასხვა ვარიანტი არსებობს, თითოეული ახნისა და განხილვის მაგივრად გთავაზობთ მაგალითებს, თუ რა დროს რა გამოყენება, დაფორმატირება გვაძლევს, ტექსტი წარმოვადგინოთ სხვადასხვა სტილში, მოვნიშნოთ რაიმე უბანი, დავეწყოთ ზემოთ, გამუქებულად, გამოჩნდეს ხაზგადასმული და ა.შ.

მაგალითი 1.16

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<em>ტექსტის გამოყოფა </em>
<br>
<strong>
Strong ტექსტი იგებს, რომ ტექსტი მნიშვნელოვანია
</strong>
<br>
<code>კომპიუტერი კოდის სტილში გამოჩნდება </code>
<br>
<samp>რადაცის გამოტანა კომპიუტერიდან </samp>
<br>
<kbd>კლავიატურის შეტანა </kbd>
<br>
<var>ცვლადების აღწერა </var>
<p>ეს არის <sup>ზემოთ ატანა </sup>
</p>
<p>ტექსტის <sub>ქვემოთ ჩამოტანა </sub>
</p>
<p>insert იგივე ins <ins>ჩვენება </ins>
</p>
<p>del მეშვეობით ტექსტი<del> იშლება </del>
</p>
ტექსტის <mark> მონიშვნა </mark>
<br>
პატარა <small> ტექსტის ჩვენება </small>
<p> ჩვენ ვსწალობთ <dfn> HTML 5 </dfn>. ტექსტის ფორმატირებას
</p>
</body>
</html>
```

შედეგი



<blockquote>

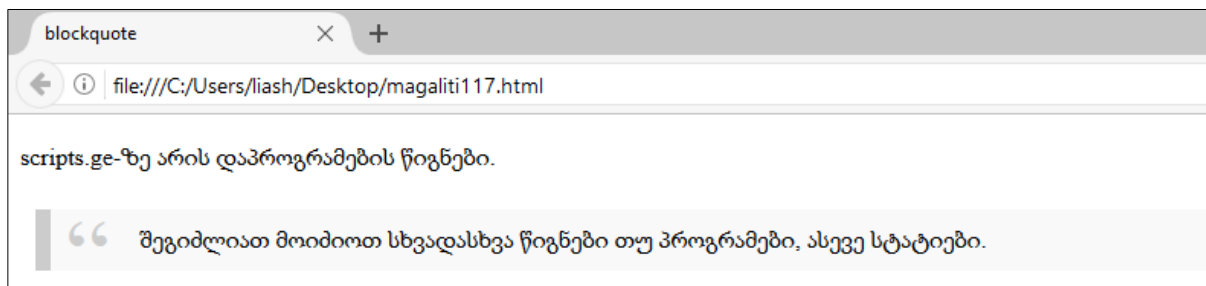
მაგალითად, თუ გვინდა რაიმე ტექსტის ციტირება და მისი გამოყოფა სხვა ტექსტიდან, შეგვიძლია გამოვიყენოთ <q> ტეგიც, რომლის ატრიბუტი არის cite ხოლო, მისი სინტაქსი ასე გამოიყურება:

```
<q cite="URL">
```

მაგალითი 1.17

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> blockquote </title>
</head>
<body>
<p>მიზანია ka.wikipedia.org
<q cite=" ka.wikipedia.org ">
Wikipedia არის თავისუფალი ენციკლოპედიის ქართულენოვანი პროექტი
<q>
აქ ბევრი საინტერესო ინფორმაცია, ჩაწერილი
</p>
</body>
</html>
```

რაც შეეხება <blockquote>, უფრო ეფექტურობისათვის ვნახოთ სხვა სკრიპტში

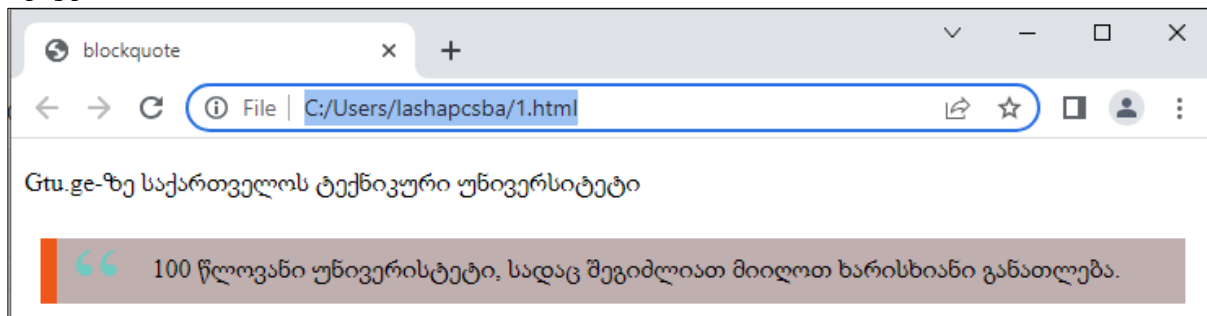


მაგალითი 1.18

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>
    blockquote
</title>
<style>
blockquote
{
    background: #c0afaf;
    border-left: 10px solid rgb(241, 86, 14);
    margin: 1.5em 10px;
    padding: 0.5em 10px;
    quotes: "\201C""\201D""\2018""\2019";
}
blockquote:before
{
    color: rgba(26, 224, 208, 0.534);
    content: open-quote;
    font-size: 4em;
    line-height: 0.1em;
    margin-right: 0.25em;
    vertical-align: -0.4em;
}
blockquote p
{
    display: inline;
}
</style>
</head>
<body>
<p>
    Gtu.ge-ზე საქართველოს ტექნიკური უნივერსიტეტი
    </p>
<blockquote cite="http://gtu.ge/">
    100 წლოვანი უნივერსიტეტი, სადაც შეგიძლიათ მიიღოთ ხარისხიანი განათლება.
```

```
</blockquote>
</body>
</html>
```

შედეგი

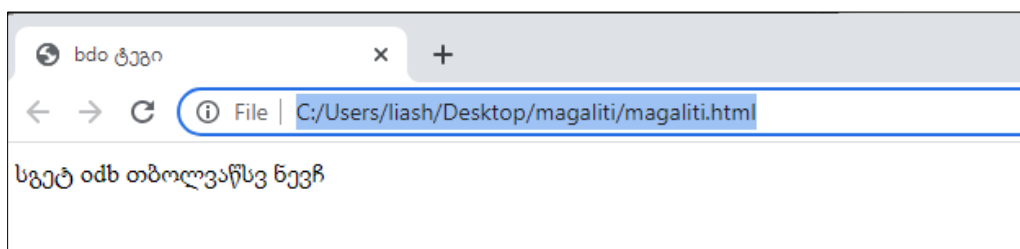


<bdo>

bdo ტეგი, რომლის მეშვეობით შესაძლებელია დაწერილი ტექსტის უკუღმა გამოჩენა, მაგალითად ავიღოთ რაიმე ტექსტი .

მაგალითი 1.19

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>bdo ტეგი</title>
</head>
<body>
<bdo dir="rtl">
ჩვენ ვსწავლობთ bdo ტეგს </bdo>
</body>
</html>
```



- ✓ ltr => ტექსტი დაიწერება მარცხნიდან მარჯვნივ;
- ✓ rtl => ტექსტი დაიწერება მარჯვნიდან მარცხნივ;
- ✓ auto => ბრაუზერი ავტომატურად აჩენს ტექსტს (მაგრამ არ არის რეკომენდებული)

ხოლო სინტაქსია:

```
<bdo dir="ltr ან rtl">
```

რაც შეეხება თვითონ *dir*, ეს არის „დირექტორი“ რომელიც განსაზღვრავს ტექსტის მიმართულებას და არის გლობალური ატრიბუტი.

სინტაქსი:

<ელემენტი dir="ltr ან rtl ან auto">

<time>

გვამლევს შესაძლებლობას, რომ განსაზღვროთ დროისა და თარიღის ინტერვალი, მისი ატრიბუტიან *datetime* მისი სინტაქსი;

<time datetime="YYYY-MM-DDThh:mm:ssTZD">

ქვემოთ მოცემულია დროის ფორმები, თუ როგორ ჩაიწერება

აღწერა	ატრიბუტი
YYYY-MM-DDThh:mm:ssTZD ან PTDHMS	დროების ჩაწერის ფორმები YYYY - წელი (მაგ; 2015) MM - თვე (მაგ; 01 იანვარი) DD - თვის დღე (რიცხვი) (მაგ;10) T ან space - გამყოფი (აუცილებელია თუ დრო მითითებულია) hh - საათი (მაგ; 21 ანუ 09.00pm) mm - წუთი (მაგ; 43) ss - წამი (მაგ; 05) TZD - Time Zone Designator (Z-განისაზღვრება როგორც Zulu, ცნობილია გრინვიჩის დროით) P - პრეფიქსი-პერიოდი D - პრეფიქსი-დღის H - პრეფიქსი-საათის M - პრეფიქსი-თვის S - პრეფიქსი-წამის

<area>

HTML <area> განსაზღვრავს *image-map* ანუ ურთერთობს სურათსა და რუკას შორის, თვითონ ტეგი საზღვრავს სურათის არეალებს, მაგალითისათვის შეიძლება ერთი სურათის დაკლიკების შემდეგ მეორე გამოჩნდეს, შემდეგ მესამე და ა.შ.

ატრიბუტი	აღწერა
alt	href წარმოდგენისას ტექსტის მითითება
coords	არეალის კოორდინატები
download	HyperLink დაკლიკების შემდეგ გადმოწერა
href	ლინკის მისამართი
hreflang	ლინკის მისამართის ენა
media	მედია/მოწყობილობა მისამართის ოპტიმიზება
rel	ურთერთობა სპეციალურ დოკუმენტსა და მისამართს შორის (URL)
shape	არეალის ნაპირი
target	URL მისამართი
type	სამიზნის მისამართის ტიპი

მოცემული ატრიბუტებიდან გამოვტოვებთ *alt*, *href*, *target*, რადგან ესენი უკვე ზემოთ განვიხილეთ, რაც შეეხება ორი ატრიბუტის, რომელსაც მოგვინებით შევხებით, ეს არის *type*, რომელიც განსაზღვრავს რადაცის ტიპს და *media*, რომელსაც შევხებით მოგვიანებით, როდესაც მულტიმედიას განვიხილავთ.



shape

სანამ *coords* განვიხილავდეთ შევხებით, *shape* ატრიბუტს, რომელიც არეალებს აძლევს გარკვეულ ფორმას, რომელიც გამოყენება *coord*-ს (კოორდინატებთან) ერთად, რომელიც განსაზღვრავს ზომას, ადგილ მდებარეობას.

სინტაქსი

```
<area shape="default ან rect ან circle ან poly">
```

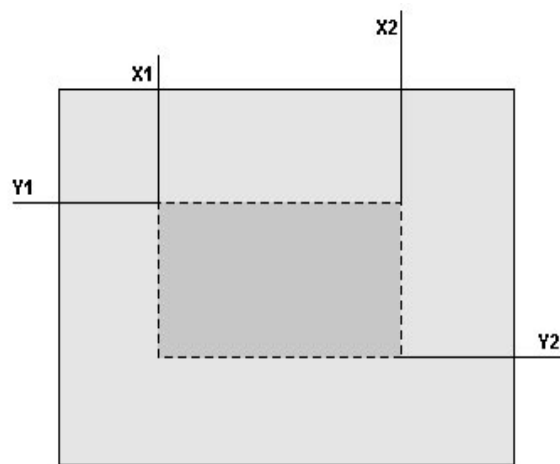
მოცემულია ატრიბუტების სიდიდეები, რომელიც ეთითება, ყველას თავის დანიშნულება აქვს, მაგალითი *default* განსაზღვრავს ავტომატურად რეგიონს, ხოლო *rect* ანუ *rectangular* (მართკუთხედ) ფიგურა, *circle*-წრიული ფიგურას და *poly*- პოლიგონურ ფიგურას.

coords

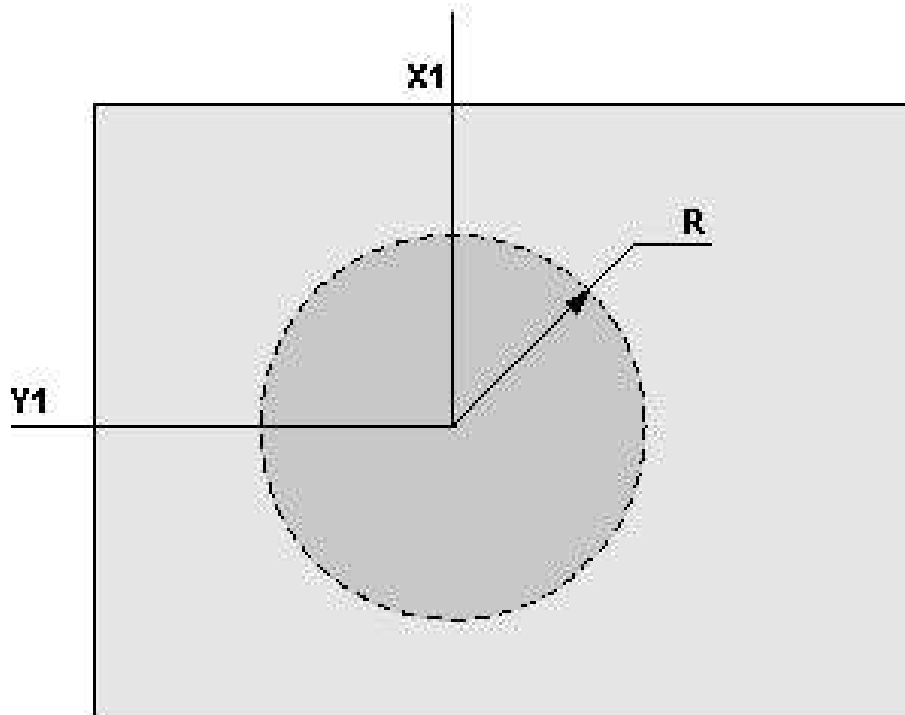
image-map-ში განსაზღვრავს კოორდინატებს, ასევე უნდა ითქვას როგორც *shape*, ისე *coords* ურთიერთშეხამებით მუშაობს და განსაზღვრავს ფორმებს, კოორდინატებს და ზომას, ორივე ატრიბუტს ერთი და იგივე განმარტება აქვს, მაგრამ განსხვავებული ატრიბუტის ელემენტები, კოორდინატები განისაზღვრება შემდეგნაირად.

სიდიდე	აღწერა
<i>x1,y1,x2,y2</i>	მართკუთხედის კოორდინატებია თანამიმდევრობით მარცხნივ, ზემოთ, მარჯვნივ და ქვემოთ
<i>x,y,radius</i>	წრეწირის ცენტრის კოორდინატები და რადიუსი
<i>x1,y1,x2,y2,...,xn,yn</i>	პოლიგონური კოორდინატები

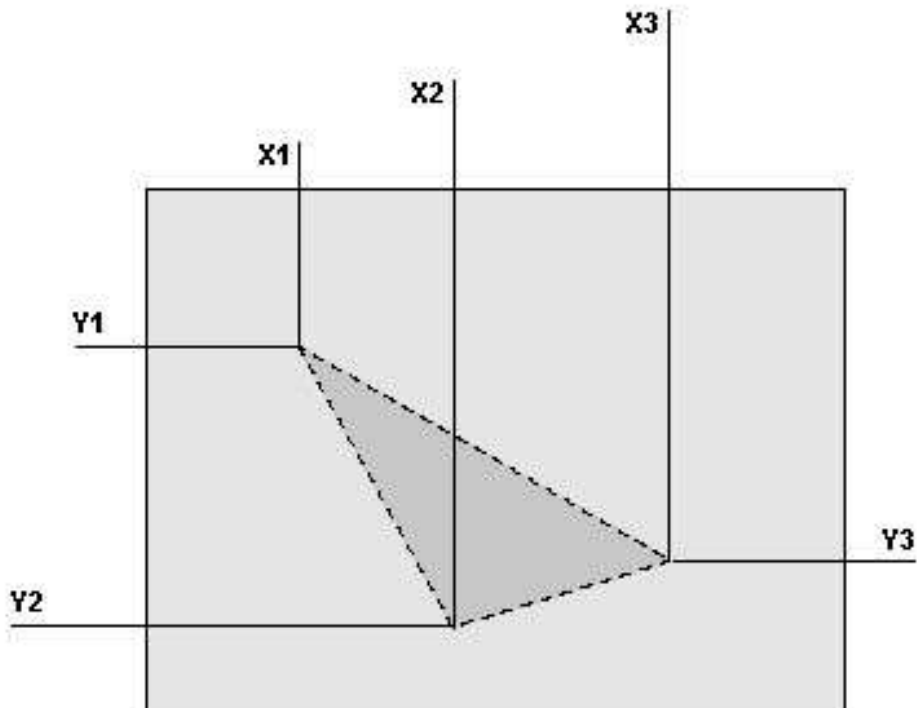
უფრო ადვილად გასაგები და აღსაქმელი რომ იყოს, მოცემულია სურათების სახით.
shape="rect" კოორდინატები (*X1,Y1,X2,Y2*)



`<area shape="circle" (X1,Y1,R)`



`shape="poly" (X1, Y1, X2, Y2, ..., Xn, Yn)`



მაგალითი 1.20

```
<!DOCTYPE html>
<html>
<head>
<title>
HTML area Tag
</title>
</head>
<body>
<p>

<map name="CHmap">
<area shape="rect" coords="89,9,294,50"
href="https://gtu.ge/bitrix/templates/GTU-primary/img/gtu-logo.webp"
alt="DOWNLOADS">
<area shape="circle" coords="297,7,407,54"
href="https://gtu.ge/bitrix/templates/GTU-primary/img/gtu-logo.webp"
alt="video">
</map>
</body>
</html>
```

შედეგი



პოზიციები ახლა ისე არის განაწილებული, რომ მარცხნივ თუ დააკლიკებთ, გადავა *download*-ებში, ხოლო მარცხნივ, ვიდეოებში.

download

გადმოწერა რაიმესი, არ აქვს მნიშვნელობა იქნება ეს ვიდეო, სურათი, HTML დოკუმენტი თუ სხვა რამ, როდესაც მომხმარებელი დააკლიკებს სურათს, ის ავტომატურად გადმოიწერს, მთავარი მოთხოვნაა, რომ *href* ელემენტი იყოს გამოყენებული.

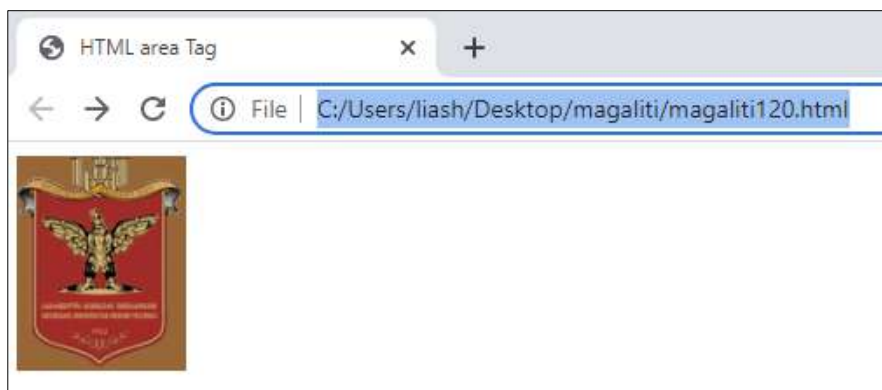
სინტაქსი

`<a download="ფაილის სახელი">`

მაგალითი 1.21

```
<!DOCTYPE html>
<html>
<head>
<title>HTML area Tag</title>
<meta charset="utf-8">
</head>
<body>
<a href="https://leqtori.gtu.ge/assets/img/small_logo.bmp" download>
  
</a>
</body>
</html>
```

შედეგი



სურათზე ნებისმიერ ადგილას დაკლიკების შემდეგ ავტომატურად გადმოიწერს, ყველა ბრაუზერს აქვს *download* ატრიბუტის მხარდაჭერა, გარდა *Internet explorer 11*, ხოლო *Windows 10*-ში ჩაკერებულ ახალ ბრაუზერს *Microsoft Edge* აქვს უკვე მისი მხარდაჭერა.

hrefLang

hrefLang ატრიბუტი განსაზღვრავს დოკუმენტის ლინგვის ენას, ენა ირჩევა *Language code*-დან, რომელსაც ქვემოთ შევხებით, ქართული *Language code* არის *ka*, მაგალითში ნაჩვენებია ილუსტრირება მოცემული ატრიბუტისა.

მაგალითი 1.22

```
<!DOCTYPE html><html>
<head>
<meta charset="utf-8">
</head>
<body>
<p><a hreflang="ka" href="http://scripts.ge">SCRIPTS.GE</a>
</p>
<p>
<a hreflang="ka" href="http://gtu.ge">საქართველოს ტექნიკური
უნივერსიტეტი </a>
</p>
</body>
</html>
```

[SCRIPTS.GE](http://scripts.ge)
[საქართველოს ტექნიკური უნივერსიტეტი](http://gtu.ge)

rel

rel ატრიბუტს დამხმარე ატრიბუტია, რომელიც ურთიერთქმედებს არსებულ დოკუმენტსა და მიბმულ დოკუმენტს შორის, მოცემულ ატრიბუტს აქვს თავისი სიდიდე:

ცვლადი	აღწერა
<i>alternate</i>	ალტერნატიული ლინკი დოკუმენტის მაგალითი, საბეჭდი გვერდი, თარგმანი და ა. შ.
<i>author</i>	დოკუმენტის ავტორი
<i>bookmark</i>	ჩასანიშნი ლინკი
<i>help</i>	დოკუმენტის help-ი
<i>license</i>	დოკუმენტის საავტორო უფლებები
<i>next</i>	შემდეგი დოკუმენტის მონიშვნა
<i>nofollow</i>	საძიებო სისტემამ რომ ვერ იპოვოს კონკრეტული ლინკი
<i>noreferrer</i>	ბრაუზერი არ აგზავნის HTTP header-ში ნახსენებ <i>hyperLink</i> -ებს
<i>prefetch</i>	ქეშირებული დოკუმენტის მონაცემები
<i>prev</i>	წინა დოკუმენტი ჩვენება
<i>search</i>	დოკუმენტში ძებნა
<i>tag</i>	საკვანძო სიტყვები დოკუმენტებში
<i>stylesheet</i>	დიზანი- CSS ფაილთან ურთერთობა

მაგალითი 1.23

```
<!DOCTYPE html>
<html><body>
<p>
```



```
<a rel="nofollow" href="gtu.ge"> არ მოძებნის ამ საიტს </a>
</p>
</body>
</html>
```

<Link>

rel ატრიბუტში მონათესავე ტეგი, რომლის საშუალებით ხდება დოკუმენტში, ლინკების მითითება, პრაქტიკაში დიზანის ფაილის მისამართებს აღწერენ ხოლმე, რაც შეეხება, ატრიბუტებს იგივე ატრიბუტები აქვს, რაც *rel*-ს.

მაგალითი

```
<html>
<head>
<meta charset="utf-8">
<head>
<link rel="stylesheet" type="text/css" href="/დიზაინი.css">
<!-- ხშირად მოცემულია შემდეგი სახით "href="/style.css">-->
</head>
<body>
</body>
</html>
```

მოცემული ტეგი მხოლოდ და მხოლოდ დაიწერება *<head>* ტეგებს შორის *</head>*.

გლობალური ატრიბუტები

ორიოდე სიტყვით შევეხოთ გლობალურ ატრიბუტს, ეს ისეთი ატრიბუტია, რომელიც ნებისმიერი ტეგის შემთხვევაში შეგვიძლია გამოვიყენოთ, ქვემოთ მოცემულია ცხრილის სახით, სახამ გავარჩევდეთ თითოეულ მათგანს, უნდა შევეხოთ **DOM**-ს, რადგანაც ზოგი მათგანი DOM-ის ელემენტებს შეიცავს.

ატრიბუტები	აღწერა
accesskey	კლავიატურის ერთგვარი იარაღი (shortcut)
class	კლასი იქმნება, სადაც შეგვიძლია ერთი ან მეტი სახელის დარქმევა ელემენტისათვის
contenteditable	შესაძლებელია ტექსტის ჩასწორება
contextmenu	მენიუს კონტექსტი, რომლის მხარდაჭერა მხოლოდ <i>firefox</i> -ს აქვს
data-*	მონაცემების შემცველის გვერდის ამოვარდნა
dir	დირექტორი, რომელიც განსაზღვრავს მიმართულებას
draggable	სადაც შესაძლებელია ტექსტისა და სურათის გადაადგილება
dropzone	იძლევა საშუალებას ჩამოშლილი ტექსტის ან კოპირების სხვა ადგილას გადატანას, სამწუხაროდ, ამ ატრიბუტს მხარდაჭერა არცერთი ბრაუზერისგან, არ აქვს
hidden	ტექსტის დამალვა
id	რომელიც უნიკალურია

Lang	განსაზღვრავს ენას, თუ ტექსტი რომელ ენაზეა დაწერილი
spellcheck	განსაზღვრავს მართლწერას
style	სტილისტური ატრიბუტი
tabindex	კლავიატურაზე <i>tab</i> ღილაკით ნავიგაციაზე გადასვლა
title	სათაური, როგორც ატრიბუტი, შეიცავს ტექსტზე ინფორმაციას
translate	თარგმნის, ამ ტეგის მხარდაჭერა არცერთ ბრაუზერს არ აქვს

რა არის DOM?

DOM (Document Object Model)-ანუ დოკუმენტის ობიექტური მოდელი, რომელიც **HTML** დოკუმენტის ელემენტებს, წარმოადგენს როგორც ობიექტის ელემენტს და აძლევს შესაძლებლობას, მის მეშვეობით მოხდეს გარკვეული ცვლილება **HTML**-ის დოკუმენტში, თითოეული ატრიბუტის განხილვისას შევხებით **DOM**-საც, რომელიც **Javascript**-ს სინტაქსის შემადგენელი ნაწილია.



Javascript არის დაპროგრამირების ენაა, რომელიც **HTML** და **CSS** დახმარებით ვებუბნებით **Web** საიტს, რისი გაკეთება გვსურს; მაგალითად ტექსტის შეცვლა, კალკულაცია და ა. შ

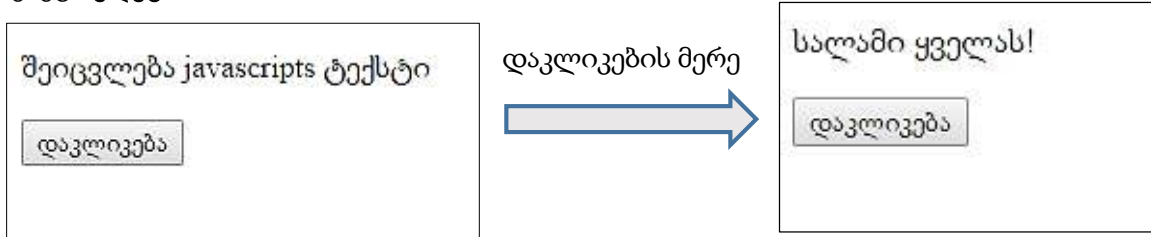
იმიტაციისათვის მოცემულია პატარა სკრიპტი

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<p id="html">შეიცვლება javascripts ტექსტი<p>
<button
  type="button" onClick="document.getElementById('html').innerHTML =
    'სალამი ყველას!'">
დაკლიკება
</button>
</body>
</html>

```

ესეც შედეგი:



HTML წინამდებარე ვერსიაში აღწერა ხდებოდა შემდეგი სახით:

```
<div id="mydiv" brand="toyota" model="rav 4">
```

Rav4 კარგი მოდელია.

```
</div>
```

ხოლო შემდეგ შეიმუშავეს ატრიბუტი data, რომელიც მონაცემებს აღწერდა:

```
<div id="mydiv" data-brand="toyota" data-model="rav4">
```

Rav4 კარგი მოდელია.

```
</div>
```

```
var mydiv=document.getElementById('mydiv')
```

```
//ჯავასკრიპტის სკრიპტის აღწერისას
```

```
var brand=mydiv.dataset.brand //დააბრუნებს „toyota“-ს
```

```
mydiv.dataset.brand='Volkswagen'// შეცვლის „data-brand“ „Volkswagen“
```

```
mydiv.dataset.brand=null //წაშლის „data-brand“ ატრიბუტს
```

რაც შეეხება // ამ სიმბოლოს ეს ჯავასკრიპტში ერთ სტრიქონიანი კომენტარია გაკეთებული როგორც HTML-ში <!--მოთავსებული ტექსტი--> .

Accesskey

არსი ის არის, რომ როდესაც მივუთითებთ რაიმე საკვანძო სიტყვას, კლავიატურაზე კომბინაციით შესაძლებელია მოვახდინოთ მარტივი გადამისამართება. ამ შემთხვევაშიც სხვადასხვა ბრაუზერის შემთხვევაში სხვადასხვანაირად ხდება ხოლმე:

- ✓ IE, Chrome, Safari, Opera:[ALT] + accesskey
- ✓ Opera 15 წინამდებარე ვერსიას:[SHIFT] [ESC] + accesskey
- ✓ Firefox:[ALT] [SHIFT] + accesskey

მაგალითი 1.24

გვაქვს მოცემული სხვადასხვა უნივერსიტეტის საიტი, რომლის კლავიატურაზე კომბინაციით გვინდა, გადამისამართდეს შესაბამის საიტზე:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
</head>
```

```
<body>
```

```
<a href="http://www.gtu.ge" accesskey="g"> სტუ </a>
```

```

<a href="http://seu.edu.ge/" accesskey="s">სეუ </a>
<a href="http://tsu.ge" accesskey="t">თსუ </a>
</body>
</html>

```

თუ ვიყენებთ *google Chrome*-ს და დავაწვებით [ALT] + გ, ფაილი ავტომატურად გადავა ტექნიკური უნივერსიტეტის საიტზე, იმავე კომბინაციით და დავაწვებით s-ს , მაშინ სეუ-ს საიტზე და ა. შ.

class-კლასი, კლასის შემთხვევაში შეგვიძლია გვქონდეს უამრავი კლასი სხვადასხვა სახელებით. კლასებთან დაკავშირებით ვილაპარაკეთ, როდესაც *div* ტეგს შევხებით.

contenteditable-შეგვიძლია მოთავსებულ ტეგებში არსებული ტექსტის რედაქტირება, მაგრამ ასევე აქვს თავისი ატრიბუტები, რომლის საშუალებით განისაზღვრება რედაქტირების უფლებები შესაძლებელია თუ არა.

true და false, თუ მივუთითებთ true-ს, მაშინ მოვახდენთ ტექსტის რედაქტირებას, false-ს შემთხვევაში ვერა.

სინტაქსი გამოიყურება შემდეგნაირად:

```
<ელემენტი contenteditable="true ან false">
```

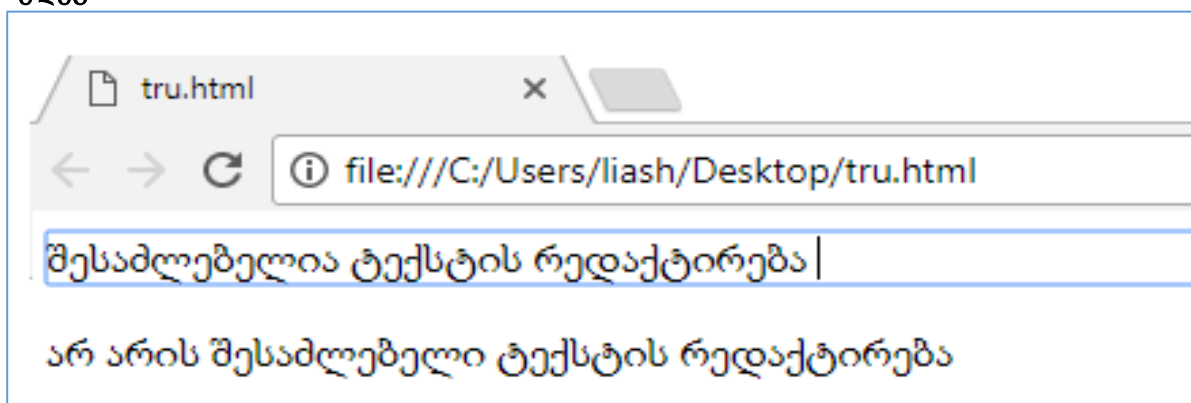
მაგალითი 1.26

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
  <p contenteditable="true">
    შესაძლებელია ტექსტის რედაქტირება
  </p>
  <p contenteditable="false">
    არ არის შესაძლებელი ტექსტის რედაქტირება
  </p>
</body>
</html>

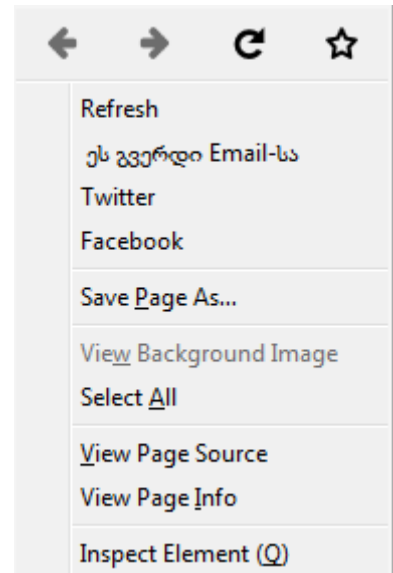
```

შედეგი



`true`-ს შემთხვევაში მაუსით ორჯერ დაკლიკეთ და თქვენთვის სასურველი ტექსტი ჩასწორდება, თუმცა სკრიპტში ცვლილება არ მოხდება.

`contextmenu`-კონტექსტური მენიუ, რომლის მხარდაჭერა ბრაუზერებიდან აქვს მხოლოდ *firefox*-ს, და გამოჩნდება ისიც იმ შემთხვევაში, თუ მაუსის მარჯვენათი „დაკლიკებით“, მივიღებთ ისეთ შედეგს, რასაც აღწერთ სკრიპტში. ჩვენი მაგალითიდან გამომდინარე, გავუწერეთ სოციალური ქსელები, როგორიცაა: *Twitter* და *Facebook*, რაც სურათიდან ჩანს, რომ თუ დავაკლიკებთ მარჯვენა მაუსს, გამოჩნდება სურათზე მოცემული ფანჯარა, უნდა აღინიშნოს ისიც, რომ ამ შემთხვევაში ვეხებით *javascript*-ის ფაილებს. სანამ HTML-ის სკრიპტზე გადავიდოდეთ, უნდა განვიხილოთ ისეთი მნიშვნელოვანი ტეგი, როგორიცაა `<menu>` და `<menuitem>`.



`<menu>` ტეგი განსაზღვრავს მენიუს, სადაც შეიძლება მენიუს ან სხვა რაიმე ბრძანების ჩამატება, მისი „დამატებითი“ ტეგი არის `<menuitem>` რომელიც განსაზღვრავს მენიუს ტიპებს და `label` ატრიბუტის საშუალებით მომხარებელს ატყობინებს ინფორმაციას ან რაიმე გარკვეულ ბრძანებას, მისი სინტაქსი ასეთია:

`<menuitem label="ტექსტი ">`

რაც შეეხება მის ატრიბუტებს გამოიყენება `label` რომელზეც ვილაპარაკეთ და `type` ანუ იგივე „ტიპი“, რომლის სინტაქსი შესაბამისად გამოიყურება, შემდეგი სახით:

`<menu type="list ან context ან toolbar">`

`List`, `context` და `toolbar` საშუალებით შესაძლებელია მენიუში სიის, კონტექსტის `toolbar`-ების შექმნა. დავუბრუნდეთ `contextmenu`-ს, რომელის სკრიპტი გამოიყურება შემდეგნაირად:

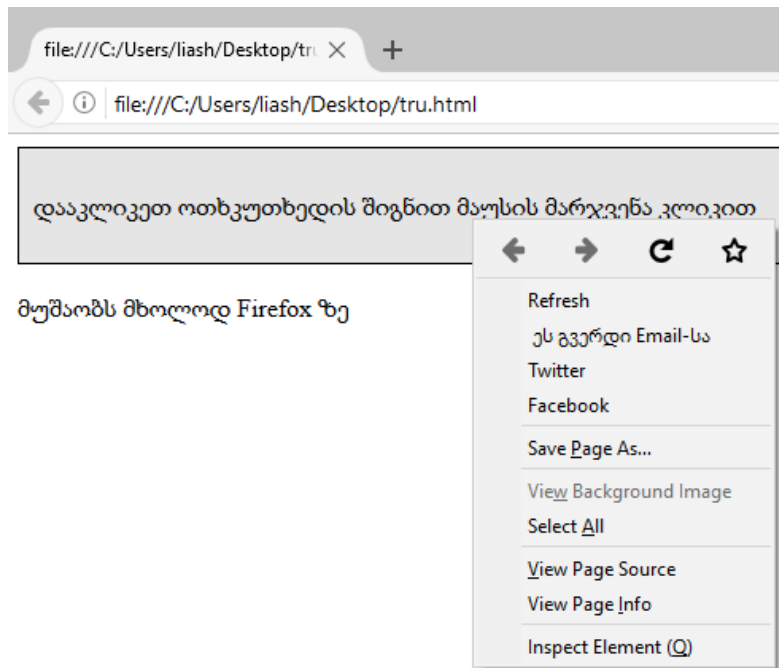
მაგალითი 1.27

```
<!DOCTYPE html>
<html>
<head>
<style>
div
{
background: #e5e5e5;
border: 1px solid black;
padding: 9px;
}
</style>
</head>
<body>
<div contextmenu="mymenu">
<p>
```

დააკლიკეთ ოთხკუთხედის შიგნით მაუსის მარჯვენა კლიკით

```
<menu type="context" id="mymenu">
  <menuitem label="Refresh" onclick="window.location.reload();"
  icon="ico_reload.png">
</menuitem>
  <menuitem label=" ეს გვერდი Email-სა"
  onclick="window.location='mailto:?body='+window.location.href;">
</menuitem>
  <menuitem label="Twitter" icon="ico_twitter.png" onclick=
  "window.open('//twitter.com/intent/tweet?text=' + window.location.href);">
</menuitem>
  <menuitem label="Facebook" icon="ico_facebook.png" onclick=
  "window.open('//facebook.com/sharer/sharer.php?u=' +
  window.location.href);">
</menuitem>
</menu>
</div>
<p>
მუშაობს მხოლოდ Firefox ზე
</p>
</body>
</html>
```

შედეგი



მუშაობს მხოლოდ Firefox ზე

რაც შეეხება *icon*-ებს, მისი მეშვეობით იკონების მითითება შეგვიძლია, ხოლო შეეხება *onClick="window.open"*-ეს *Event*-ებია, რომელზეც ცოტა ქვემოთ ვიმსჯელოთ, ჯერ ჯერობით ამ ბრძანების მეშვეობით დავიმახსოვროთ, ხოლო „დაკლიკებით“ ახალ ფანჯარაში უნდა გაიხსნას.



მოცემული ტეგი თანამედროვე ბრაუზერებში აღარ მუშაობს, მსგავსი მენიუს გაკეთება შესაძლებელია javascript-ის საშუალებით, ამიტომ ამ ეტაპზე თუ როგორ უნდა გაკეთდეს, არ განვიხილავთ.

data-*

გლობალური ატრიბუტია, რომელიც HTML 5-ში ჩამატებული ახალი ტეგია და გამოყენება მომხმარებლის მონაცემების აღსაწერად, *data* ატრიბუტს აქვს ერთი დამხმარე ატრიბუტი, რომელიც აღწერს თვითონ მონაცემს და შემდეგ ამ მონაცემის კლასიფიცირება ხდება, მაგალითი: ავიღოთ ავტომობილი, ლათინური შრიფტით დავწეროთ *-manqana* და დავყოთ ეს „მანქანა“ სხვადასხვა ფირმის მოდელთან და ასევე ავიღოთ ცხოველი, ტიპი, მაგრამ სანამ სკრიპტს ვნახავდეთ, გავეცნოთ *data* ატრიბუტის სინტაქსს:

<ელემენტი data-= "რადაც ცვლადი">*

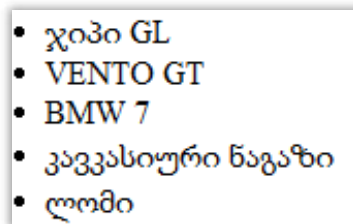
მაგალითი 1.28

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8">
<title>data-ს აღწერა </title>
```

```

</head>
<body>
  <!-- ტიპად აღებულია manqana ხოლო კლასიფიცირება ხდება მისი როგორც
  „მერსედესი“, „ფოლკსვაგენი“, "BMW" და შედეგად მოდელებია-->
  <li data-manqan="Mercedes">ჯიპი GL</li>
  <li data-manqana="volkswagen">VENTO GT</li>
  <li data-animal-type="BMW">BMW 7</li>
  <!-- მაგალითი ორი ავილოთ სხვადასხვა ცხოველები-->
  <li data-cxovelebi="ძაღვი">კავკასიური ნაგაზი</li>
  <li data-cxovelebi="კატა">ლომი</li>
</body>
</html>
შედეგი

```



dir- ატრიბუტთან დაკავშირებით ვიტყვით, რომ ის მიმართულებას აღწერს, აღარ ვიმეორდებთ ამ ატრიბუტთან დაკავშირებით **<bdo>** რაც ტეგისას ვისაუბრეთ.

draggable-ატრიბუტი, რომელიც გვამლევეს, ჩავაგდოთ სადღაც რაღაც, მაგალითისათვის ავილოთ რაიმე ტექსტი და ოთხკუთხედში ჩავსვათ, ანალოგიურად როგორც ტექსტის, ისე სურათის ჩასმაც შეიძლება.

მისი სინტაქსია შემდეგი სახისაა:

<ელემენტი draggable="true ან false ან auto">

- ✓ **true** როცა შესაძლებელია ელემენტის ჩაგდება
- ✓ **false** როცა არ არის შესაძლებელი ელემენტის ჩაგდება
- ✓ **auto** როცა ავტომატურად გამოიყენებს ბრაუზერი და არ ჭირდება რაიმე ნებართვა.

ეს ატრიბუტი მუშაობს **DOM** გამოყენების დროს, დამოუკიდებლად სინტაქსურ შეცდომასთან არ გვექნება საქმე, მაგრამ ბრაზერში ვერაინაირ მოქმედებას ვერ შევასრულებთ.

<div draggable="true">ჩამავდე</div>

უნდა ითქვას, რომ **ondragstart** და **setData()** გარეშე ვერ მოახდენს ჩაგდებას ატრიბუტმა უნდა იცოდეს, რა უნდა ჩავარდეს და როდის უნდა დაიწყოს.

- ✓ **Drag** და **Drop** (ჩაგდება და გადაადგილება) განვავრცოთ, თუ როგორ ხდება, პირველ რიგში: შევუბოთ ისეთ ფუნქციას, რომელიც აღწერს, თუ რა უნდა გადაადგილდეს **ondragstart**-ის მეშვეობით, ფუნქცია კი არის **drag(event)** შემდეგში, რასაც უნდა შევუბოთ, არის **dataTransfer.setData()** მეთოდი, რომელიც აღწერს მონაცემის ტიპს და სიდიდეს:


```
function drag(ev)
{
    ev.dataTransfer.setData("ტექსტი", ev.target.id);
}
```

სადაც მონაცემთა ტიპი არის „ტექსტი“ სიდიდე *id*, რომელიც უნდა გადაადგილდეს პირობითად მივანიჭოთ „*drag1*“. ახლა დგება მეორე საკითხი, თუ სად უნდა გადაადგილდეს. რაც შეეხება „ჩაგდებას“, *ondragover* პასუხისმგებელია, რა მონაცემები უნდა გადაადგილდეს ანუ ერთმა ელემენტმა უნდა ჩაანაცვლოს მეორე ელემენტი, რასაც ჰქვია *event.preventDefault()* რომელიც არის *ondragover*-ის ივენთი. ივენთები იგივეა, რაც „მოვლენები“, რომელიც განსაზღვრავს, თუ რა უნდა მოუვიდეს *HTML*-ის ელემენტებს.

```
function drop(ev)
{
    ev.preventDefault();
    var data = ev.dataTransfer.getData("ტექსტი");
    ev.target.appendChild(document.getElementById(data));
}
```

გამოიძახებს სტანდარტულზე დაყენებულ ბრაუზერს

არსებული ინფორმაციის გადატანა ხდება *dataTransfer.getData()* მეთოდით. ეს მეთოდი დააბრუნებს იმავე ტიპის ინფორმაციას, რაც ჰქონდა მინიჭებული *setData()* მეთოდის დროს.

მაგალითი 1.29

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<style>
    #div1
    {
        width:293px;
        height:270px;
        padding:40px;
        border:6px solid #CC0000;
        background:#e5e5e5;
    }
</style>
<script>
function allowDrop(ev)
{
    ev.preventDefault();
}
function drag(ev)
{

```

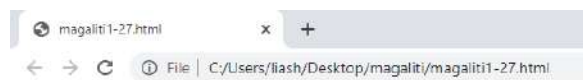
```

        ev.dataTransfer.setData("ტექსტი", ev.target.id);
    }
    function drop(ev)
    {
        ev.preventDefault();
        var data = ev.dataTransfer.getData("ტექსტი");
        ev.target.appendChild(document.getElementById(data));
    }
</script>
</head>
<body>
    <p> ჩავაგდოთ რაიმე სურათი: </p>
    <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
    </div>
    <br>
    
</body>
</html>

```

src="ლინკი" ჩასვით ნებისმიერი სურათი, რომელიც გსურთ, ჩვენს შემთხვევაში, საქართველოს ტექნიკური უნივერსიტეტის სტუდენტური თვითმმართველობის ოფიციალური ვებ გვერდიდან, გამოვიყენეთ ფოტო და ავტირთეთ ერთ სურათების ჰოსტინგზე <https://imgbb.com/>-ი ამ ეტაპზე ერთ ერთი საუკეთესოა ჰოსტინგი გახლავთ. ქვემოთ იხილეთ შედეგი

ჩავაგდოთ რაიმე სურათი:



ჩავაგდოთ რაიმე სურათი:



მოვლენები HTML 5-ში

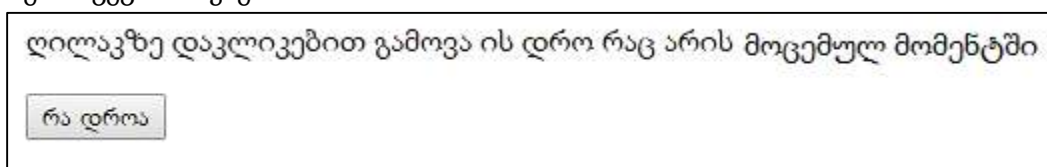
HTML5-ში მოვლენები განისაზღვრება იმით, თუ რა არის სკრიპტში გაწერილი. მაგალითად: მოვლენებში შეიძლება იყოს ღილაკზე დაკლიკება, დროის გამოტანა და ა. შ. HTML აძლევს უფლებას javascript-თან ერთად, რომ დაემატოს HTML-ის ელემენტებს

<რალაც ელემენტი რალაც ივენთი=' რალაც JavaScript'>

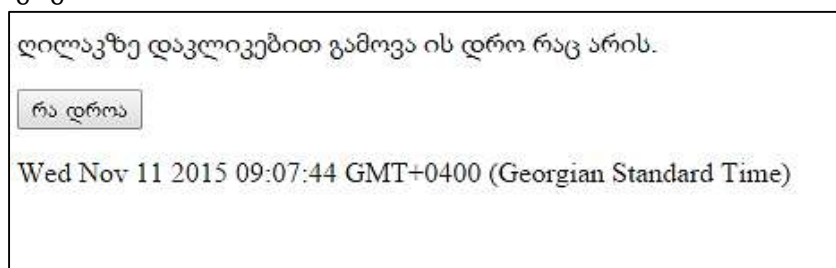
მაგალითი 1.30

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
<p>ღილაკზე დაკლიკებით გამოვა ის დრო, რაც არის მოცემულ მომენტში
</p>
  <button onClick="displayDate()">რა დროა </button>
<script>
function displayDate()
{
  document.getElementById("დრო").innerHTML = Date();
}
</script>
<p id="დრო">
</p>
</body>
</html>
```

ჩვენს შემთხვევაში იქნება:



დაკლიკების მერე:



HTML ის გლობალური ივენტები (მოვლენები)

Event	აღწერა
onchange	HTML ელემენტი შეიცვლება
onClick	მომხმარებელი დააკლიკებს
onmouseover	HTML ელემენტი შეიცვლება მაუსის მოძრაობისას
onmouseout	კურსორი მოშორდება იმ ელემენტს
onkeydown	დააჭერს კლავიატურის კომბინაციით
onLoad	ბრაუზერი მორჩება გვერდის ჩატვირთვას

Dropzone-ით შესაძლებელია ჩამოშლილი ტექსტის ან სურათის კოპირება და სხვა ადგილას გადატანა. სამწუხაროდ, ამ ატრიბუტს არცერთი ბრაუზერის მხარდაჭერა არ აქვს და ჯერ ჯეროებით დამუშავების ეტაპზეა გადავიდეთ შემდეგ **hidden** ატრიბუტზე.

hidden-რალაც ტექსტის დამალვა, მაგალითი საიტზე ავტორიზაციის შემდეგ უნდა დაიმალოს, შესვლის ფორმა, ასეთ შემთხვევებში გამოიყენება ეს ატრიბუტი.

სინტაქსი არის:

<ელემენტი hidden>

Lang

ელემენტის კონტენტის, რომელიც შეიცვას სხვადასხვა ან რომელიმე ენას და რეგიონს, პროგრამისტები მიუთითებენ და აღწერენ, რომ გასაგები იყოს ბრაუზერისათვის რომელ ენის/რეგიონის კოდთან (*Language Code*), აქვს საქმე.

სინტაქსი:

<ელემენტი Lang="ენის კოდი">

ქართული ენის კოდი, რომელიც მიეთითება არის *ka*. HTML 5-ში, ნებისმიერი ელემენტთან შეგვიძლია მივუთითოთ ის, რაც არ იყო წინამდებარე HTML 4.01 ვერსიაში.

მაგალითი 1.31

```
<!DOCTYPE html>
<html>
<body>
<p>აღწეროთ პარაგრაფი.</p>
<p Lang="ka">რაიმე ტექსტი</p>
</body>
</html>
```

HTML5 Lang ატრიბუტით დეკლარირდება (ანუ აღიწერება) როგორც მთლიანი, ისე ნაწილობრივი გვერდიც, რაც აძლევს ბრაუზერს, თუ საძიებო სისტემებს გვერდის სწრაფად წაკითხვის და პოვნის შესაძლებლობას, ასევე რეკომენდებულია W3C (*World Wide Web Consortium*) მიერ, რომ ნებისმიერი დოკუმენტი საკუთარი რეგიონის თუ ენის მიხედვით იქნას აღწერილი.

ქვემოთ არის აღწერილი ISO 639-1 Language Code ენის ISO კოდის მიხედვით, ISO იშიფრება, როგორც International Organization for Standardization (სტანდარტიზაციის საერთაშორისო ორგანიზაცია)

რომელიც დაარსდა 1943 წელს და მას მერე 2013 წლის მონაცემებით 164 ქვეყანაში მოქმედებს. ეს არის არასამთავრობო დამოუკიდებელი ორგანიზაცია, რომელიც აწესებს 164 ქვეყნისათვის საერთაშორისო სტანდარტებს, რომლის ცენტრალური ფილიალია ჟენევაში.

ISO 639-1	1967 (as ISO 639)	2002
ISO 639-2	1998	1998
ISO 639-3	2007	2007
ISO 639-4	2010-07-16	2010-07-16
ISO 639-5	2008-05-15	2008-05-15
ISO 639-6	2009-11-17	

რაც შეეხება ISO-639, როდესაც რაიმე სტანდარტი ან ენა ემატება ან ხდება გარკვეული ცვლილებები, რა თქმა უნდა, იცვლება კოდის დასახელებაც, საქართველო ISO 639-1 დან ფიქსირდება და ამოუღებლად დომინირებს შემდეგ ISO კოდებშიც.

ISO 639-1 ენის კოდები (არასრული)

<u>რეგიონი</u>	<u>ქვეყანა</u>	<u>ენა</u>	<u>639-1</u>	<u>639-2/T</u>	<u>639-2/B</u>
<u>northwest Caucasian</u>	<u>Abkhaz</u>	аԥсуа бызшәа, аԥсәәа	ab	abk	abk
<u>Afro-Asiatic</u>	<u>Afar</u>	Afaraf	aa	aar	aar
<u>Indo-European</u>	<u>Afrikaans</u>	Afrikaans	af	afr	afr
<u>Niger-Congo</u>	<u>Akan</u>	Akan	ak	aka	aka
<u>Indo-European</u>	<u>Albanian</u>	Shqip	sq	sqi	alb
<u>Afro-Asiatic</u>	<u>Amharic</u>	አማርኛ	am	amh	amh
<u>Afro-Asiatic</u>	<u>Arabic</u>	العربية	ar	ara	ara
<u>Indo-European</u>	<u>Aragonese</u>	aragonés	an	arg	arg
<u>Indo-European</u>	<u>Divehi, Dhivehi, Maldivian</u>	ދިވެހި	dv	div	div

<u>რეგიონი</u>	<u>ქვეყანა</u>	<u>ენა</u>	<u>639-1</u>	<u>639-2/T</u>	<u>639-2/B</u>
<u>Indo-European</u>	<u>Dutch</u>	Nederlands, Vlaams	nl	nld	dut
<u>Si- noTibetan</u>	<u>Dzongkha</u>	ཇོངཀ་	dz	dzo	dzo
<u>Indo-European</u>	<u>English</u>	English	en	eng	eng
<u>Constructed</u>	<u>Esperanto</u>	Esperanto	eo	epo	epo
<u>Uralic</u>	<u>Estonian</u>	eesti, eesti keel	et	est	est
<u>Niger-Congo</u>	<u>Ewe</u>	Ewe	ee	ewe	ewe
<u>Indo-European</u>	<u>Faroese</u>	føroyskt	fo	fao	fao
<u>Austronesian</u>	<u>Fijian</u>	vosa Vakaviti	fj	fij	fij
<u>Uralic</u>	<u>Finnish</u>	suomi, suomen kieli	fi	fin	fin
<u>Indo-European</u>	<u>French</u>	français, langue française	fr	fra	fre
<u>Niger-Congo</u>	<u>Fula</u> , <u>Fulah</u> , <u>Pulaar</u> , <u>Pular</u>	Fulfulde, Pulaar, Pular	ff	ful	ful
<u>Indo-European</u>	<u>Galician</u>	galego	gl	glg	glg
<u>South Caucasian</u>	<u>Georgian</u>	ქართული	ka	kat	geo
<u>Indo-European</u>	<u>Divehi</u> , <u>Dhivehi</u> , <u>Maldivian</u>	ދިވެހި	dv	div	div

spellcheck

დავუბრუნდეთ გლობალურ ატრიბუტებს, ამ ატრიბუტებშიც ტექსტის შემოწმების ატრიბუტს განვიხილავთ, რომელიც დამატებულია HTML 5-ში W3C-ის მიერ, რამდენად გრამატიკულად (ან მართებულად) არის დაწერილი ესა თუ ის სიტყვა. სინტაქსი:

<ელემენტი spellcheck="true ან false">



true შემთხვევაში შეამოწმებს



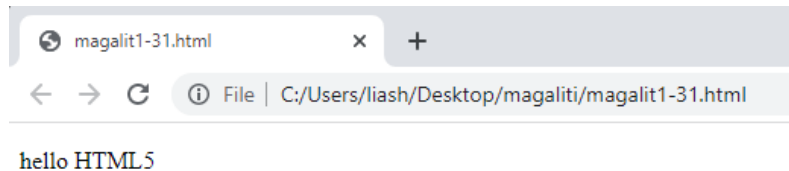
false არ შეამოწმებს.

ბრაუზერმა თუ არ აჩვენა, პარამეტრებიდან ჩაურთეთ და შემდეგ გამოჩნდება

მაგალითი 1.32:

```
<!DOCTYPE html>
<html>
<body>
<p contenteditable="true" spellcheck="true">
  hello HTML5
</p>
</body>
</html>
```

შედეგი



style



style არის ატრიბუტი, რომელიც თვითონ HTML ის დოკუმენტში აღწერს სტილს, მაგრამ ძირითად CSS სტილის ფაილში აღიწერება ხოლმე, ეს იმ შემთხვევაში გამოიყენება, თუ არ ვიყენებ CSS ფაილს.

სინტაქსი:

<ელემენტი style="სტილის განსაზღვრა ">

style ატრიბუტში გამოიყენება ძირითადად შემდეგი დამხარე ატრიბუტები, რომლებიც განსაზღვრავს:



background-color უკანა ფონის შესაქმნელად.



color ტექსტის ფერებისათვის.



font-family ტექსტის შრიფტის/ფონტის შესარჩევად.



font-size ფონტის ზომის გამოსაყენებლად.



text-align ტექსტის პოზიციისათვის

მაგალითი 1.33

```
<!DOCTYPE html>
<html>
<body style="background-color:Lightgrey">
  <h1 style="color:blue">
    ტექსტი ლურჯად
  </h1>
<h1 style="font-family:verdana"> verdana ფონტში მომცემულია ტექსტი
</h1>
  <p style="font-size:110%">
    ტექსტის ზომა 110% .
  </p>
<h1 style="text-align:center">
```

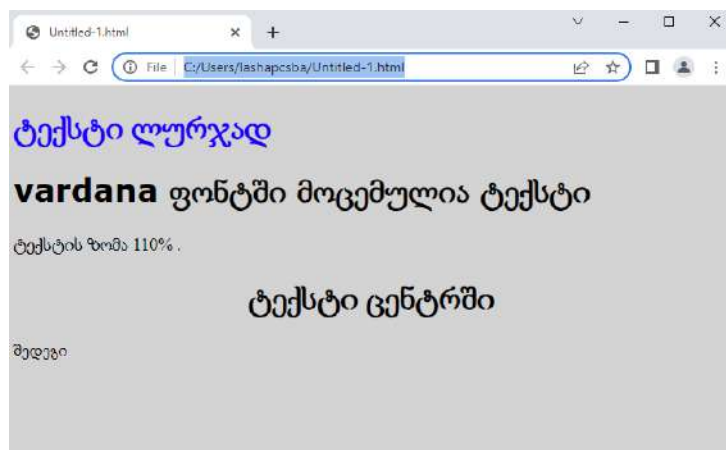
ტექსტი ცენტრში

</h1>

</body>

</html>

შედეგი



tabindex

ტაბის გამოყენებით შესაძლებელია კლავიატურაზე Tab კლავიშზე დაწოლით ნუმერაციის მიხედვით გადაადგილდება.

სინტაქსი:

<ელემენტი tabindex="რიცხვი">

მაგალითი 1.34

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
</head>
```

```
<body>
```

```
<a href="http://www.gtu.ge" tabindex="1">სტუ</a> |
```

```
<a href="https://leqtori.gtu.ge/" tabindex="3">ცხრილები</a> |
```

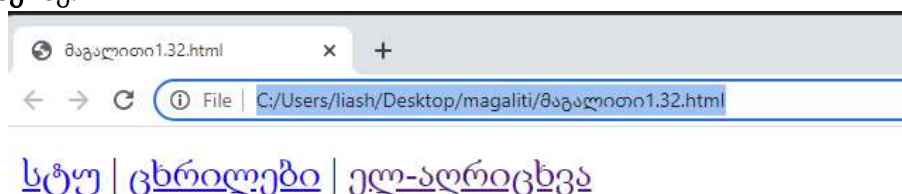
```
<a href="https://viciadmin.gtu.ge/" tabindex="2">ელ-აღრიცხვა</a>
```

```
</body>
```

```
</html>
```

შედეგი

ვხედავთ, რომ პირველი სტუ აირჩია, შემდეგ ცხრილებს-ზე გადავა და ბოლოს ელ-აღრიცხვაზე-ზე.



title

სათაურის მეშვეობით ხდება ტექსტის ელემენტის აღწერა, როგორც *abbr* ტეგის აღწერისას სკრიპტში დავინახავთ, რომ წერია:

```
<p>
  <a href="http://gtu.ge">
    <abbr title="საქართველოს ტექნიკური უნივერსიტეტი">სტუ
  </abbr>
</a>
</p>
```

Translate

ამ ატრიბუტის აზრი ის არის, რომ შეძლოს თარგმნოს ელემენტი, მისი სინტაქსი შეიცავს *yes* და *no* ჯერჯერობით ამ ატრიბუტის მხარდაჭერა არცერთ ბრაუზერს არ აქვს იმ მიზეზის და გამო, რომ ჯერ კიდევ დამუშავების ეტაპზეა.

სინტაქსი:

<ელემენტი *translate*="yes ან no ">

HTML ფორმები

Web ფორმები *HTML 5* ში მოიხსენება როგორც **Web Form 2.0**, სადაც დამატებულია ახალი



ველები და ფორმების კონტროლერები, პირველად ვებფორმები დამატებული იყო **Opera**-ს მიერ რომელიც შექმნა იან ჰიქსონმა, ოფიციალური თარიღი არის 2005 წლი 11 აპრილი.

ფორმები აღიწერება **<form>** ტეგში. მისი დამხარე ელემენტი არის **<input>** რომელიც განსაზღვრავს რაღაცის შეტანას.

ტეგი	აღწერა
<form>	განისაზღვრება <i>HTML</i> გვერდის ფორმა
<input>	შეტანის კონტროლი
<textarea>	მრავალსტრიქონიანი შეტანის არეალი
<label>	ჭდე
<fieldset>	აჯგუფებს და აკავშირებს ფორმაში
<legend>	სათაური არის <fieldset> ელემენტის
<select>	ჩამოსაშლელი ველი
<optgroup>	აჯგუფებს ჩამოსაშლელ ველში
<option>	საზღვრავს პარამეტრებს
<button>	დილაკი
<keygen>	განსაზღვრავს წყვილ გასაღებებს
<output>	გამოაქვს შედეგი

ფორმების დაწერა რამდენიმე ეტაპს მოიცავს, მართო შეიძლება მომხმარებლის ინტერფეისის დაწერა და გამოყენება, გამოყენებულ სერვერული მხარდაჭერა (*implementing the server-side processing*), რომელიც ურთერთქმედებს სერვერთან, ასევე არის მომხმარებლის ინტერფეისის კონფიგურაცია.

ელემენტი **<input>** რამდენიმე ვარიანტი შეიძლება ვიხილოთ, დამოკიდებულია type-ტიპზე, რომელიც თავის მხრივ მოიცავს:

text რომელიც განსაზღვრავს ტექსტის დაწერას;

radio განსაზღვრავს ღილაკებს, მონიშვნის ველს, რომლის არჩევისას შეიძლება იყოს ერთ ან რამდენიმე;

submit აღნიშნავს დასტურის ღილაკს, რომელიც, ველების შევსების შემდეგ შეგვიძლია დავადასტუროთ;

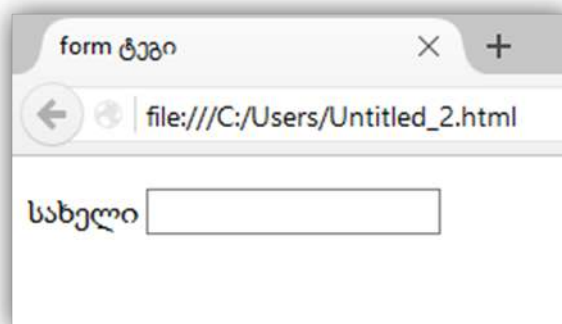
<form> სინტაქსი

<form>
რაც ელემენტები
</form>

მაგალითი 1.35

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8">
<title>form ტეგი</title>
</head>
<body>
<form>
<p><Label>სახელი <input>
</Label></p>
</form>
</body>
</html>
```

შედეგი



Label გვძლევს საშუალებას, გამოვაცხადოთ ტექსტი ან სიმბოლო. ქართულად ითარგმნება როგორც ჭდე, ხოლო **radio** ან **checkbox** ატრიბუტები გვძლევს შესაძლებლობას, რომ ავიჩიოთ ერთი ან რამდენიმე ვარიანტი, **<fieldset>** მეშვეობით

შესაძლებელია შეიქმნას სამუშაო ველი, ხოლო **<Legend>** მაგალითისათვის და გამოიყურება შემდეგნაირად:

მაგალითი 1.36

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>form ტეგი</title>
</head>
<body>
<form>
  <p>
    <label>სახელი: <input></label></p>
    <fieldset>
      <legend> აქ Legend ადგილია
    </legend>
    <p>
      <label>ეს label აქედან კი radio-> <input type=radio name=size> ტესტი
      1
    </label>
    </p>
  </fieldset>
  <br>
  <fieldset>
    <legend> აქ Legend ადგილია</legend>
    <p><label>ეს label აქედან checkbox-> <input type=checkbox> ტესტი 2
    </label></p>
  </fieldset>
</form>
</body>
</html>
```

შედეგი

form ტეგი

file:///C:/Users/data.html

სახელი:

აქ legend ადგილია

ეს label აქედან კი radio-> ☐ ტესტი 1

აქ legend ადგილია

ეს label აქედან checkbox-> ☐ ტესტი 2

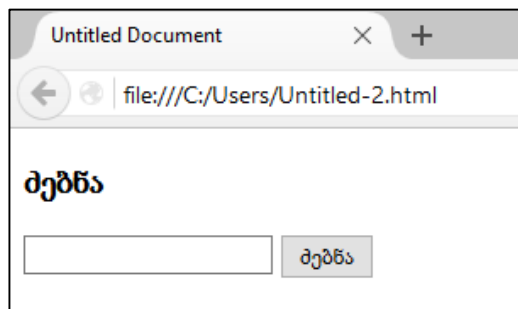
როგორც კი შევავსებთ ნებისმიერ ფორმას და დავადასტურებთ, იგზავნება მოთხოვნა სერვერზე და უკან იგი აბრუნდება შედეგს, ამ გადაცემის დროს გადაიცემა *name/value*, წყვილი. **Name** ატყობინებს ფორმის კონტროლიერი შესახებ, ხოლო *value* კი მომხმარებელმა გამოიყენა, მოინიშნა რაიმე *checkbox*-ზე თუ *radio*-ს მეშვეობით რაიმე სხვა მოქმედება შესრულდა, მაგალითი:

- ✔ თუ გვაქვს ექვსი ჩასაწრი ველი ე.წ. *textbox*-ი, ექვსივე ველს უნდა, ჰქონდეს აღწერა, თუ რომელი *textbox*-ი რომელს შესაბამება და რა ინფორმაციის მატარებელია.

მაგალითი 1.37

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>
ძებნა
</title>
</head>
<body>
<form action="http://www.siti.ge/search.aspx" method="get">
<h3>ძებნა
</h3>
<input type="text" name="საძიებო ტექსტი">
<input type="submit" value="ძებნა">
</form>
</body>
</html>
```

შედეგი



თვალსაჩინოებისათვის, რაიმე ჩვენთვის საინტერესო გვერდი, რომ ავიღოთ დავინახავთ, რომ მისი ფორმა მოიცავს ბევრ ქვეფორმებს როგორიცაა, ძებნის ფუნქცია, ჩაწერის ფუნქცია, მონიშვნის თუ დასტურის ფორმა, თუ გაქვს ერთი ან რამდენიმე ფორმა, ერთ გვერდზე მაინც სერვერზე იგზავნება როგორც ერთი მთლიანი ფორმა, ტრადიციულად გვერდებზე *<form>* ელემენტზე ზრუნავს ორი ატრიბუტი: *Action* და *Method*

action ატრიბუტი

action ატრიბუტის მიზანია, როდესაც ფორმა შეივსება ყველა ველის მონაცემით რათა გადაგზავნოს სერვერზე, მაგალითად; შესვლის ფორმის დროს ივსება სახელი (ან ელ-ფოსტა) და პაროლი, ერთ-ერთი რეალური საიტის მაგალითზე ვნახოთ, სადაც შესვლის გვერდი ვებსერვერზე იმყოფება .

```
<form class="ipsPad ipsForm ipsForm_vertical"
action="https://viciadmin.gtu.ge/#/Login/Login method="post"
accept-charset="utf-8" - validate="" data-ipsvalidation="">
```

შედეგი

სანამ სკრიპტს გავარჩევდეთ, ცხრილის სახით მოცემული გვაქვს ფორმის ატრიბუტები, ზოგიერთ მათგანს განვიხილავთ, ხოლო ზოგი ზემოთ იქნა განხილული.

ატრიბუტი	აღწერა
<i>accept-charset</i>	ფორმის დეკოდირება
<i>action</i>	სპეციალური მისამართი, სადაც ხდება ფორმის დადასტურება
<i>autocomplete</i>	ბრაუზერი, რომელიც თვითონ ასრულებს ფორმას
<i>enctype</i>	შიფრავს მონაცემს
<i>method</i>	HTTP მეთოდი (<i>default: GET</i>).
<i>name</i>	ფორმის ინდენტიფიკატორი
<i>novalidate</i>	არ იყოს ნებადართული
<i>target</i>	მისამართი, ანუ გვერდი გაიხსნას იმავე თუ ახალ <i>tab</i> -ში

რაც შეეხება მაგალითის სკრიპტს, *class*-ებზე ზემოთ ვილაპარაკეთ ხოლო *accept-charset* ამ ატრიბუტით ვხდებით, რომ დეკოდირება ხდება *utf-8* ში, ამ ატრიბუტი ფორმებში გამოყენება.

სინტაქსი:

```
<form accept-charset="character_set">
```

ხოლო *novalidate* ატრიბუტის მეშვეობით *დასტურის* შემდეგ მონაცემები ანუ ცარიელ ველები, თუ დავაწვებით *დასტურ* ღილაკს, შეტყობინებას გამოგვიტანს, რომ ველები შევსებული არ არის.

სინტაქსი:

`<form novalidate>`

method ატრიბუტი

ამ ატრიბუტის მეშვეობით ინფორმაცია იგზავნება სერვერზე ორი საშუალებით, რომელიც დაკავშირებულია HTTP მეთოდთან:

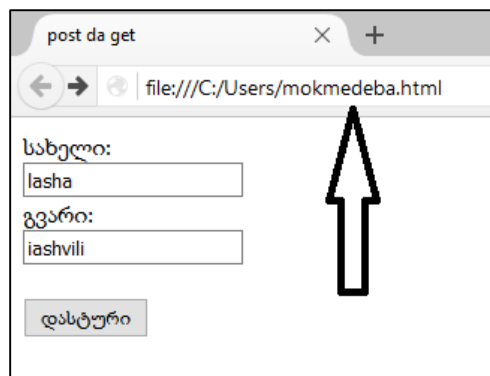
- *get* მეთოდით, რომელიც აგზავნის მონაცემებს, წარმოდგება როგორც ბმულის (URL) ნაწილი.
- *post* მეთოდი, კი მალავს ინფორმაციას HTTP სათაურში;

name ატრიბუტი

name ატრიბუტების არსი არის ის, რომ ყველა *input* ტიპს უნდა ჰქონდეს თავისი *name*-სახელი მინიჭებული.

მაგალითი

`mokmedeba?saxeli=Lasha&GVARI=iashvili`



action-ში აღწერილს პირობითად დავარქვათ *mokmedeba*, დავაკვირდეთ, დაკლიკების მერე „დასტურ“-ზე ბმულს, როგორ შეიცვლება მაგალითი 1.38



მაგალითი 1.38

```
<!DOCTYPE html>
<html>
<head>
<title>post da get</title>
  <meta charset="utf-8">
</head>
<body>
  <form action="mokmedeba">
    სახელი:
    <br>
    <input type="text" name="saxeli" value="lasha">
```

```

        <br>
გვარი:
        <br>
        <input type="text" name="GVARI" value="iashvili">
        <br>
        <br>
        <input type="submit" value="დასტური">
    </form>
</body>
</html>

```

enctype ატრიბუტი

HTTP post მეთოდის საშუალებით მონაცემებს აგზავნის სერვერზე, ამ დროს გამოიყენება *enctype* ატრიბუტი, რომელიც ბრაუზერში ახდენს მონაცემების ენკოდირებას. ეს ხდება მანამ, სანამ მონაცემებს გადააგზავნის სერვერზე, მაგრამ ბრაუზერს, როგორც წესი, აქვს სამი ენკოდირების მხარდაჭერა:

- *application/x-www-form-urlencoded* სტანდარტულად ბრაუზერი ამ მეთოდს იყენებს ისეთი სიმბოლოები, რომლის დროს არ შეადგენს ალფავიტურ სიმბოლოებს და მხოლოდ მოიცავს ისეთ სიმბოლოებს, როგორიცაა პლუს ნიშანი, გაყოფა ან თუნდაც კლავიატურაზე ე. წ space-ი, Web სერვერზე, სანამ გადაიგზავნება ყველა სიმბოლოს გარდაქმნის „+“ ნიშნად, ხოლო სპეციალური სიმბოლოები, რომელიც შესაძლებელი არის, მათი დეკოდირება გარდაქმნის *ASCII HEX* სიდიდების მიხედვით.
- *multipart/form-data* იძლევა შესაძლებლობას, რომ გადაიგზავნოს მონაცემები ნაწილებად და თითოეული ნაწილი საბოლოო ჯამში წარმოადგენს მთლიან ფორმას, მაგალითისათვის ავიღოთ ფაილების (მაგალითი სურათის) ატვირთვა სერვერზე. ეს ატრიბუტი არ ახდენს სიმბოლოების დეკოდირებას, სერვერისათვის გასაგები უნდა იყოს სერვერზე რა ტიპის მონაცემი იტვირთება.
- *text/plain* აგზავნის მონაცემებს სერვერზე, როგორც არამოდულიზებული გარდაქმნის რა ცარელ სივრცებს „+“ ეზად და უცვლელად ტოვებს *ASCII* სიმბოლოებს. სინტაქსი არის:

```
<form enctype="მეთოდი">
```

autocomplete ატრიბუტი

ამ ატრიბუტის მითითება გვამდევს იმის შესაძლებლობას, რომ ველები, რომლებიც დამახსოვრებული აქვს სახელი, გვარი, ტელეფონი ან თუნდაც პაროლი ველებში ავტომატურად ჩნდება მისი სინტაქსია:

```
<input autocomplete="on ან off">
```

უნდა აღინიშნოს, რომ ზოგ ბრაუზერს სჭირდება პარამეტრებიდან ჩართვა, რათა წაითხოს, ამ ატრიბუტი აქვს ორი თვისება *on*, რომელიც სტანდარტულად სულ ჩართულია და გაკვეთილი პირობებისა თუ ამოცანების გათვალისწინებელით აქვს *off* ფუნქცია.

ეს ატრიბუტი მუშაობს *<input>* ისეთ ტიპებთან, როგორებიც არის: *text*, *search*,

url, tel, email, password, datepickers, range, და color ამ ტიპებს ცოტა ქვემოთ აღწერთ, რაც შეეხება მაგალითს, როგორც ქვემოთ დავინახავთ კლავიატურაზე დაწოლით დამახსოვრებულ ინფორმაციას ამოგვიგდებს.

მაგალითი 1.39

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
  <form action=" " method="get" autocomplete="on">
    სახელი:<input type="text" name="saxeli"><br>
    E-mail: <input type="email" name="email"><br>
    <input type="submit" name="დასტური">
  </form>
</body>
</html>
```

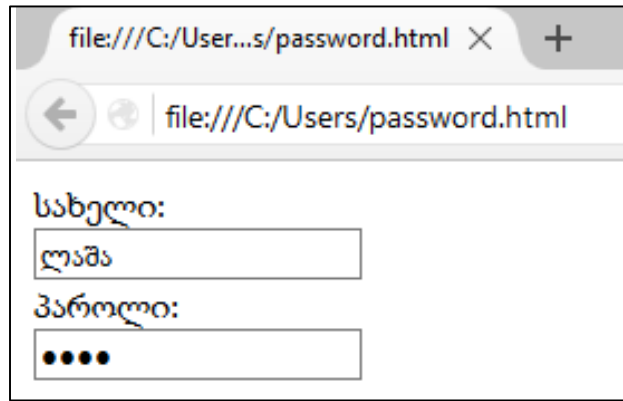
ტექსტის შეტანა ველებში(*textbox*) შეგვიძლია ნებისმიერ ვებგვერდებზე. ყველა ველს აქვს თავის დანიშნულება და შესაბამისად გვპასუხობს იმ მოთხოვნაზე, რასაც ვავსებთ, საერთოდ არსებობს, კონტროლერი ფორმის შევსების სამი ტიპი:

- ✦ *Single-Line* კონტროლერი-ერთ სტრიქონიან ჩასაწერი ველი, ეს შემთხვევა გამოყენება როგორც მარტო <input> ელემენტთან, ისე *textbox*-ებთან.
- ✦ *Password* კონტროლერი-ძალიან ჰგავს ერთ სტრიქონიან შეტანის ველს. განსხვავება მხოლოდ ის არის, რომ ტექსტს რომ შევიტანთ გამოჩნდება, ფიფქებით ან შავი წერილებით. ამ კონტროლერს ძირითად გამოყენებენ საიტზე ავტორიზაციისას, როდესაც სახელს და პაროლს მიუთითებენ.

მაგალითი 1.40

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <form action="">
    სახელი:<br>
    <input type="text" name="სახელი">
    <br>
    პაროლი:<br>
    <input type="password" name="პაროლი">
  </form>
</body>
</html>
```


შედეგი



➤ *Multiline*-მრავალ სტრიქონიანი კონტროლერი არის ის, სადაც შეგვიძლია ერთ სტრიქონზე მეტი ჩაწერა, ამისთვის გამოიყენებენ `<textarea>` ელემენტს.

HTML 5 აქვს დამატებული ბევრი ახალი `<input>` ტიპი, რომელიც ზოგადად აღწერენ ინფორმაციას Web გვერდზე:

- ➔ **color**: ფერების არჩევა;
- ➔ **data**: კალენდარული თარიღის შეტანა;
- ➔ **datetime**: დროისა და რიცხვის შეტანა *Greenwich*-ის დროით;
- ➔ **datetime-local**: ადგილობრივი დრო და თარიღი;
- ➔ **email**: ელ-ფოსტის მითითება, შესალებელია როგორც ერთ სტრიქონიანი, ისე მრავალ სტრიქონზე მითითებით;
- ➔ **month**: თვისა და წლის მითითება;
- ➔ **number**: რიცხვის შეტანა;
- ➔ **range**: შეტანილი ინფორმაციის გადადგილება, როგორც სლაიდად;
- ➔ **search**: ძებნა;
- ➔ **tel**: ტელეფონის რიცხვი;
- ➔ **time**: საათი, წუთების და წამების აღწერა;
- ➔ **url**: ვებსაიტის მისამართი;
- ➔ **week**: ასახავს არსებულ კვირას.

მაგალითი 1.41

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> ტიპები </title>
</head>
<body>
<div>
  <h3> ფერები
</h3>
```

```

        <input type="color" name="color">
    </div>
    <div>
        <h3> კალენდარული თარიღის შეტანა </h3>
        <input type="date" name="date">
    </div>
    <div>
        <h3> დროისა და რიცხვის შეტანა Greenwich-ის დროით.
    </h3>
        <input type="datetime" name="datetime">
    </div>

    <div>
        <h3> ადგილობრივი დრო და თარიღი.
    </h3>
        <input type="datetime-local" name="datetime-local">
    </div>
    <div>
        <h3> Email
    </h3>
        <input type="email" name="email">
    </div>
    <div>
        <h3> თვისა და წლის მითითება </h3>
        <input type="month" name="month">
    </div>
    <div>
        <h3> რიცხვის შეტანა. </h3>
        <input type="number" name="number">
    </div>
    <div>
        <h3> :შეტანილი ინფორმაციის გადადგილება, როგორც სლაიდად. </h3>
        <input type="range" id="range" name="range">
        <output for="range" id="output"></output>
    </div>
    <div>
        <h3> ძებნა </h3>
        <input type="search" name="search" results="5" autosave="saved-
searches">
    </div>
    <div>
        <h3> ტელეფონი </h3>
        <input type="tel" name="tel">
    </div>
    <div>
        <h3> საათი, წუთებისა და წამების აღწერა. </h3>
        <input type="time" name="time">
    </div>
    <div>

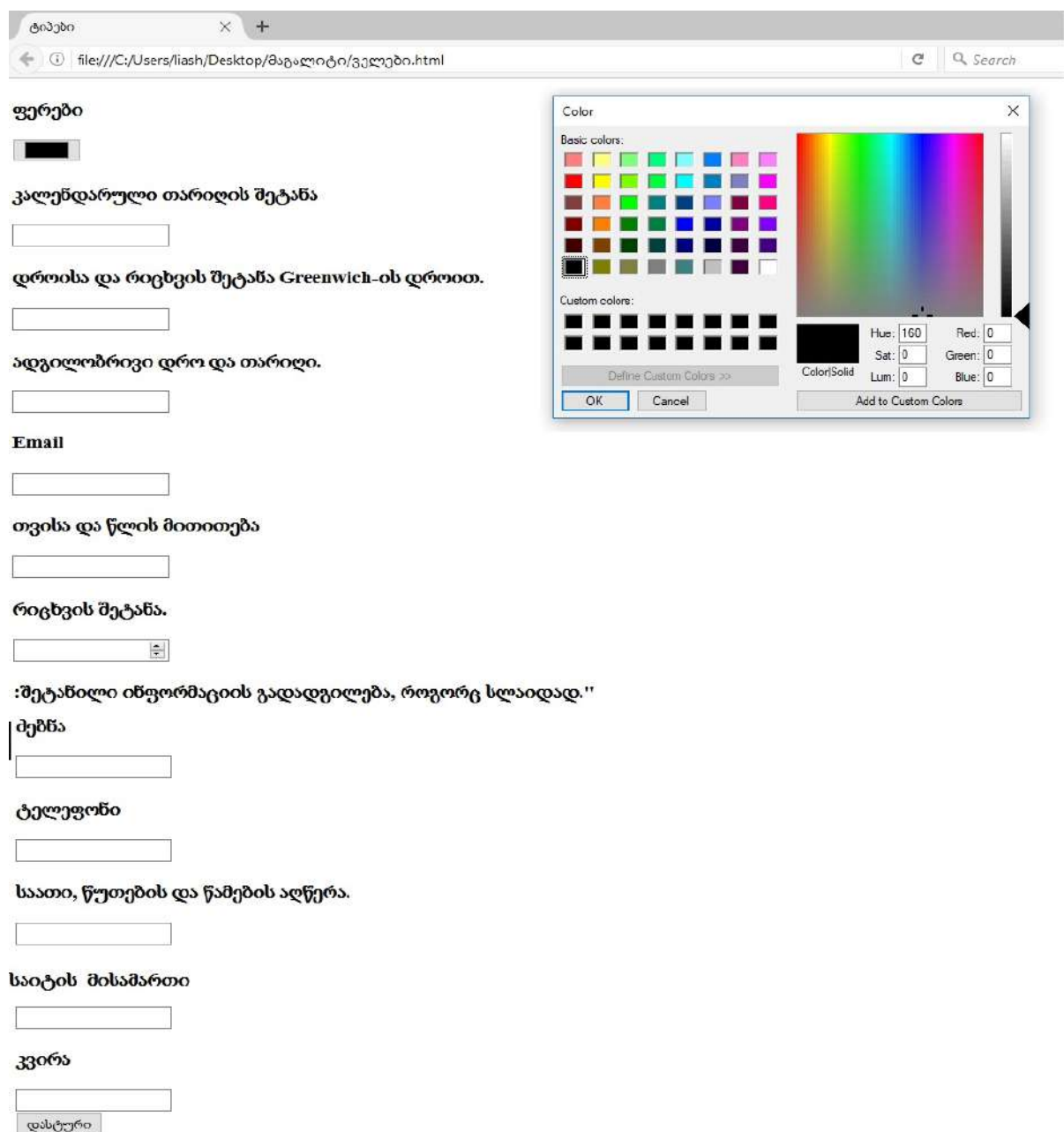
```

```

        <h3> საიტის მისამართი საიტის </h3>
        <input type="url" name="url">
    </div>
</div>
        <h3>კვირა</h3>
        <input type="week" name="week">
    </div>
</div>
        <input type="submit" value="დასტური">
    </div>
</body>
</html>

```

შედეგი



ტოპუზი

კალენდარული თარიღის შეტანა

დროისა და რიცხვის შეტანა Greenwich-ის დროით.

ადგილობრივი დრო და თარიღი.

Email

თვისა და წლის მითითება

რიცხვის შეტანა.

:შეტანილი ინფორმაციის გადადგილება, როგორც სლაიდად."

ძეზნა

ტელეფონი

საათი, წუთების და წამების აღწერა.

საიტის მისამართი

კვირა

დასტური

`<input>` ელემენტი, როდესაც იყენებს `type` ატრიბუტს და მინიჭებული აქვს `text`. იგი შეგვიძლია ატრიბუტი შეუძლია განავრცო და გამოიყენო სხვა ატრიბუტებშიც როგორიცაა:

- ყველა უნივერსალური ატრიბუტი;
- `disabled`, `readonly`, `form`, `autocomplete`, `autofocus`, `list`, `pattern`, `required` და `dirname`.

ცხრილში მოცემულია სტანდარტული ტექსტის შესაყვანი ატრიბუტები.

ატრიბუტი	აღწერა
<code>name</code>	სახელობითი ატრიბუტი, რომელსაც ენიჭება <code>name/value</code> წყვილი იგზავნება სერვერზე. დავამახსოვროთ , რომ თითოეული <code>name/value</code> წყვილის წარმოდგენა არის ფორმის კონტროლისათვის, სადაც <code>name</code> -სახელი ინდეფიცირებას ახდენს ფორმის კონტროლერის და <code>value</code> არის, რასაც მომხმარებელი შეიყვანს.
<code>value</code>	სიდიდე ან ცვლადი, რომლის საშუალებით ფორმა იტვირთება.
<code>size</code>	სიმბოლოების შეყვანის რაოდენობა, რომელიც განსაზღვრავს, თუ რამდენი სიმბოლო უნდა შეიყვანოს მომხმარებელმა, თუ მითითებულია 10 სიმბოლო, ხოლო მომხმარებელმა შეიყვანა 10 ზე მეტი, მას აღარ ჩაწერს.

მრავალ სტრიქონიანი ველი იქმნება `<textarea>` ტეგის მეშვეობით, რომელსაც აქვს დამხურავი ტეგი `</textarea>`, ამ ტეგში შეგვიძლია განუსაზღვრელი სხვადასხვა სიმბოლოს აკრეფა. მისი ტექსტის არეალს განსაზღვრავს მისივე `col` და `row`, რომელიც სტანდარტულია, თუ რა ზომის გვინდა იყოს ტექსტის არეალი, მაგალითად: ავიღოთ `col=70`, ხოლო `row=50`, მაშინ გვექნება ასეთი სახე

მაგალითი 1.42

```

<!DOCTYPE html>
<html>
<head>
<title> სტრიქონი </title>
  <meta charset="utf-8">
</head>
<body>
<textarea name="textarea" cols="30" rows="10">
მრავალ სტრიქონიანი ველი
</textarea>
</body>
</html>

```



ვხედავთ, რომ ტექსტის არეალი სტანდარტულად, როგორც ჩანს, საერთოდ ტექსტის ფორმის საკითხები CSS მეშვეობით წყდება. ცხრილში მოცემულია `<textarea>` ტეგის დამატებითი ატრიბუტები:

ატრიბუტი	აღწერა (გამოყენება <code>input</code> ტეგშიც)
<i>autofocus</i>	ლოგიკური ატრიბუტი, რომელიც მიუთითებს, რომ ელემენტი უნდა იყოს ფოკუსირებული, როცა გვერდი იტვირთება
<i>cols</i>	აღწერს <code>textarea</code> სიგანეს.
<i>disabled</i>	მიუთითებს, რომ მოცემული <code>textarea</code> უნდა დაიმალოს.
<i>form</i>	<code><form></code> ელემენტი აღწერს, <code><textarea></code> -ს და ურთიერთქმედებს <code>id</code> საშუალებით
<i>maxLength</i>	მაქსიმალური რაოდენობა სიტყვების.
<i>placeholder</i>	მოკლედ აღწერს ტექსტური ველის დანიშნულებას.
<i>readonly</i>	ტექსტური ველი, სადაც კითხვით მხოლოდ რეჟიმია.
<i>required</i>	ველი, რომელიც აუცილებად უნდა შეივსოს.
<i>rows</i>	ტექსტური ველის რიგი, აღწერს სიგრძეს.
<i>wrap</i>	ტექსტური ველი, როგორ უნდა იქნეს ჩაკეცილი/ჩახვეული.

განვიხილოთ თითოეული ატრიბუტი და მათი სინტაქსი. ისეთი ატრიბუტი როგორიც არის `name`, `value`, `size`, `col` და `row` აღარ ვლაპარაკობთ, რადგან არა ერთხელ შევხდით ამ ატრიბუტებს:

autofocus

HTML 5 ში ჩაემატა ახალი ატრიბუტი, მისი მოვალეობა არის, რომ ფოკუსირებული იყოს, თუ როდის მოხდება გვერდების ჩატვირთვა. მაგალითად, ავიღოთ ფორმა, სადაც აღწერილია, სახელი, პაროლი ელ-ფოსტა, მივუთითოთ, რომ, როდესაც გვერდს ჩატვირთავს (გახსნის) ავტომატურად მაუსის კურსორი ელ-ფოსტაზე იდგეს.

სინტაქსი:

`<textarea autofocus>`

ასევე ატრიბუტის გამოყენება ხდება `<input>` ტეგშიც

სინტაქსი:

`<input autofocus>`

მაგალითი 1.43

```
<!DOCTYPE html>
```

```
<html Lang="ka">
```

```
<head>
```

```
<meta charset="utf-8">
```

```

<title>autofocus ატრიბუტი </title>
</head>
<body>
  <h2>autofocus გაცნობა</h2>
  <form method="get">
<table>
<tr>
<td>სახელი </td>
<td>
<input type="text" Name="სახელი"></td>
</tr>
<br>
<tr><td>პაროლი </td>
<td><input type="password" name="პაროლი" >
</td></tr>
<br>
<tr><td>Email </td>
<td>
<input type="email" name="ელ-ფოტა" autofocus>
</td>
</tr>
<tr>
<td>დასტური </td>
<td><input type="submit" value="დასტური">
</td></tr>
</table>
</form>
</body>
</html>

```

შედეგი

autofocus ატრიბუტი

file:///C:/Users/autofocus.html

autofocus გაცნობა

სახელი

პაროლი

Email

დასტური

disabled

ველი რომელიც არ გვინდა იყოს აქტიური, შეგვიძლია გამოვიყენოთ *disabled* ატრიბუტი, ამ ატრიბუტს იყენებს იმ შემთხვევაში, თუ წინ რაიმე გარკვეული პირობა აქვს, მაგალითად, რაიმე უნდა მოინიშნოს და შემდეგ გააქტიურდეს, მაგრამ ეს ყველაფერი *JavaScript* ურთერთობისას იქნება, ხოლო სტატიკური გვერდებისას მხოლოდ გაუქმებულად გამოჩნდება.

მისი სინტაქსი არის:

`<textarea disabled>`

ან

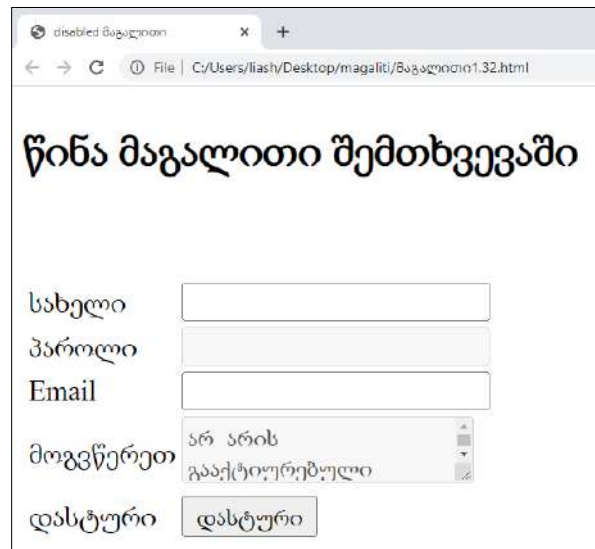
`<input disabled>`

წინა მაგალითში ჩავამატოთ `<textarea>` და პაროლის ველი გავაუქმოთ.

მაგალითი 1.44

```
<!DOCTYPE html>
<html lang="ka">
  <head>
    <meta charset="utf-8">
    <title>disabled მაგალითი </title>
  </head>
  <body>
    <h2>წინა მაგალითი შემთხვევაში</h2>
    <form method="get">
      <table>
        <tr>
          <td>სახელი </td><td><input type="text" name="სახელი"></td>
        </tr><br>
        <tr>
          <td>პაროლი </td><td><input type="password" name="პაროლი" disabled >
          </td></tr>
        <br>
        <tr><td>Email </td><td><input type="email" name="ელ-ფოსტა" autofocus>
        </td>
        </tr>
        <tr>
          <td>მოგვეჩერეთ </td><td><textarea disabled>არ არის
          გააქტიურებული</textarea>
          </td>
        </tr>
        <tr><td>დასტური </td><td><input type="submit"
        value="დასტური"></td></tr>
      </table>
    </form>
  </body>
</html>
```

შედეგი



form

ფორმის განსაზღვრა ხდება `id` საშუალებით, ანუ `form`-შიც რა `id`-საც მიანიჭებს ის, უნდა გამოცხადეს `<textarea>`-ში, ფორმის გარეთ მდებარეობს, მაგრამ მას ეკუთვნის.

სინტაქსი:

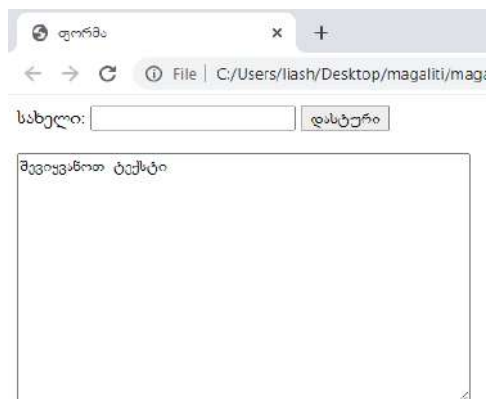
```
<textarea form="ფორმის_id">
```

შევქმნათ `textbox` და `<textarea>`, სადაც აღწერილი იქნება ფორმის `id` ელემენტი

მაგალითი 1.45

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
<title> ფორმა</title>
</head>
<body>
<form action="/" id="ფორმა">
  სახელი: <input type="text" name="სახელი">
  <input type="submit" value="დასტური">
</form>
<br>
  <textarea rows="14" cols="50" name="saxeli" form="ფორმა">
    შევიყვანოთ ტექსტი
  </textarea>
</body>
</html>
```


შედეგი



maxLength

ამ ატრიბუტის საშუალებით შესაძლებელია მაქსიმალური სიმბოლოების დაწერა, ანალოგიური ველის შექმნა და დაწერაც შეიძლება `<input>` ტეგშიც
სინტაქსი:

```
<textarea maxLength="რიცხვი">
```

დავწეროთ ისეთი მაგალითი, სადაც შეუძლებელი იქნება `<textarea>`-ში 100 სიმბოლოზე მეტი რომ დაიწეროს, ხოლო ტექსტურ ველში ე. წ. `textbox`-ებში კი 20 სიმბოლოზე მეტი იყოს, დაუშვებელი.

მაგალითი 1.46

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> maxLength </title>
</head>
<body>
  <form action="http://www.scripts.ge" method="post">
    <p>სახელი:<br>
    <input type="text" name="სახელი" value="" size="20" maxLength="20">
    </p>
    <p>ტელეფონი:<br>
    <input type="tel" name="ტელეფონი" value="" size="20"
    maxLength="20">
    </p>
    <p>Email:<br>
    <input type="email" name="Email" value="" size="20" maxLength="20">
    </p>
    <p>საცვარელი ფერი:<br>
    <input type="color" name="ფერები" value="" size="20"
    maxLength="130"></p>
    <p>დაბადების თარიღი:<br>
    <input type="date" name="თარიღი" value="" size="20" maxLength="20"></p>
    <textarea cols="50" maxLength="100"> ჩაწერეთ რაიმე ტექსტი.... </textarea>
    <p><input type="submit" value="დასტური"></p>
  </form>
```

</body>

</html>

შედეგი

სახელი:

ტელეფონი:

Email:

საყვარელი ფერი:

დაბადების თარიღი:

ჩაწერეთ რაიმე ტექსტი....

დასტური

placeholder

ტექსტური ველი გვეუბნება ტექსტის დანიშნულებას, სად რა უნდა ჩაიწეროს. მაგალითად სად უნდა მოხდეს სახელისა და გვარის ან პაროლის თუ ელ-ფოსტის ჩაწერა, მას მერე, რაც კურსორს „ჩავაკლიკებთ“, *placeholder*-ის ტექსტის გაქრება.

ეს ატრიბუტი მუშაობს <input> ისეთ ტიპებთან, როგორებიც არის: *text, search, url, tel, email*, და *password*.

სინტაქსი:

```
<textarea placeholder="ტექსტი რომელიც უნდა გამოჩდეს">
```

მაგალითი 1.47

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title> placeholder</title>
```

```
</head>
```

```
<body>
```

```
<form action="http://www.scripts.ge" method="post">
```

```
<p>სახელი:<br>
```

```
<input type="text" name="txtName" value="" size="20" maxLength="20"
placeholder="სახელი">
```

```
</p>
```

```
<textarea cols="50" maxLength="10" placeholder="ჩაკლიკების მერე გაქრება ეს
ტექსტი">
```

```
</textarea>
```

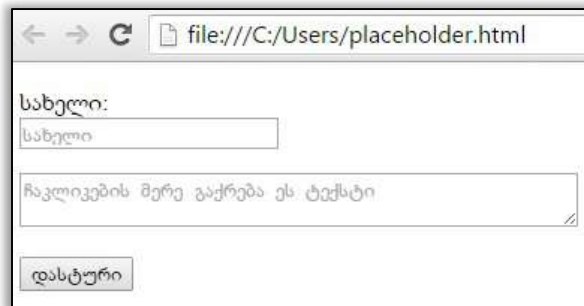
```
<p><input type="submit" value="დასტური">
```

```

</p>
</form>
</body>
</html>

```

შედეგი



readonly

ტექსტურ ველში ჩაწერილი ტექსტს მხოლოდ წაკითხვის რეჟიმში იმყოფება, მაშინ ცვლილებების შეტანის უფლება არ გვაქვს, (წაშლის ან რედაქტირების) ასეთი ტიპის ველებში გამოიყენება *readonly* ატრიბუტი.

სინტაქსი:

```
<textarea readonly>
```

მაგალითი 1.48

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>readonly</title>
```

```
</head>
```

```
<body>
```

```
<form action="gtu.ge">
```

```
<p>
```

```
<textarea rows="10" cols="20" readonly>
```

მოცემული ტექსტის რედაქტირება შეუძლებელია, მხოლოდ წაკითხვა

შესაძლებელი

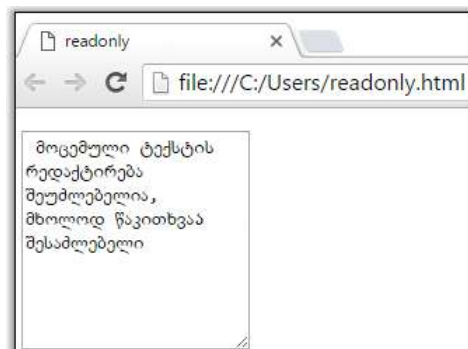
```
</textarea></p>
```

```
</form>
```

```
</body>
```

```
</html>
```

შედეგი



required

ფორმის შევსებისას გვხდება ხშირად ისეთი ველები, რომელიც აუცილებელია, რომ შეივსოს და მისი გამოტოვება არ შეიძლება, ამისთვის HTML 5-ში ჩამატებულია სპეციალური *required*-ატრიბუტი, რომელიც მოითხოვს კონკრეტულის ველის შევსებას, ეს ველი შეიძლება იყოს სახელი და გვარი ან ტელეფონი.

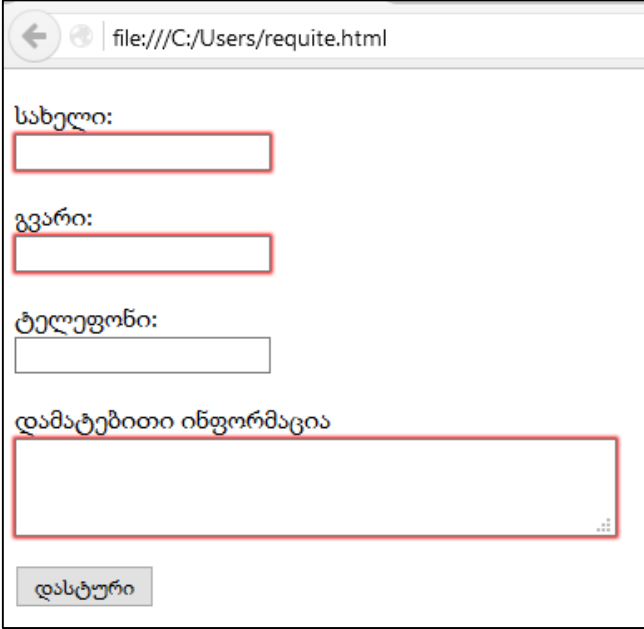
სინტაქსი:

`<textarea required>`

მაგალითი 1.49

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> required </title>
</head>
<body>
<form action="http://www.gtu.ge" method="post">
<p>სახელი:<br>
<input type="text" name="სახელი" value="" size="20" maxLength="20"
required></p>
<p>გვარი:<br>
<input type="text" name="გვარი" maxLength="20" required></p>
<p>ტელეფონი:<br>
<input type="tel" name="ტელეფონი" value="" size="20" maxLength="20"></p>
<p> დამატებითი ინფორმაცია<br>
<textarea name="დამატებითი ინფორმაცია" cols="50" required></textarea>
<p><input type="submit" value="დასტური"></p>
</form>
</body>
</html>
```

შედეგი



სახელი:

გვარი:

ტელეფონი:

დამატებითი ინფორმაცია

დასტური

სურათიდან ჩანს, რომ ველზე უბრალოდ „დასტურს“ თუ დავაწვებით, მიგვანიშნებს, თუ რომელი ველი უნდა შეივსოს. ფორმის შევსებისას ხშირად ფიფქიტაც(*) აღნიშნავენ, რომ აუცილებელია მისი შევსება. სურათზე ჩანს მეორე შემთხვევა, სადაც რომელი ველი შევსო და თუ გამოგვრჩა შევსება, მაშინ შეტყობინებას ამოგვიგდებს „*please fill out this field*“ რომელიც მიგვითითებს, რომ მისი შევსება აუცილებელია.

wrap

ითარგმნება, როგორც ხვეულა, ამ ტერმინს გამოვიყენებთ ჩვენც, *wrap* საშუალებით შესაძლებელი არის `<textarea>`-ში თუ ბევრი ტექსტი წერია ჩნდება ხვეულა, რომლის საშუალებით ქვემოთ შესაძლებელია ჩამოიწიოს ტექსტი:

სინტაქსი:

`<textarea wrap="soft ან hard">`

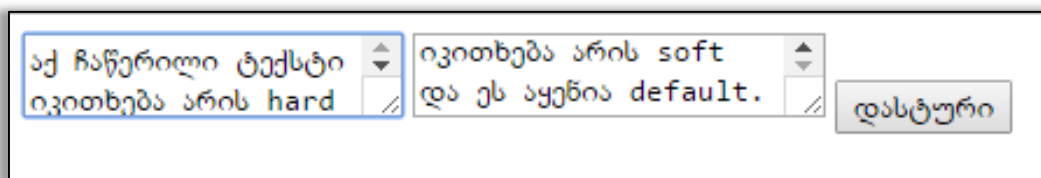
soft შემთხვევაში, როდესაც მომხარებელი ტექსტს კითხულობს და გადადის ახალ სტრიქონზე. სერვერზე იგზავნე ინფორმაცია, რომ მომხარებელი ისევ კითხულობს იმავე სტრიქონს

hard შემთხვევაში მომხარებელი ყოველი დაჭერისას სერვერზე იგზავნება, რომ კითხულობს ახალ სტრიქონს.

მაგალითი 1.50

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
  <form action="dawyeba.html">
    <textarea rows="2" cols="20" name="text1" wrap="hard">
      აქ ჩაწერილი ტექსტი იკითხება არის hard
    </textarea>
    <textarea rows="2" cols="22" name="text2" wrap="soft"> აქ ჩაწერილი
      ტექსტი იკითხება არის soft და ეს აყენია default.</textarea>
    <input type="submit" value="დასტური">
  </form>
</body>
</html>
```

შედეგი



The screenshot shows a web form with two dropdown menus. The first dropdown menu is labeled 'აქ ჩაწერილი ტექსტი' and has 'იკითხება არის hard' selected. The second dropdown menu is labeled 'იკითხება არის soft' and has 'და ეს აყენია default.' selected. To the right of the dropdown menus is a button labeled 'დასტური'.

<option>

ჩამოსაშლელი სიის საშუალებით, მომხმარებლებს ეძლევათ შესაძლებლობა, რომ აირჩიოს ფერები, სასურველია ავტომობილი, ქვეყანა ან სხვა, ასეთი მენიუს შესაქმნელად საჭიროა გამოვიყენოთ <option>, რომელიც მდებარეობს <select> და <datalist>-ის შიგნით.

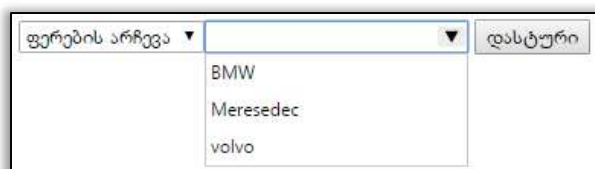
მოვიყვანოთ მაგალითი, სადაც <select>-ის გამოყენებით შესაძლებელი იქნება ფერების არჩევა, ხოლო იმავე ფორმაში შევქმნათ არჩევითი სია საიტების <datalist> გამოყენებით, <datalist> არის <input> პარამეტრები.

<datalist list="saxeli"> მიუთითეთ, ის უნდა გამოცხადდეს id-შიც

მაგალითი 1.51

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<form action="ragaca.html">
<select name="perebi">
  <option selected="selected" value="">ფერების არჩევა </option>
  <option value="red">წითელი </option>
  <option value="green">მწვანე </option>
  <option value="blue">ლურჯი </option>
  <option value="yviteli">ყვითელი </option>
</select>
<input list="მანქანა" name="">
<datalist id="მანქანა">
  <option value="BMW">
  <option value="Meresedec">
  <option value="volvo">
</datalist>
<input type="submit" value="დასტური">
</form>
</body>
</html>
```

შედეგი



The screenshot shows a web form with a dropdown menu labeled 'ფერების არჩევა' and a button labeled 'დასტური'. The dropdown menu is open, showing a list of car brands: BMW, Meresedec, and volvo.

ამასთან ერთად არის `<optgroup>` ტეგი, რომელიც გამოარჩევს ჩამოსაშლელ მენიუს, ქმნის ჯგუფს და წარმოადგენს მომხარებლისათვის დაჯგუფებულად:

მაგალითი 1.52

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> optgroup</title>
</head>
<body>
<form action="http://gtu.ge" method="post">
<label> ავტომობილები</label>
<select>
  <optgroup label="მსუბუქი">
    <option>Golf3 </option>
    <option>Vectra </option>
  </optgroup>
  <optgroup label="სამგზავრო">
    <option>მიკრო ავტობუსი</option>
    <option>ავტობუსი</option>
  </optgroup>
</select>
</form>
</body>
</html>
```

შედეგი



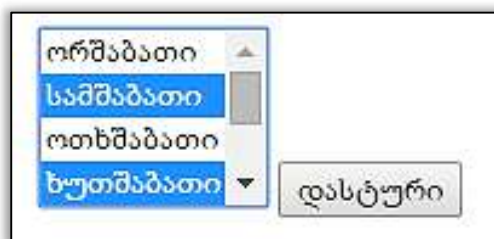
მაგალითით ვხედავთ, რომ დაჯგუფებული არის სიაში როგორც „მსუბუქი“, ისე „სამგზავრო“ ავტომობილები, ჩანს, თუ როგორ დაჯგუფდება `<optgroup>` საშუალებით და გამოიყენებს ისეთ ატრიბუტებს, როგორიც არის `disabled` და `label`, თუ რომელიმე ჯგუფის გათიშვა გვჭირდება, შეგვიძლია გამოვიყენოთ `<optgroup label="სამგზავრო" disabled>`, რის შემდეგადაც „სამგზავრო“-ში შემავალი სია არ იქნება აქტიური.

თუ გვინდა მოვნიშნოთ ერთზე მეტი პარამეტრი, მაშინ მოგვიწევს `multiple` ატრიბუტის გამოყენება, მაგალითისათვის ავიღოთ კვირის დღეები, რომლის მონიშვნაც შეგვიძლია (კლავიატურაზე `Ctrl` დაჭერისა და მაუსისის ერთობლივი კომბინაციით).

მაგალითი 1.53

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> multiple
</title>
</head>
<body>
<form action="saiti.ge" method="get" name="dgeebi">
<select name="კვირა" multiple>
  <option value="ორშ">ორშაბათი</option>
  <option value="სამ">სამშაბათი</option>
  <option value="ოთხ">ოთხშაბათი</option>
  <option value="ხუთ">ხუთშაბათი</option>
  <option value="პარ">პარასკევი</option>
  <option value="შაბ">შაბათი</option>
  <option value="კვ">კვირა</option>
</select>
<br>
<input type="submit" value="დასტური">
</form>
</body>
</html>
```

შედეგი



<button>

ლილაკის ტეგი, რომლის საშუალებითაც შესაძლებელია ლილაკების შექმნა და დაჭერა. მისი ატრიბუტებია: *autofocus*, *disabled*, *form*, *formaction*, *formenctype*, *formmethod*, *formnovalidate*, *formtarget*, *name*, *type*, *value* ასევე ამ ტეგის გამოყენება შეიძლება *form*-ის შიგნითაც, ტექსტის თუ სურათის დასმა, შესაძლებელია ისეთი ლილაკების შექმნა, როგორიც არის *submit* (დასტური), *reset* (ფორმის განულება), თვითონ შიგვე *button* ლილაკის შექმნა და მასში სამივე ელემენტის გამოყენება.

მაგალითი 1.54

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```



```

<title>button</title>
</head>
<body>
<form>
<button type="submit"> დასტური</button>
<p>
  <button type="reset"><b> ფორმის გასუფთავება </b> </button></p>
<p>
  <button type="button">
 </button>
</p>
</form>
</body>
</html>

```

შედეგი:



<keygen>

ახალი ტეგი, რომელიც ჩამატებულია HTML5 ში და წარმოდგენილია, როგორც ფორმის კონტროლერი, რომელიც გენერირდება როგორც ღია და დახურულ გასაღებად. ეს ერთგვარი დაცვითი მეთოდია შეტევებისაგან თავის ასარიდებლად, ამ ტეგის არსი არის ის, რომ არსებობს ორი გასაღები ღია (*public*) და პრივატული (*private*), როდესაც ფორმა შეივსება, ამის შემდეგ მომხარებელი დასტურის ღილაკს დააჭერს, ღია გასაღები გადაეცემა სერვერზე, შევქმნათ *textarea*, სადაც ჩაწერილი ტექსტი იქნება დაცული.

მაგალითი 1.55

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
  <title>keygen</title>
</head>

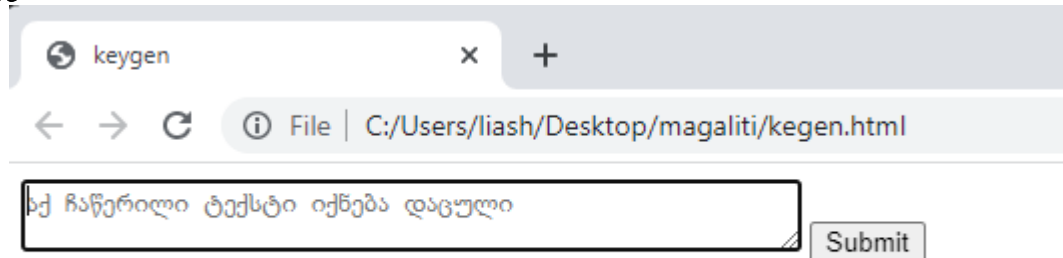
```

```

<body>
<form action="" " method="post" enctype="multipart/form-data">
<keygen name="key">
    <textarea cols="50" placeholder="აქ ჩაწერილი ტექსტი იქნება დაცული">
    </textarea>
<input type=submit value="Submit">
</form>
</body>
</html>

```

შედეგი



მისი ატრიბუტები არის *autofocus*, *challenge*, *disabled*, *form*, *keytype*, *name* რაც შეეხება *challenge* ატრიბუტს, რომელიც არ განგვიხილავს, ადასტურებს, რომ საწყისად წარმოდგენილია ღია გასაღები.

keytype ატრიბუტი არის დაცვითი ალგორითმი, რომელსაც სტანდარტულად ენიჭება *RSA*.

სინტაქსი

```
<keygen keytype="rsa ან dsa ან ec">
```

ატრიბუტი აღწერა

- | | |
|------------|--|
| <i>rsa</i> | ენიჭება სტანდარტულად <i>rsa</i> ალგორითმი რომელიც აძლევს მომხარებელს შესაძლებლობას, რომ აირჩიოს <i>rsa</i> გასაღების სიძლიერე. |
| <i>dsa</i> | <i>DSA</i> დაცვითი ალგორითმი მომხარებლებს აძლევს საშუალებას, აირჩიოს <i>DSA</i> გასაღების ზომა. |
| <i>ec</i> | <i>EC</i> დაცვითი ალგორითმი მომხარებლებს აძლევს საშუალებას, რომ აირჩიოს <i>EC</i> გასაღების სიძლიერე. |

ეს გასაღებები სხვადასხვა ბრაუზერში სხვადასხვაინარად წარმოდგება და რა თქმა უნდა თითოეული დაშიფვრის ზომა *bit* ებში გამოიხატება, მაგალითი, *Mozilla Firefox*, *Google Chrome* და *Opera*-ში წარმოდგება როგორც *2048 (High Grade)*, *1024 (Medium Grade)* ხოლო *Safari 5*-ში, როგორც *2048 (High Grade)*, *1024 (Medium Grade)*, *512 (Low Grade)* რაც შეეხება *Internet Explorer* საერთოდაც არ აქვს მისი მხარდაჭერა. მთლიანი სკრიპტი გამოიყურება შემდეგნაირად:

```

<keygen name="სახელი" challenge="სტრუქტონი" keytype="ტიპი" keyparams="pqg-params">

```

სკრიპტში მივაქციოთ ყურადღება **keytype="ტიპი"**, სადაც „ტიპის“ მაგივრად იწერება *rsa*, *dsa* ან *ec*.

keytype პარამეტრში უნდა მივუთითოთ, რომელი გასაღები უნდა მივუთითოთ, ამ ეს იქნება *RSA*, *DSA* თუ *EC* გასაღებები, სამივე შემთხვევაში აუცილებელია, რომ მივუთითოთ **name** და **challenge** ატრიბუტები, ხოლო **keyparms** ატრიბუტი მხოლოდ გამოიყენებს *DSA* და *EC* გასაღებებს და იგნორირებას უკეთებს *RSA*, ხოლო *PQG* არის სინონიმი **keyparam-**ის ამიტომაც იწერება შემდეგი სახით **keyparams="pqg-params"** ან **pqg="pqg-params"**.

RSA-გასაღებები არ გამოჩნდება **keyparam**-სი თუ მაინც იქნა წარმოდგენილი, მომხარებელს ეძლევა შესაძლებლობა, აირჩიოს, რა სიძლიერის გასაღები შეიძლება აირჩიოს, უფრო ხშირად იყენებენ მაღალი სიმძლავრის-*"high"* (2048 bit) დაშიფრის ალგორითმს, ვიდრე საშუალო-*"medium"* სიმძლავრის (1024 bit) *RSA* არის კრიპტოგრაფიის ასიმეტრიული ალგორითმი (*public* სიტყვა-გასაღები), რომელთა მნიშვნელოვან გამოყენებას პოვებს ელექტრონულ კომერციაში, განსაკუთრებით ინტერნეტში საიდუმლო მონაცემების გაცვლა-გამოცვლისათვის. ეს ალგორითმი შეიქმნა რონ რივესტის, ადი შამირის და ლენ ადლემანის მიერ 1977 წელს, საიდანაც წარმოდგა მისი სახელწოდება.

DSA-გასაღებები თუ არის მითითებული **keyparam**-ში, ეს ნიშნავს რომ გამოყენება *PQG* პარამეტრები, რომელიც თავის მხრივ დაფუძნებულია *BASE64* ენკოდირებაზე, რომლის საშუალებითც მომხარებლებს ეძლევათ შესაძლებლობა, აირჩიონ გასაღების ზომა.

EC (Elliptic curve)-გასაღებები, ელიფსური გასაღები, ეს არის კრიფტოგრაფიაში მიმართულება, რომლის განხილვას ახლა არ გავარჩევთ, საერთოდ კრიპტოგრაფია განიხილება, როგორც მათემატიკისა და კომპიუტერული მეცნიერებების განაყოფი, და მჭიდროდ დაკავშირებულია მეცნიერების ისეთ დარგებთან, როგორიცაა: ინფორმაციის თეორია, კომპიუტერული უსაფრთხოება და ინჟინერია, რაც შეეხება თვითონ ამ დაშიფრის მეთოდს, ძალიან რომ არ გადავიდეთ კრიპტოგრაფიაში, ვიტყვით, რომ მათემატიკიდან ამავე სახელწოდებით არის წამოსული, ხოლო, დაპროგრამებაში წარმოდგენს ერთ-ერთი მძლავრი დაშიფრის მეთოდს და მომხარებლის მხრიდან შესაძლებელია მისი გამოყენება.

<output>

საშუალებით ხდება ნებისმიერი კალკულაციის შედეგის გამოტანა, მისი ატრიბუტებია **for**, **name** და **form**, რაც შეეხება ბრაუზერებს ყველას აქვს მისი მხარდაჭერა გარდა *Internet Explorer*.

გავეცნოთ ახალ **for** ატრიბუტს, რომელიც ურთიერთქმედებს შედეგსა და კალკულაციის შორის, რომლის აუცილებელი მოთხოვნა არის, რომ უნდა ჰქონდეს ელემენტის ID.

სინტაქსი:

<output for="ელემენტის ID ">

მაგალითი 1.56

<!doctype html>

<html>

<head>

<meta charset="utf-8">

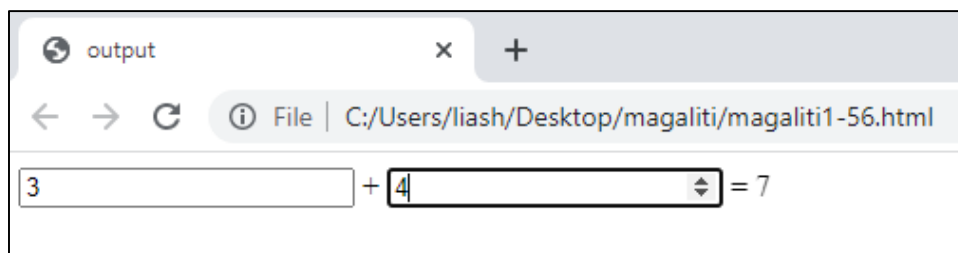
<title>output</title>

```

</head>
<body>
  <form onsubmit="return false" oninput="result.value = a.valueAsNumber
  + b.valueAsNumber">
    <input name="a" type="number" step="any"> +
    <input name="b" type="number" step="any"> =
    <output name="result" for="a b"> >
  </output>
</form>
</body>
</html>

```

შედეგი



რაც შეეხება სკრიპტში არსებულ ელემენტებს *onsubmit*, *oninput* და *step*, რომელთაც არ გავცნობივართ ორიოდ სიტყვით შევხვით.

Onsubmit-არის ივენტი ანუ მოვლენა, რომელიც სრულდება, და მუშაობს მაშინ, როცა ფორმა დასტურდება, ძირითადად მუშაობს javascripts ენასთან ერთად, სინტაქსი

```
<form onsubmit="სკრიპტი">
```

ან

```
<ელემენტი onsubmit="სკრიპტი">
```

რაც შეეხება *oninput* ატრიბუტს სრულდება მაშინ, როდესაც *<input>* შევიტანთ რაიმეს, ეს იქნება ტექსტი თუ ციფრი, ანალოგიურად ესეც მუშაობს javascripts ენასთან

მაგალითი 1.57

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<p>ჩაწერე რაიმე ტექსტი</p>
<input type="text" id="myInput" oninput="myFunction()">
<p id="რამე">
</p>
<script>
function myFunction()
{

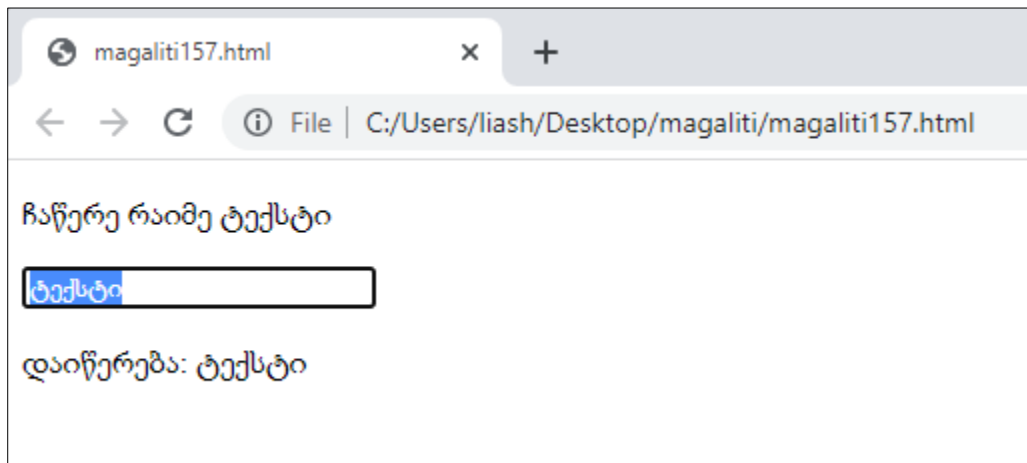
```

```

    var x = document.getElementById("myInput").value;
    document.getElementById("რამე").innerHTML = "დაიწერება: " + x;
  }
</script>
</body>
</html>

```

შედეგი



ხოლო ატრიბუტები, რომელიც მოცემული გვაქვს და არ განგვიხილავს მოცემულია ცხრილის სახით

step	ინტერვალის შეტანილ რიცხვებს შორის
min	მინიმალური სიდიდე
max	მაქსიმალური სიდიდე
pattern	უნდა შეიცავდეს რაღაც გარკვეულ სიმბოლოს

step ტეგს გამოყენებით შეგვეძლება ყოველივე რაღაც ზომის ერთეულად გადახტეს, მაგალითი, თუ **step="2"** მაშინ 2 ერთეულით გაიზრდება რიცხვი: 2, 4, 6 და ა. შ.

სინტაქსი

```
<input step="რიცხვი">
```

ხოლო რაც შეეხება **min** და **max** მასში შეიძლება გამოყენებული იქნას მინიმალური და მაქსიმალური სიდიდეები, მაგალითი, მინიმალურად ავიღოთ 1990 წელი, ხოლო, თუ მიუთითებთ 1989 ან 2016 წელს, არაფერს არ გამოიტანს, რაც შეეხება **pattern**-ს, მისი საშუალებით შეიძლება, რომ დასაშვები სიმბოლოების *რაოდენობის* ან დასაშვები *თვითონ სიმბოლოები* განისაზღვროს.

სინტაქსი:

```
<input pattern="regex">
```

მაგალითი 1.58

```

<!doctype html>
<html>
<head>

```

```

<meta charset="utf-8">
<title>მაგალითი </title>
</head>
<body>
<form action="dd"> <br>
<strong>
რიცხვების ზრდა </strong>
  <input type="number" name="points" step="2">
  <strong>
  <br>
მაქსიმალური </strong>
  <input type="date" name="თარიღი" max="2016-01-01"> <br>
  მინიმალური
  <input type="date" name="თარიღი" min="1990-01-01">
  <br>
<input type="submit">
</form>
</body>
</html>

```

შედეგი

სურათზე ვხედავთ, ველზე დაკლიკების მერე რომ მაქსიმალური ვადით თარიღი არის 2016 წლის იანვარი, ხოლო მინიმალური 1990 წლის 1 პირველი იანვარი

The image shows two side-by-side screenshots of a web form titled 'რიცხვების ზრდა' (Increase the number). Both screenshots show a 'მაქსიმალური' (Maximum) date field set to 'mm/dd/yyyy' and a 'მინიმალური' (Minimum) date field set to 'mm/dd/yyyy'. A 'Submit' button is visible below the date fields.

The left screenshot shows the 'max' date field set to 'January 2016'. Below the date field, a calendar for January 2016 is displayed, showing days from 27 to 31. The 'min' date field is empty.

The right screenshot shows the 'min' date field set to 'July 1915'. Below the date field, a list of years from 1990 to 1994 is displayed. The 'max' date field is empty.

რაც შეეხება *regex* ითარგმნება, როგორც *რეგულარული გამოსახულება* (*Regular expression*) და მუშაობს ისეთ ტიპებთან, როგორიც არის *text*, *date*, *search*, *url*, *tel*, *email*, და *password*.

მოვიყვანოთ ორიოდე მაგალითი, სადაც შესაძლებელი იქნება საიტის მისამართის მორგება, ხოლო მეორე მაგალითში ჩავწეროთ სიმბოლოების რაოდენობა.

მაგალითი 1.59

```

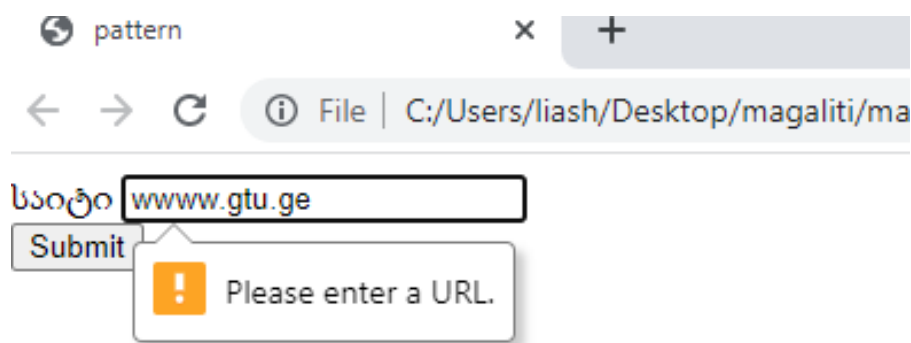
<!doctype html>
<html>
<head>

```

```

<meta charset="utf-8">
<title>pattern</title>
</head>
<body>
<form action="pattern">
საიტი
<input type="url" name="საიტი" pattern="https?:/\.+" title="შეიცავდეს
http://"><br>
<input type="submit">
</form>
</body>
</html>
შედეგი

```



რაც შეეხება მეორე მაგალითს, ჩავანაცვლოთ მეორე სკრიპტში `<input>`, სადაც შეგვიძლია როგორც დიდი ლათინური ასოებით ასევე პატარა ასოებით შევიყვანოთ 5 სიმბოლო, თუ აღმოჩნდა 5 სიმბოლოზე მეტი შეტყობინებას დაგვიწერს ანალოგიური ტექსტით „შეიყვანეთ ხუთი სიმბოლო“.

```

<input type="text" name="simbolo" pattern="[A-Za-z]{5}" title="შეიყვანეთ
ხუთი სიმბოლო">

```

ცხრილში მოცემულია არის რეგულარული გამოსახულების ყველა ანუ *regex* სიმბოლოდა მნიშვნელობა, მათი მეშვეობით შესაძლებელია მივუთითოთ კონკრეტული სიმბოლოები და ციფრები.

სიმბოლოები	აღწერა
<code>\d [0-9]</code>	ერთი ციფრი 0 -დან 9-მდე
<code>\D [^0-9]</code>	ნებისმიერი სიმბოლო, ციფრების გარდა
<code>\s</code>	Space, tab
<code>[A-Z]</code>	დიდი ლათინური ასოები
<code>[A-Za-z]</code>	ნებისმიერი სიმბოლო, რეგისტრში
<code>[0-9]{6}</code>	6 ციფრი
<code>[A-Za-z]{5,}</code>	არაუმცირეს 5 სიმბოლოსი
<code>[0-9]{, 4}</code>	არაუმეტეს 4 სიმბოლოსი
<code>[0-9]{2, 7}</code>	ორიდან შვიდამდე ციფრები
<code>^[a-zA-Z]+\$</code>	ნებისმიერი ლათინური სიმბოლო

<code>^[0-9]+\$</code>	ნებისმიერი სიმბოლო
<code>[0-9]{6}</code>	პოსტის ინდექსი
<code>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}</code>	IP-მისამართი
<code>\0</code>	ნულოვანი სიმბოლოს პოვნა null
<code>\n</code>	ახალ ხაზზე გადასვლა
<code>\f</code>	გვერდის გადაფურცვლა
<code>\r</code>	სტრიქონის დასაწყისში გადასვლა
<code>\t</code>	ჰორიზონტალური ტაბულირება
<code>\v</code>	ვერტიკალური ტაბულირება
<code>\w</code>	პოულობს სიმბოლოებს
<code>\W</code>	ტოვებს სიმბოლოებს
<code>\d</code>	ციფრებს აღიქვამს
<code>\D</code>	არ აღიქვამს
<code>\s</code>	Space, tab გამოყენება
<code>\S</code>	არ იყენებს space, tab
<code>\b</code>	პოულობს საწყის და საბოლოო ასოებს
<code>\B</code>	ტოვებს საწყის და საბოლოო მასოებს
<code>\xxx</code>	იპოვის 8 ციფრს
<code>\xdd</code>	გამოიყენებს თექვსმეტობით სიმბოლოს dd
<code>\uxxxx</code>	უნიკოდის გამოყენება თექვსმეტობითში

ფაილების მონიშვნა

დავუშვათ, გვინდა რამდენიმე ფაილის მონიშვნა *Word* დოკუმენტის ან სურათის ატვირთვა, რაც ხშირად გვხდება ინტერნეტ სივრცესი, როგორ არის ეს შესაძლებელი, გავეცნოთ ორი `<input>` ტეგის ორ ატრიბუტს, თავის ელემენტებით, როგორიც არის *file*, *accept* და *dirname*.

File-ტიპი განსაზღვრავს ტიპს, რა მოხლენა უნდა მოხდეს.

accept- ატრიბუტი განსაზღვრავს სერვერზე რა ტიპის ინფორმაცია იგზავნება, ეს იქნება, სურათის, ვიდეოს თუ აუდიო ფაილი, ასეთ ტიპებს *MIME* ტიპებს უწოდებენ.

ფაილის გაფართოება	<i>gif, .jpg, .png, .doc</i>
<i>audio/*</i>	აუდიო ფაილები
<i>video/*</i>	ვიდეო ფაილები
<i>image/*</i>	სურათის ფაილები
<i>media_type</i>	ყველა სტანდარტული მედია ფაილები

მაგალითი 1.59

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>ფაილის ატვირთვა</title>

```

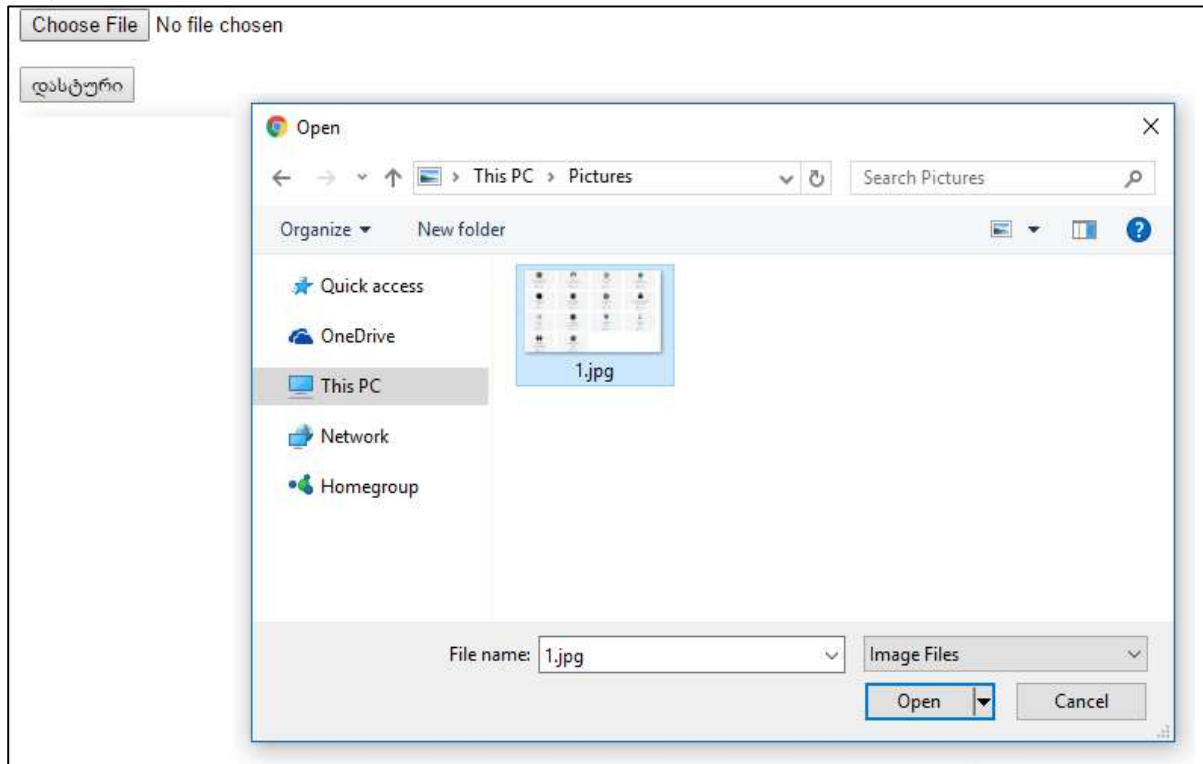


```

</head>
<body>
<form action="" method="post"
name="ფაილი ატვირთვა" enctype="multipart/form-data">
<input type="file" name="ატვირთვა" accept="image/*">
<p><input type="submit" value="დასტური" ><p>
</form>
</body>
</html>

```

შედეგი



dirname-ატრიბუტს გამოაქვს ტექსტის მიმართულება და დამოკიდებულია მიმართულებაზე, სტანდარტულად მარცხიდან-მარჯვნიდან იწერება *Ltr*, ხოლო ასევე სტანდარტულად რელავანტურია *input* ისეთ ტიპებთან, როგორც *text* და *search* და ასევე *<textarea>*

```

<input type="text" name="saxeli" dirname="saxeli.dir">
<input type="submit">

```

MIME-(*Multipurpose Internet Mail Extensions*) ითარგმნება, როგორც მულტი დანიშნულების ინტერნეტფოსტის გაფართოებები, რომელიც არის სტანდარტი, რომ დაადგინოს, თუ ფაილი რა ტიპის შემცველია, ეს შეიძლება იყოს ვიდეო, აუდიო, ტექსტური და ა.შ, ახლაც შეიძლება თქვენს ეკრანზე იყოს *Word* ფაილი, რომლის გაფართოება *.docx*, სურათი, რომლის ფორმატი არის: *jpg, png* და ა.შ

MIME-ფაილი *HTML*-ში განისაზღვრება როგორც *text/html*

MIME ტიპი კონტენტი თავისი გაფართოებით

აპლიკაცია	MIME Type	გაფართოება
Corel Envoy	application/envoy	evy
fractal image file	application/fractals	fif
Windows print spool file	application/futuresplash	spl
HTML application	application/hta	hta
Atari ST Program	application/internet-property-stream	acx
BinHex encoded file	application/mac-binhex40	hqx
Word document	application/msword	doc
Word document template	application/msword	dot
	application/octet-stream	*
binary disk image	application/octet-stream	bin
Java class file	application/octet-stream	class
Disk Masher image	application/octet-stream	dms
executable file	application/octet-stream	exe
LHARC compressed archive	application/octet-stream	lha
LZH compressed file	application/octet-stream	lzh
CALS raster image	application/oda	oda
ActiveX script	application/olescript	axs
Acrobat file	application/pdf	pdf
Outlook profile file	application/pics-rules	prf
certificate request file	application/pkcs10	p10
certificate revocation list file	application/pkix-crl	crl
Adobe Illustrator file	application/postscript	ai
postscript file	application/postscript	eps
postscript file	application/postscript	ps
rich text format file	application/rtf	rtf
set payment initiation	application/set-payment-initiation	setpay

<i>აპლიკაცია</i>	<i>MIME Type</i>	<i>გაფართოება</i>
<i>set registration initiation</i>	<i>application/set-registration-initiation</i>	<i>setreg</i>
<i>Excel Add-in file</i>	<i>application/vnd.ms-excel</i>	<i>xla</i>
<i>Excel chart</i>	<i>application/vnd.ms-excel</i>	<i>xlc</i>
<i>Excel macro</i>	<i>application/vnd.ms-excel</i>	<i>xlm</i>
<i>Excel spreadsheet</i>	<i>application/vnd.ms-excel</i>	<i>xls</i>
<i>Excel template</i>	<i>application/vnd.ms-excel</i>	<i>xlt</i>
<i>Excel worspace</i>	<i>application/vnd.ms-excel</i>	<i>xlw</i>
<i>Outlook mail message</i>	<i>application/vnd.ms-outlook</i>	<i>msg</i>
<i>serialized certificate store file</i>	<i>application/vnd.ms-pkicertstore</i>	<i>sst</i>
<i>Windows catalog file</i>	<i>application/vnd.ms-pkiseccat</i>	<i>cat</i>
<i>stereolithography file</i>	<i>application/vnd.ms-pkistl</i>	<i>stl</i>
<i>PowerPoint template</i>	<i>application/vnd.ms-powerpoint</i>	<i>pot</i>
<i>PowerPoint slide show</i>	<i>application/vnd.ms-powerpoint</i>	<i>pps</i>
<i>PowerPoint presentation</i>	<i>application/vnd.ms-powerpoint</i>	<i>ppt</i>
<i>Microsoft Project file</i>	<i>application/vnd.ms-project</i>	<i>mpp</i>
<i>WordPerfect macro</i>	<i>application/vnd.ms-works</i>	<i>wcm</i>
<i>Microsoft Works database</i>	<i>application/vnd.ms-works</i>	<i>wdb</i>
<i>Microsoft Works spreadsheet</i>	<i>application/vnd.ms-works</i>	<i>wks</i>
<i>Microsoft Works word processor document</i>	<i>application/vnd.ms-works</i>	<i>wps</i>
<i>Windows help file</i>	<i>application/winhelp</i>	<i>hlp</i>
<i>binary CPIO archive</i>	<i>application/x-bcpio</i>	<i>bcpio</i>
<i>computable document format file</i>	<i>application/x-cdf</i>	<i>cdf</i>
<i>Unix compressed file</i>	<i>application/x-compress</i>	<i>z</i>
<i>gzipped tar file</i>	<i>application/x-compressed</i>	<i>tgz</i>
<i>Unix CPIO archive</i>	<i>application/x-cpio</i>	<i>cpio</i>
<i>Photoshop custom shapes file</i>	<i>application/x-csh</i>	<i>csh</i>

<i>აპლიკაცია</i>	<i>MIME Type</i>	<i>გაფართოება</i>
<i>Kodak RAW image file</i>	<i>application/x-director</i>	<i>dcr</i>
<i>Adobe Director movie</i>	<i>application/x-director</i>	<i>dir</i>
<i>Macromedia Director movie</i>	<i>application/x-director</i>	<i>dxr</i>
<i>device independent format file</i>	<i>application/x-dvi</i>	<i>dvi</i>
<i>Gnu tar archive</i>	<i>application/x-gtar</i>	<i>gtar</i>
<i>Gnu zipped archive</i>	<i>application/x-gzip</i>	<i>gz</i>
<i>hierarchical data format file</i>	<i>application/x-hdf</i>	<i>hdf</i>
<i>internet settings file</i>	<i>application/x-internet-signup</i>	<i>ins</i>
<i>IIS internet service provider settings</i>	<i>application/x-internet-signup</i>	<i>isp</i>
<i>ARC+ architectural file</i>	<i>application/x-iphone</i>	<i>iii</i>
<i>JavaScript file</i>	<i>application/x-javascript</i>	<i>js</i>
<i>LaTeX document</i>	<i>application/x-latex</i>	<i>latex</i>
<i>Microsoft Access database</i>	<i>application/x-msaccess</i>	<i>mdb</i>
<i>Windows CardSpace file</i>	<i>application/x-mscardfile</i>	<i>crd</i>
<i>CrazyTalk clip file</i>	<i>application/x-msclip</i>	<i>clp</i>
<i>dynamic link library</i>	<i>application/x-msdownload</i>	<i>dll</i>
<i>Microsoft media viewer file</i>	<i>application/x-msmediaview</i>	<i>m13</i>
<i>Steuer2001 file</i>	<i>application/x-msmediaview</i>	<i>m14</i>
<i>multimedia viewer book source file</i>	<i>application/x-msmediaview</i>	<i>mvb</i>
<i>Windows meta file</i>	<i>application/x-msmetafile</i>	<i>wmf</i>
<i>Microsoft Money file</i>	<i>application/x-msmoney</i>	<i>mny</i>
<i>Microsoft Publisher file</i>	<i>application/x-mspublisher</i>	<i>pub</i>
<i>Turbo Tax tax schedule list</i>	<i>application/x-msschedule</i>	<i>scd</i>
<i>FTR media file</i>	<i>application/x-msterminal</i>	<i>trm</i>
<i>Microsoft Write file</i>	<i>application/x-mswrite</i>	<i>wri</i>
<i>computable document format file</i>	<i>application/x-netcdf</i>	<i>cdf</i>
<i>Mastercam numerical control file</i>	<i>application/x-netcdf</i>	<i>nc</i>

<i>აპლიკაცია</i>	<i>MIME Type</i>	<i>გაფართოება</i>
<i>MSX computers archive format</i>	<i>application/x-perfmon</i>	<i>pma</i>
<i>performance monitor counter file</i>	<i>application/x-perfmon</i>	<i>pmc</i>
<i>process monitor log file</i>	<i>application/x-perfmon</i>	<i>pml</i>
<i>Avid persistant media record file</i>	<i>application/x-perfmon</i>	<i>pmr</i>
<i>Pegasus Mail draft stored message</i>	<i>application/x-perfmon</i>	<i>pmw</i>
<i>personal information exchange file</i>	<i>application/x-pkcs12</i>	<i>p12</i>
<i>PKCS #12 certificate file</i>	<i>application/x-pkcs12</i>	<i>pfx</i>
<i>PKCS #7 certificate file</i>	<i>application/x-pkcs7-certificates</i>	<i>p7b</i>
<i>software publisher certificate file</i>	<i>application/x-pkcs7-certificates</i>	<i>spc</i>
<i>certificate request response file</i>	<i>application/x-pkcs7-certreqresp</i>	<i>p7r</i>
<i>PKCS #7 certificate file</i>	<i>application/x-pkcs7-mime</i>	<i>p7c</i>
<i>digitally encrypted message</i>	<i>application/x-pkcs7-mime</i>	<i>p7m</i>
<i>digitally signed email message</i>	<i>application/x-pkcs7-signature</i>	<i>p7s</i>
<i>Bash shell script</i>	<i>application/x-sh</i>	<i>sh</i>
<i>Unix shar archive</i>	<i>application/x-shar</i>	<i>shar</i>
<i>Flash file</i>	<i>application/x-shockwave-flash</i>	<i>swf</i>
<i>Stuffit archive file</i>	<i>application/x-stuffit</i>	<i>sit</i>
<i>system 5 release 4 CPIO file</i>	<i>application/x-sv4cpio</i>	<i>sv4cpio</i>
<i>system 5 release 4 CPIO checksum data</i>	<i>application/x-sv4crc</i>	<i>sv4crc</i>
<i>consolidated Unix file archive</i>	<i>application/x-tar</i>	<i>tar</i>
<i>Tcl script</i>	<i>application/x-tcl</i>	<i>tcl</i>
<i>LaTeX source document</i>	<i>application/x-tex</i>	<i>tex</i>
<i>LaTeX info document</i>	<i>application/x-texinfo</i>	<i>texi</i>
<i>LaTeX info document</i>	<i>application/x-texinfo</i>	<i>texinfo</i>

აპლიკაცია	MIME Type	გაფართოება
<i>unformatted manual page</i>	<i>application/x-troff</i>	<i>roff</i>
<i>Turing source code file</i>	<i>application/x-troff</i>	<i>t</i>
<i>TomeRaider 2 ebook file</i>	<i>application/x-troff</i>	<i>tr</i>
<i>Unix manual</i>	<i>application/x-troff-man</i>	<i>man</i>
<i>readme text file</i>	<i>application/x-troff-me</i>	<i>me</i>
<i>3ds Max script file</i>	<i>application/x-troff-ms</i>	<i>ms</i>
<i>uniform standard tape archive format file</i>	<i>application/x-ustar</i>	<i>ustar</i>
<i>source code</i>	<i>application/x-wais-source</i>	<i>src</i>
<i>internet security certificate</i>	<i>application/x-x509-ca-cert</i>	<i>cer</i>
<i>security certificate</i>	<i>application/x-x509-ca-cert</i>	<i>crt</i>
<i>DER certificate file</i>	<i>application/x-x509-ca-cert</i>	<i>der</i>
<i>public key security object</i>	<i>application/ynd.ms-pkipko</i>	<i>pko</i>
<i>zipped file</i>	<i>application/zip</i>	<i>zip</i>

აუდიო ფაილები

აპლიკაცია	MIME Type	გაფართოება
<i>audio file</i>	<i>audio/basic</i>	<i>au</i>
<i>sound file</i>	<i>audio/basic</i>	<i>snd</i>
<i>midi file</i>	<i>audio/mid</i>	<i>mid</i>
<i>media processing server studio</i>	<i>audio/mid</i>	<i>rmi</i>
<i>MP3 file</i>	<i>audio/mpeg</i>	<i>mp3</i>
<i>audio interchange file format</i>	<i>audio/x-aiff</i>	<i>aif</i>
<i>compressed audio interchange file</i>	<i>audio/x-aiff</i>	<i>aifc</i>
<i>audio interchange file format</i>	<i>audio/x-aiff</i>	<i>aiff</i>
<i>media playlist file</i>	<i>audio/x-mpegurl</i>	<i>m3u</i>
<i>Real Audio file</i>	<i>audio/x-pn-realaudio</i>	<i>ra</i>
<i>Real Audio metadata file</i>	<i>audio/x-pn-realaudio</i>	<i>ram</i>

<i>აპლიკაცია</i>	<i>MIME Type</i>	<i>გაფართოება</i>
<i>WAVE audio file</i>	<i>audio/x-wav</i>	<i>wav</i>

სურათის ფაილები

<i>აპლიკაცია</i>	<i>MIME Type</i>	<i>გაფართოება</i>
<i>Bitmap</i>	<i>image/bmp</i>	<i>bmp</i>
<i>compiled source code</i>	<i>image/cis-cod</i>	<i>cod</i>
<i>graphic interchange format</i>	<i>image/gif</i>	<i>gif</i>
<i>image file</i>	<i>image/ief</i>	<i>ief</i>
<i>JPEG image</i>	<i>image/jpeg</i>	<i>jpe</i>
<i>JPEG image</i>	<i>image/jpeg</i>	<i>jpeg</i>
<i>JPEG image</i>	<i>image/jpeg</i>	<i>jpg</i>
<i>JPEG file interchange format</i>	<i>image/pipepeg</i>	<i>jfif</i>
<i>scalable vector graphic</i>	<i>image/svg+xml</i>	<i>svg</i>
<i>TIF image</i>	<i>image/tiff</i>	<i>tif</i>
<i>TIF image</i>	<i>image/tiff</i>	<i>tiff</i>
<i>Sun raster graphic</i>	<i>image/x-cmu-raster</i>	<i>ras</i>
<i>Corel metafile exchange image file</i>	<i>image/x-cmx</i>	<i>cmx</i>
<i>icon</i>	<i>image/x-icon</i>	<i>ico</i>
<i>portable any map image</i>	<i>image/x-portable-anymap</i>	<i>pnm</i>
<i>portable bitmap image</i>	<i>image/x-portable-bitmap</i>	<i>pbm</i>
<i>portable graymap image</i>	<i>image/x-portable-graymap</i>	<i>pgm</i>
<i>portable pixmap image</i>	<i>image/x-portable-pixmap</i>	<i>ppm</i>
<i>RGB bitmap</i>	<i>image/x-rgb</i>	<i>rgb</i>
<i>X11 bitmap</i>	<i>image/x-xbitmap</i>	<i>xbm</i>
<i>X11 pixmap</i>	<i>image/x-xpixmap</i>	<i>xpm</i>
<i>X-Windows dump image</i>	<i>image/x-xwindowdump</i>	<i>xwd</i>

Mail ფაილები

აპლიკაცია	MIME Type	გაფართოება
MHTML web archive	message/rfc822	mht
MIME HTML file	message/rfc822	mhtml
Windows Live Mail newsgroup file	message/rfc822	nws

ტექსტური ფაილები

აპლიკაცია	MIME Type	გაფართოება
Cascading Style Sheet	text/css	css
H.323 internet telephony file	text/h323	323
HTML file	text/html	htm
HTML file	text/html	html
Exchange streaming media file	text/html	stm
NetMeeting user location service file	text/iuls	uls
BASIC source code file	text/plain	bas
C/C++ source code file	text/plain	c
C/C++/Objective C header file	text/plain	h
text file	text/plain	txt
rich text file	text/richtext	rtx
Scitext continuous tone file	text/scriptlet	sct
tab separated values file	text/tab-separated-values	tsv
hypertext template file	text/webviewhtml	htt
HTML component file	text/x-component	htc
TeX font encoding file	text/x-setext	etx
vCard file	text/x-vcard	vcf

ვიდეო ფაილი

აპლიკაცია	MIME Type	გაფართოება
MPEG-2 audio file	video/mpeg	mp2
MPEG-2 audio file	video/mpeg	mpa
MPEG movie file	video/mpeg	mpe
MPEG movie file	video/mpeg	mpeg
MPEG movie file	video/mpeg	mpg
MPEG-2 video stream	video/mpeg	mpv2
Apple QuickTime movie	video/quicktime	mov
Apple QuickTime movie	video/quicktime	qt
Logos Library system file	video/x-la-asf	lsf
streaming media shortcut	video/x-la-asf	lsx
advanced systems format file	video/x-ms-asf	asf
ActionScript remote document	video/x-ms-asf	asr
Microsoft ASF redirector file	video/x-ms-asf	asx
audio video interleave file	video/x-msvideo	avi
Apple QuickTime movie	video/x-sgi-movie	movie

ვირტუალური ფაილები

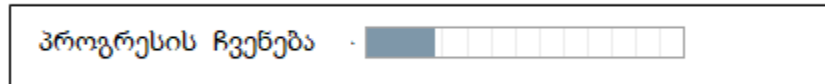
აპლიკაცია	MIME Type	გაფართოება
Flare decompiled actionscript file	x-world/x-vrml	flr
VRML file	x-world/x-vrml	vrml
VRML world	x-world/x-vrml	wrl
compressed VRML world	x-world/x-vrml	wrz
3ds max XML animation file	x-world/x-vrml	xaf
Reality Lab 3D image file	x-world/x-vrml	xof

<progress>

ამ ტეგის საშუალებით შესაძლებელია ამოცანის პროგრესის განსაზღვრა პროცენტულად ან სხვა სიდიდით, რომლის დამხმარე ატრიბუტებია *max* და *value*, ასევე იყენებს გლობალურ ატრიბუტებს, და ურთიერთქმედებს *javascript*-თან:

```
<progress value="30" max="100">30 %</progress>
```

შედეგი



<meter>

HTML5 მოიცავს ისეთ ახალ ელემენტებს, როგორებიცაა *time*, *mark* და მათ შორის არის *<meter>* ტეგიც, მისი გამოყენება შეგვიძლია როგორც რაღაც სკალარული სიდიდის საჩვენებლად, მისი მონათესავე არის *<progress>* ტეგი, მაგრამ სხვაობის აღსაქმელად *<meter>* ტეგი წარმოვიდგინოთ როგორც სტატიკური, ხოლო *<progress>* ტეგი როგორც დინამიკური ტეგი, ხოლო რაც შეეხება მის ატრიბუტებს, აქვს ექვსი ატრიბუტი *value*, *min*, *max*, *high*, *low* და *optimum*, განვიხილოთ *low*, *high* და *optimum* ატრიბუტები.

- ✓ **low**-ატრიბუტი იგივე არის, რაც *min*, განისაზღვრება როგორც მეტი, ვიდრე *min*, ხოლო ნაკლები, როგორც *high* და *max* ატრიბუტის სიდიდეზე.
- ✓ **high**-ატრიბუტი ნაკლებია *max* ატრიბუტზე და მეტია, ვიდრე *min* და *low* ატრიბუტის სიდიდეზე.
- ✓ **optimum**- ატრიბუტის არის მისაღების სიდიდე, ანუ მაქსიმუმსა და მინიმუმს შორის, მაგრამ სხვადასხვა ატრიბუტის გამოყენებით სხვადასხვა ფერი წარმოჩნდება, ავიღოთ მაგალითი, სადაც ხდება სამი მოსწავლის გამოცდების შედეგების ნახვა, რომლის სკრიპტიც გამოიყურება:

მაგალითი 1.60

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>meter</title>
</head>
<body>
<p>მარიამი: <meter min="0" low="40" high="90" max="100" value="95">
</meter></p>
<p> თაკო:
  <meter min="0" low="40" high="90" max="100" value="55">
  </meter> </p>
<p> მანანა
  <meter max="100" low="25" high="75" optimum="76" value="5">
  </meter>
```

```

</p>
</body>
</html>

```

შედეგი

მარიამი: 

თაკო: 

მანანა: 

- ✓ თუ $Low < optimum < high$
 $value$ არის $<Low > high$ გამოჩნდება ყვითელი საწინააღმდეგო შემთხვევაში მწვანედ.
- ✓ თუ $Low < high < optimum$:
 $value$ არის $<Low$, გამოჩნდება წითელი
 $value$ არის $<high$, გამოჩნდება ყვითელი საწინააღმდეგო შემთხვევაში მწვანედ.
- ✓ თუ $optimum < Low < high$:
 $value$ არის $> high$, გამოჩნდება წითლად.
 $value$ არის $> Low$, გამოჩნდება ყვითლად საწინააღმდეგო შემთხვევაში მწვანედ.

DISABLED და READ-ONLY

დავუბრუნდეთ ამ ორ ელემენტს, რომელიც უკვე განვიხილეთ ზემოთ და ყველა ელემენტთან მიმართებით წარმოვადგინოთ, თუ რომელ ტეგთან რომელი ელემენტი მუშაობს.

ცხრილში ნაჩვენებია, რომ კონრეტულ ელემენტს აქვს თუ არა მხარდაჭერა, ხოლო მხარდაჭერა აღვნიშნოთ-✓, ხოლო როცა არ აქვს – სიმბოლოთი.

ელემენტები	READONLY	DISABLED
<code><textarea></code>	✓	✓
<code><input type="text"></code>	✓	✓
<code><input type="checkbox"></code>	-	✓
<code><input type="radio"></code>	-	✓
<code><input type="submit"></code>	-	✓
<code><input type="reset"></code>	-	✓
<code><input type="button"></code>	-	✓
<code><input type="color"></code>	-	✓
<code><input type="date"></code>	✓	✓
<code><input type="datetime"></code>	✓	✓

<code><input type="datetime-local"></code>	✓	✓
<code><input type="email"></code>	✓	✓
<code><input type="file"></code>	-	✓
<code><input type="month"></code>	✓	✓
<code><input type="number"></code>	✓	✓
<code><input type="range"></code>	-	✓
<code><input type="search"></code>	✓	✓
<code><input type="tel"></code>	✓	✓
<code><input type="url"></code>	✓	✓
<code><input type="week"></code>	✓	✓
<code><input type="time"></code>	✓	✓
<code><select></code>	-	✓
<code><option></code>	-	✓
<code><optgroup></code>	-	✓
<code><keygen></code>	-	✓
<code><fieldset></code>	-	✓
<code><button></code>	-	✓

HTML 5 მულტიმედია

თანამედროვე კომპიუტერულ ტექნოლოგიურ ხანაში მულტიმედიის სხვადასხვა ფორმატი არსებობს, საერთოდ მულტიმედია არის ინფორმაციის გადაცემა, იქნება ეს ვიდეო, აუდიო, სურათი თუ ანიმაცია, ყველა ფორმატი, რომელიც გარკვეული ინფორმაციის მატერიალად არის, მულტიმედია.

WEB ტექნოლოგიებში იგი გვხვდება სხვადასხვა ტიპით და ფორმატით, თავდაპირველად ინტერნეტში ბრაუზერებს ჰქონდა ინფორმაციის მხარდაჭერა მხოლოდ ტექსტურ რეჟიმში, შემდეგ მოხდა დამატება სურათის, ხმის და ა.შ.

მულტიმედიის ფორმატში (ვიდეო და აუდიო) შედის ისეთი ფორმატები, როგორებიცაა: **სურათის ფორმატი: jpg, png და gif**

ვიდეო ფორმატები: .swf, .wav, .mp3, .mp4, .mpg, .wmv, და .avi.

HTML 5 სტანდარტებში ახალი ფორმატები არის დამატებული MP4, WebM, და Ogg, ამ სამ ფორმატის მხარდაჭერა ყველა თანამედროვე ბრაუზერს აქვს მაგალითი MP4 ხშირად გვხვდება ინტერნეტში, სხვადასხვა ვიდეო საიტებს, როგორიც არის youtube.

☞ კონტეინერი და კოდაკები

კოდაკები არის დაშიფრვის/გაშიფრვის სისტემა, რომელიც თარგმნის ყველაფერს 1 და 0 ციფრებად ანუ ბინარულად, გარდაქმნის ყველაფერს, რასაც უყურებთ და ვუსმენთ.

☞ *MP4* ფორმატი აბრევიატურა არის *MPEG-4 Part 14*, ასევე მოიხსენიებენ როგორც *MPEG-4 AVC*, ხოლო თავის მხრივ კიდევ იშიფრება როგორც *Advanced Video Coding* როგორც სახელიდან ვხდებით, ეს ის ფორმატია, რომელის საშუალებით ვიდეო ფაილების დეკოდირებას ახდენს და პირველად გვხდება 1998 წლიდან, ხოლო *MPEG* იშიფრება, როგორც *Motion Pictures Expert Group* (მოდრავი სურათების ექსპერტთა ჯგუფი). *MP4* ფაილი არის ყველაზე მისაღები ფორმატი ინტერტენ სივრცეში, რადგან ამ ფორმატის საშუალებით შესაძლებელია ისე მოახდინოს ფაილის ზომის შემცირება, რომ არ დაკარგოს ხარისხი, ამის გარდა, მოიცავს *H.264* ვიდეო კოდაკსა და *AAC* აუდიო კოდაკების იმ ფორმატსაც რომელიც აქვს ყველა ბრაუზერს, მხარდაჭერა, გარდა *Internet Explorer* არ აქვს.

☞ *WebM* ვიდეო ფორმატი და ცნობილია, როგორც *Matroska* ანუ *Matroska Multimedia Container*, რომელიც თავის მხრივ მოიცავს ვიდეო (*.MKV*), აუდიო (*.MKA*), სუბტიტრებს (*.MKS*) და სტერეოსკოპურ/3D ვიდეო (*.MK3D*) ფაილებს, მაგალითი ყველა ბრაუზერის გარდა *Gecko (Firefox)*-ში წარმოჩნდება *MIME* ტიპებში, როგორც *video/webm* ვიდეო კონტეინერი და აუდიო კონტეინერი - *audio/web*, ხოლო ყოველთვის იყენებს *VP8* ან *VP9* ვიდეო კოდაკს და *Vorbis* ან *Opus* აუდიო კოდაკს.

☞ *Ogg* კონტეინერი, რომელიც შექმნილი არის *Theora* ვიდეო კოდაკის და *Vorbis* აუდიო კოდაკის საფუძველზე, ასევე ამ ფორმატის პოპულარიზებაც პოვა, რადგან შესაძლებელია ფაილის ზომის შემცირება ხარისხის დაუკარგავად.

<video>

ვიდეო ტეგი *HTML 5*-ში გვაძლევს შესაძლებლობას ვუყუროთ ვიდეო ფაილებს, რა თქმა უნდა, ამის შესაძლებლობა ადრეც გვქონდა, მაგრამ ადრე ხდება *Adobe Flash* პლაგინის გამოყენება და ჩანერგვა/ჩაყენება ბრაუზერში, ხოლო *HTML 5*-ში შემუშავდა ტეგი, რომლის მხარდაჭერა თანამედროვე ყველა ბრაუზერისათვის გასაგებია და არ საჭიროებს გარკვეული პლაგინის დაყენებას, თუ ამას კონკრეტული საიტი არ მოითხოვს, ხოლო თვითონ <video> ელემენტი შემუშავდა 2007 წლის თებერვალში *Opera Software*-მიერ.

<video> აქვს თავის ატრიბუტები, როგორიცაა: *width*, *height* და *src*, მათი საშუალებით იგებს ვიდეოს სიმაღლეს და სიგანეს, თუ მითითებული არ აქვს ბრაუზერისათვის გაუგებარია, რა პარამეტრებში გამოაჩინოს ვიდეო, ამის გარდა, უნდა იცოდეს, სად არის ვიდეო და რა მისამართიან მოხდეს ჩატვირთვა ანუ რომელი დირექტორში იმყოფება, რაზეც *src* არის პასუხისმგებელი

სინტაქსი:

```
<video width="300" height="150" src="ვიდეო.mov">  
აქ ჩაწერილი ტექსტი გამოჩნდება ვიდეოს ქვეშ  
</video>
```

ტეგებს შუაში მოთავსებული ტექსტი, რა თქმა უნდა, გამოჩნდება ეკრანზე, მაგრამ `<video>` პასუხისმგებელია თვითონ ვიდეო ფაილებზე, ბრაუზერმა ვერ მოახდინა მისი დეკოდირება მაგისათვის შექმნეს `<source>`, რომელიც პასუხისმგებელია ვიდეო და აუდიო ფაილებზე და ამავდროს პასუხისმგებელია მის ფაილების ფორმატზე და მუშაობს ისე, რომ მისთვის ნაცნობ პირველივე ფორმატს წაიკითხავს, კიდევ ერთი ტეგი, რომელიც უნდა აღინიშნოს არის `<track>` და განსაზღვრავს ტრეკებს, ამას ცოტა მოგვიანებით შევეხებით, რადგან აქვს თავის ატრიბუტები და მათი განხილვა მოგვიწევს.

მაგალითში ნაჩვენებია ვიდეო ტეგის სტანდარტული სკრიპტი, როგორ იწერება, ვიდეო მისამართი არის <http://saiti.ge/video.mp4>, ხოლო ფორმატი, როგორც ვხედავთ, არის `mp4`, სკრიპტში არსებულ პირველ ორ სტრიქონს გამოტოვებს, რადგან არცერთ ფორმატს არ შეესაბამება, სანამ მაგალითზე გადავიდოდეთ, განვიხილოთ `<source>` ტეგი.

`<source>`

სოფრისით შესაძლებელია სხვადასხვა დირექტორის ფაილის ფორმატის მითითება, როგორც ზემოთ აღვნიშნე ბრაუზერს აქვს შესაძლებლობა რომ ამოირჩიოს და წაიკითხოს რეალურად არსებული ფორმატი, ისეთი საიტი, როგორიც `youtube`-ა, სადაც მილიონობით ვიდეო არის, ყველა შემთხვევა უნდა იქნას გათვალისწინებული, ამიტომაც გარკვეული პრობლემა რომ არ წარმოიქმნეს მიეთითება სხვადასხვა ტიპი, იმისთვისა საჭირო რომ მომხარებელი ინტერნეტში სხვადასხვა ზომის, ტიპის თუ გაფართოების ვიდეო/აუდიო ფაილს ტვირთავს.

მაგალითი 1.61

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> video ტეგები</title>
</head>
<body>
<video poster="http://saiti.ge/surati.jpg" controls>
<source src="http://saiti.ge/video.webm" type="video/webm">
<source src="http://saiti.ge /video.ogv" type="video/ogg">
<source src="http:// saiti.ge /video.mp4" type="video/mp4">
</video>
</body>
</html>
```

შედეგი



რაც შეეხება *controls* ატრიბუტს, ეს ისეთი ატრიბუტია, რომელიც პასუხისმგებელია ხმის აწევა-დაწევაზე, დაპაუზებაზე და ა. შ. ანალოგიურად შესაძლებელია *DOM* მეთოდების, თვისებების გამოყენება.

ცხრილის სახით მოცემულია *video* ტეგის ატრიბუტები

ატრიბუტები	აღწერა
<i>autoplay</i>	ვიდეოს ჩართვა ავტომატურად
<i>controls</i>	ვიდეო კონტროლით ხდება განსაზღვრა, მაგალითი დაკვრა/დაპაუზება, ავტომატურად დაკვრა და ა.შ
<i>height</i>	პლეიერის სიმაღლა
<i>loop</i>	ჩაიკვლა, ანუ ვიდეო, როდესაც დამთავრდება, ყველა ჯერზე იწყებს თავიდან
<i>muted</i>	ხმის გათიშვა
<i>poster</i>	სანამ ვიდეო დაუკრავს, მანამდე სურათი გამოჩნდება
<i>preload</i>	როგორ ჩაიტვირთოს ვიდეო, ან თუ ჩაიტვირთება, იძახებს რა რეჟიმში მოხდეს გვერდის ჩატვირთვა: <i>auto</i> , <i>metadata</i> , <i>none</i>
<i>src</i>	ვიდეო ფაილის მისამართი
<i>width</i>	ვიდეო პლეიერის სიგანე

`<video>` ვიდეო ატრიბუტი, რომელიც მოცემული გვაქვს დამატებით განიხლვას არ საჭიროებს, გავეცნოთ `<source>` ტეგის ატრიბუტებს.

ატრიბუტი	აღწერა
<i>media</i>	რესურსის მედია ტიპი
<i>src</i>	დირექტორი-ფაილის მისამართი
<i>type</i>	რა ტიპის ფაილია

media ატრიბუტი, რომელიც მოცემულია ჯერ ჯერობით, მისი მხარდაჭერა არცერთ ბრაუზერს არ აქვს, ალბათ სამომავლოდ ბრაუზერში მისი ინტეგრაცია იგეგმება, ამიტომაც თანამედროვე ბრაუზერში უახლვეს მომავალში ვიხილავთ ამ ატრიბუტის შესაძლებლობას, მაგრამ ამ ეტაპზე მისი ინტეგრაცია ხდება სხვადასხვა მოწყობილობის ფაილების ოპტიმიზირებისათვის.

სინტაქსი:

`<source media="სიდიდე">`

ჩვენს ვიდეოს მაგალითს თუ ვნახავთ, მასში მოვახდინოთ *media* ატრიბუტის გამოყენება, მის ვიზუალიზაციას ვერ მოვახდენთ რადგან ბრაუზერებს როგორც ზემოთ ავღნიშნეთ არ აქვს მხარდაჭერა

```
<video poster="http://saiti.ge/surati.jpg" controls>
<source src="http://saiti.ge/video.webm" type="video/webm" media="screen
and (min-width: 500px)">
<source src=" http://saiti.ge/video.ogv" type="video/ogg" media="screen
and (min-width: 500px)">
<source src=" http://saiti.ge/video.mp4" type="video/mp4" media="screen
and (min-width:500px)">
```

ოპერატორები, რომელებიც შეიძლება გამოვიყენოთ:

and და ოპერატორი
not უარყოფილი ოპერატორი
, ან ოპერატორი

მოწყობილობები

სიდიდეები	აღწერა
<i>all</i>	ყველა მოწყობილობაზე
<i>aural</i>	საუბრის სინქრონიზაციას
<i>braille</i>	<i>Braille</i> მოწყობილობა
<i>handheld</i>	პატარა ეკრანი, პატარა გაფართოება
<i>projection</i>	პროექტორი
<i>print</i>	ბეჭდის ნახვა/ბეჭდის გვერდები
<i>screen</i>	კომპიუტერის ეკრანები
<i>tty</i>	ტერმინალები, პორტატული მოწყობილობები შეზღუდული ჩვენების შესაძლებლობები. მათთვის, გამოვიყენოთ, როგორც pixel ერთეული.
<i>tv</i>	ტელევიზორის ან მსგავსი მოწყობილობები.

სიდიდე

სიდიდე	აღწერა
<i>width</i>	სიგანე მაგალითი: <i>media="screen and (min-width:500px)"</i>
<i>height</i>	სიმაღლე მაგალითი: <i>media="screen and (max-height:500px)"</i>
<i>device-width</i>	ეკრანის/ქაღალდის გაფართოება სიგანე შეგვიძლია გამოვიყენოთ: მაგალითი: <i>media="screen and (device-width:500px)"</i>
<i>device-height</i>	ეკრანის/ქაღალდის გაფართოება სიმაღლე შეგვიძლია გამოვიყენოთ: მაგალითი: <i>media="screen and (device-height:500px)"</i>
<i>orientation</i>	ეკრანის/ქაღალდის ორიენტაცია სიდიდეები რომელიც მივუთითოთ: <i>"portrait"</i> ან <i>"landscape"</i> მაგალითი: <i>media="all და (orientation: landscape)"</i>
<i>aspect-ratio</i>	ეკრანის სიგანის/სიმაღლის გაფართოება მაგალითი: <i>media="screen and (aspect-ratio:16/9)"</i>
<i>device-aspect-ratio</i>	<i>device-width/device-height</i> გაფართოება ეკრანის/ქაღალდის მაგალითი: <i>media="screen and (aspect-ratio:16/9)"</i>
<i>color</i>	თითოეული ბიტი თითოეულ ფერზე ეკრანზე/ქაღალდზე მაგალითი: <i>media="screen and (color:3)"</i>
<i>color-index</i>	ფერების ინდექსი მაგალითი: <i>media="screen and (min-color-index:256)"</i>
<i>monochrome</i>	მონოქრომზე თითოეული ბიტი თითოეულ პიქსელზე : მაგალითი: <i>media="screen and (monochrome:2)"</i>
<i>resolution</i>	პიქსელის დანიშნულება (<i>dpi</i> ან <i>dpcm</i>) ეკრანის/ქაღალდი მაგალითი: <i>media="print and (resolution:300dpi)"</i>

<i>scan</i>	სკანირების მეთოდი, შესაძლო სიდიდეები „ <i>progressive</i> “ და „ <i>interlace</i> “. მაგალითი: <i>media="tv ან (scan:interlace)"</i>
<i>grid</i>	ინფორმაციის გამოტანა grid ან bitmap. სიდიდეები "1" grid-სთვის, "0" სხვა შემთხვევა. მაგალითი: <i>media="handheld და (grid:1)"</i>

Monochrome-აღწერს ფერწერას, გრაფიკას, დიზაინს, ან ფოტოს ერთი ფერში, ხოლო მონოქრომატული ობიექტის ან გამოსახულებას ასახავს ფერებში შეზღუდული ფერთა ჩვენებას.

ან აღწერს უბრალოდ ბმულს (ლინკს), სადაც ეს ფაილი მდებარეობს.
Type- რაც შეეხება, როგორც ზემოთ, აქაც განსაზღვრავს ტიპებს, ამ შემთხვევაში მხოლოდ გამოიყენება ცნობილი ტიპები, როგორიცაა:

- ვიდეო ტიპები:
video/ogg
video/mp4
video/webm
- აუდიო ტიპები:
audio/ogg
audio/mpeg

<track>

HTML5 ტრეკების მოიცავს ისეთ ელემენტებს, როგორიცაა: <video> და <audio> ელემენტები, ძირითადად გვხვდება სუბტიტრების სახით, თვითონ სიტყვა ითარგმნება, როგორც „მიყვება ნაკვალევს“ ან რაღაცის მიყოლა, მისი ფორმატი არის *WebVTT (.vtt)* რომელიც თავის მხრივ იშორება, როგორც *Web Video Text Tracks* (ვებ ვიდეო ტექსტის ტრეკერი), რომელის დეკოდირების სახით გვხვდება, როგორც *UTF-8* ხოლო *MIME* ტიპებში გვხვდება, როგორც *text/vtt*, ქვემოთ მოცემული ატრიბუტების ცხრილის საფუძველზე დავეწყოთ სკრიპტი და გამოვიყენოთ ყველა ატრიბუტი, ხოლო განხილვა ეტაპობრივად მოვახდინოთ.

ატრიბუტები	აღწერა
<i>default</i>	მრავალი სხვადასხვა ენის სუბტიტრების მთავარი ენის სუბტიტრი, რომელიცაა
<i>kind</i>	ტრეკის ნაირსახეობა
<i>label</i>	ტრეკის სათაურის ტექსტი
<i>src</i>	ფაილის მისამართი
<i>srcLang</i>	ტრეკის ტექსტის ინფორმაცია ისეთი, როგორიცაა სუბტიტრები

განვიხილოთ თითოეული ატრიბუტი მათივე მნიშვნელობებით:

default

ატრიბუტის გამოყენებისას ენა, რომელიც არის მითითებული, ის იქნება პრიოტოტული და შემდეგ მეორე ან მესამე ენა.

kind

- ✔ *Captions*-განსაზღვრავს ხმის ასევე ტექსტურ ეფექტებს;
- ✔ *Chapters*-მედია ფაილს სათაურებად დაყოფს;
- ✔ *Subtitles*-ქვე ტექსტები, რომელიც აუდიო მოსმენის დროს გაუგებერია მაყურებლისათვის;
- ✔ *Metadata*-იყენებს რაღაც სკრიპტს, რომელიც მომხარებლები ვერ ხედავენ;
- ✔ *Descriptions*-ვიდეო კონტენტის აღწერა;

srcLang

განსაზღვრავს რა ენას თუ რომელ ენაზე არის სუბტიტრებს, რაც შეეხება დანარჩენა ტრიბუტებს, არ შეეხებით რადგან უკვე განხილული გვაქვს.

მაგალითი 1.62

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> video ტეგები</title>
</head>
<body>
  <video poster="http://saiti.ge/surati.jpg" controls
  preload="metadata">
    <source src="http://saiti.ge/video.webm" type="video/webm">
    <source src="http://saiti.ge/video.ogv" type="video/ogg">
    <source src="http://saiti.ge/video.mp4" type="video/mp4">
    <track src="http://saiti.ge/wigni/en.vtt" kind="subtitles"
    srcLang="en" label="English" default>
    <track src="http://saiti.ge/wigni/qa.vtt" kind="subtitles"
    srcLang="ka" label="georgian">
  </video>
</body>
</html>
```

<audio>

აუდიო ტიპის ფაილებზე პასუხისმგებელი ტეგი, ანალოგიური არის ვიდეო ტეგისა და განსხვავდება მათ შორის აუდიო ტიპების მითითებაა, მაგალითისათვის ავიღოთ რაიმე მუსიკა და ჩავსვათ, *<source>* ტეგში არსებული მისამართებიდან, რომელიც იქნება მისთვის გასაგები ან რომელ ფაილსაც წაიკითხავს, იმ ფაილს დაუკრავს.

```
<audio controls>
```

```
<source src="მუსიკა.ogg" type="audio/ogg">
```

```
<source src="მუსიკა.mp3" type="audio/mpeg">
</audio>
```

მისი ტიპებია:

```
MP3-audio/mpeg
Ogg-audio/ogg
Wav-audio/wav
```

<object>

HTML დოკუმენტში **<object>** გამოყენება ხდება რაიმეს ჩასანერგად, როგორც არის *java* აპლეტები, *PDF readers*, *Flash Players*, ან პლაგინები უნდა ითქვას ისიც, რომ მასში ასევე სხვა HTML დოკუმენტის მოთავსებაც შეიძლება, HTML5-ში წინა ვერსიისგან განსხვავებით, მას არ აქვს ატრიბუტები.

მაგალითი 1.63

```
<!DOCTYPE html>
<html>
<body>
  <object width="300%" height="500px" data="http://gtu.ge">
  </object>
</body>
</html>
```

გამოიტანს ფორუმს ან სხვა რაიმე HTML დოკუმენტს, რასაც მივუთითებთ, ასევე შესაძლებელია ფლემ ვიდეო/ანიმაციის მითითება და ჩასმა.

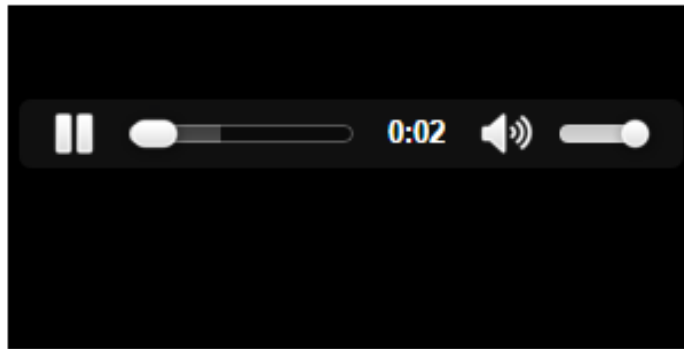
<param>

<param> ელემენტის საშაულებით ეთითება პარამეტრები, რომელიც ინერგება შემდგომ **<object>** ტეგში, მაგრამ თუ რა ტიპის პარამეტრები ეთითება, დამოკიდებულია რა ტიპის ობიექტია, მაგალითად, თუ ობიექტმა უნდა ჩატვირთოს *Flash MP3 player* გვერდში, მაშინ ეთითება ფაილის ტიპი როგორც არის MP3 რომ შემდეგ იპოვოს და დაუკრას, მაგრამ თუ გვაქვს ვიდეო ფაილი, რომელიც უნდა ჩაიტვირთოს ან სანამ მომხმარებელი დაკვრის ღილაკს დააჭერდეს ავტომატურად დაუკრას, ეთითება შესაბამისი პარამეტრები, მისი ატრიბუტებია *name* და *value*.

მაგალითი 1.64

```
<!DOCTYPE html>
<html>
<head>
<title> param ტეგი
</title>
</head>
<body>
<object data="http://saiti.ge/musica.mp3">
<param name="autoplay" value="true">
</object>
</body>
</html>
```

შედეგი



<iframe>

HTML დოკუმენტში უნდა მოახდინოს სხვა საიტის, ფაილის ან არსებული აუდიო/ვიდეო ფაილის მიმაგრება, მისი ატრიბუტებია: *name*, *sandbox*, *src* და *width*, ჩვენთვის უცხო ატრიბუტია *sandbox*, რომელიც პასუხისმგებელია ფაილის გაგზავნაზე, მისი ატრიბუტია:

<i>allow-forms</i>	ფორმაზე ნების დართვა
<i>allow-pointer-lock</i>	APS- ნების დართვა
<i>allow-popups</i>	ნების დართვა <i>popups</i>
<i>allow-same-origin</i>	<i>iframe</i> შინაარსი განიხილება, როგორც ერთისა და იმავე წარმოშობის
<i>allow-scripts</i>	სკრიპტების ნებართვა
<i>allow-top-navigation</i>	ნებას რთავს სკრიპს და წვდომა აქვს სერვერზე

მაგალითი 1.65

```
<!DOCTYPE html>
<html>
<body>
<iframe src="რაიბე.htm" sandbox="allow-same-origin allow-scripts">
</iframe>
</body>
</html>
```

HTML 5-ს აქვს ბევრი შესაძლებლობა და მათ ქვემოთ შევხებით, რაც შეეხება თვითონ HTML5-ს, განისაზღვრება მოცემული ფორმულით *HTML5=CSS3+Javascript*, ქვემოთ ცხრილში მოცემულია ყველა ელემენტი, რომელიც HTML 5-ში სიახლეს წარმოადგენს.

ახალი სემანტიკა/სტრუქტურული ელემენტები

ტეგები	აღწერა
<article>	განსაზღვრავს ერთგვარ სათაურს დოკუმენტში
<aside>	გვერდის ჩანართი
<bdi>	განისაზღვრება ტექსტის ნაწილი, რომელიც შეიძლება ნაწილობრივ დაფორმატდეს და მიმართულება შეეცვალოს

<details>	დეტალებით მომხმარებელს შეუძლია გამოჩინოს ან დამალოს
<dialog>	დიალოგური ველი
<figcaption>	<figure> ელემენტში განსაზღვრავს სურათს
<figure>	კონტეინერი, რომელიც წარმოდგება როგორც ილუსტრაცია, ხაზვის და ა.შ.
<footer>	ქვედა კოლენტეტური
<header>	დოკუმენტში „სათაური“
<main>	განსაზღვრავს დოკუმენტში ძირითად კონტენტს.
<mark>	ტექსტის მონიშვნა
<menuitem>	მენიუს ღილაკები, რომელთა ჩამატებაც შეგვიძლია
<meter>	სკალარული საზომი
<nav>	ნავიგაცია
<progress>	პროგრესი
<rp>	<ruby> ტექსტს განსაზღვრავს ბრაუზერში
<rt>	დასავლეთ აზიის სიმბოლოების აღსაქმელად გამოყენება
<ruby>	აზიური ანოტაცია (for East Asian typography)
<section>	დოკუმენტს განსაზღვრავს სექციებად
<summary>	ხილულია მხოლოდ <details>
<time>	თარიღი/დრო
<wbr>	წყვეტა

ახალი ფორმის ელემენტები

ტეგები	აღწერა
<datalist>	განსაზღვრავს შეტანის კონტროლერს
<keygen>	ველის წყვილი გასაღები
<output>	გამოაქვს კალკულაციის შედეგი

Input ტიპები და ატრიბუტები

ახალი input ტიპები	ახალი ატრიბუტები
• color • date • datetime • datetime-local • email • month • number	• autocomplete • autofocus • form • formaction • formenctype • formmethod • formnovalidate

• <i>range</i> • <i>search</i> • <i>tel</i> • <i>time</i> • <i>url</i> • <i>week</i>	• <i>formtarget</i> • <i>height and width</i> • <i>list</i> • <i>min and max</i> • <i>multiple</i> • <i>pattern (regexp)</i> • <i>placeholder</i> • <i>required</i> • <i>step</i>
---	---

მედია ელემენტები

ტეგები	აღწერა
<audio>	ტეგი განსაზღვრავს ხმასა და მუსიკას
<embed>	პლაგინის სახით გვევილენება, დოკუმენტში ჩანერგვა
<source>	სოურსი <video> და <audio> ტეგების
<track>	ტრეკები <video>და <audio>
<video>	კინოს ან ვიდეოს აღმწერელი ტეგი

თავი II

CSS მიმოხილვა

CSS ანუ *Cascading Style Sheet*-კასკადური სტილის ენაა, რომელიც, განსაზღვრავს დოკუმენტის დიზაინსა და სტილს, ჩვენ იერარქიულად ყველა აქტიურ ტეგებსა და ატრიბუტებს განვიხილავთ, (*CSS1, CSS2* და *CSS3-ში*), რომელიც დღემდე აქტიურია, CSS შექმნის მიზანი იყო, რომ *Web* გვერდები, რომელიც იქმნება, წარმოჩენილიყო უფრო ეფექტურად და ლამაზად.

ვებ გვერდების წარმოჩენა ინტერნეტში დღესდღეისობით აქტუალური თემაა, და ამის შესაძლებლობას CSS ენა იძლევა მოიცავს ისეთ კონტროლერებს, როგორიცაა ფონტის ფერი, მენიუს კიდეები, უკანა ფონი, პარაგრაფებს შორის სივრცეს და ა.შ

CSS მარტივი სასწავლია, თუ როგორ წარმოჩნდეს ვებ გვერდები სტილისტურად, ზოგავ დროს რომ გააფორმო საიტის გვერდი, ყოველ ჯერზე რომ არ მოგვიწიოს *HTML* ტეგის ატრიბუტების წერა, მაგალითად, სადაც გამოყენებულია *<p>*, ტეგი წარმოჩნდეს, როგორც წითელი ფერი, ყველა ჯერზე *HTML* დოკუმენტში ფერების წერისაგან თავს ვარიდებთ და ამ საკითხს გადავწყვეტთ *CSS*-ით.

HTML და *CSS* უფრო გასაგებად რომ ვთქვათ, *HTML*-ის საშუალებით გვერდი აღიწერება შინაარსობრივად, ხოლო *CSS* საშუალებით, სტილისტურად.

მაგალითად 2.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
p {
    color: red;
    text-align: left;
}
A
{
color:green;
}
</style>
</head>
<body>
<p> პარაგრაფის დემონსტრირება </p>
<a> ვიწყებთ CSS!</a>
<p> აღიწერება პარაგრაფი </p>
</body>
</html>
```

შედეგი

პარაგრაფის დემონსტრირება

ვიწყებთ CSS!

აღიწერება პარაგრაფი

ვხედავთ, რომ სადაც პარაგრაფის ტეგი, ნახა ყველა ტექსტი გაწითლდა (როცა სკრიპტს აკრეფთ და გაუშვებთ ბრაუზერში), თვითონ CSS გამოჩნდა 1994 წლის 10 ოქტომბერს, რომლის შექმნის იდეა გახლდათ ნორვეგიელი მეცნიერის *Håkon Wium Lie*-ს და შესთავაზა W3C ჯგუფს, რომ შეექმნათ *Cascading Style Sheets* (ანუ შემოკლებით CSS) ენა.

Cascading Style Sheets პირველი ვერსია ანუ CSS1, რომელის რეკომენდაციას W3C უწევს, ჩნდება ოფიციალურად უკვე 1996 წლის დეკემბრიდან, რომელიც მარტივად აღწერდა HTML-ის ტეგებს, მაღევე პოვა პოპულირირება და შემუშავდა მეორე ვერსია (CSS2) 1998 წლის მაისში, ასევე W3C რეკომენდაციით ოფიციალურად უკვე ჩამატებული იყო სხვადასხვა მოწყობილობის მხარდაჭერა, ფონტების აღქმა და ცხრილები და ა.შ, ხოლო CSS3 ანუ მესამე ვერსია 1999 წლის ივნისიდან გაჩნდა, რომელიც აგებს და ამავე დროს ხვეწს წინა ვერსიის ვებ გვერდებს, მისი შესაძლებლობები, როგორც ძველი ისე ახალი შესაძლებლობები, შემოიფარგლება:

- ✓ *Selectors*
- ✓ *Box Model*
- ✓ *Backgrounds and Borders*
- ✓ *Image Values and Replaced Content*
- ✓ *Text Effects*
- ✓ *2D/3D Transformations*
- ✓ *Animations*
- ✓ *Multiple Column Layout*
- ✓ *User Interface*

და სხვა.

CSS 3 ყველა აქტუალურ საკითხებს გავივლით, მაგრამ საწყის ეტაპზე საჭიროა გავეცნოთ მის სინტაქსს.

სინტაქსი არის წერის მეთოდი, HTML5-ისგან განსხვავებული სინტაქსი აქვს, მის სინტაქსში იქმნება ერთგვარი „წესები“, რითაც შესაძლებელია რომ აღიწეროს ესა თუ ის მოვლენა და მიენიჭოს გარკვეული პარამეტრი, მაგალითად, სურათის ზომა, დახრილობა, გამჭვირვალება ან ტექსტის შრიფტის ზომა, სხვადასხვა ალტერნატიული ფონტების მითითება, ძირითადად ეს უკანასკნელი ლათინური ალფაბეტის განლაგებისას ხდება, თუმცა ქართულშიც არსებობს სხვადასხვა ფონტი, ყველაზე აქტუალურია და ხშირად გამოყენებადია ევრანული კლავიატურა, რომელსაც „*Sylfaen*“ სახელწოდებით ვიცნობთ, ასევე *AcadNusx*, *LitNusx* და სხვ.



CSS სინტაქსი

CSS სინტაქსის წერის დროს, ხდება გარკვეული წესის შემუშავება, თუ როგორ უნდა დაიწეროს სკრიპტი და ამ წესებს აუცილებად მოიაზრება სამის პარამეტრით, ესენია:

- **Selector**- სელექტორი ეს არის ის რაც უნდა მოინიშნოს, ანუ HTML ტეგი რომელიც იქნება არჩეული, მაგალითად, თუ გვაქვს `<h1>`, მაშინ ჩვენი სელექტორი იქნება `h1`, თუ `<p>` ანუ პარაგრაფი, მაშინ პარაგრაფი და ა.შ.
- **Property**-თვისებაში იგულისხმება HTML ტეგის ტიპის ატრიბუტის თვისება, მარტივად რომ ვთქვად, ყველა ატრიბუტი ფაქტობრივად უკვე განისაზღვრება CSS თვისებით, ეს იქნება საზღვრები, ფერები, სურათი პიქსელები და ა. შ
- **Value**-სიდიდე ანუ იგულისხმება, მაგალითად უნდა იყოს ტექსტის პოზიციები: `text-align: Left` ანუ მარცხივ იქნება, აქედან გამომდინარე ჩვენი *value* არის *Left*.

CSS სინტაქსი ნაჩვენებია, თუ როგორ ხდება მისი დეკლარირება ანუ აღწერა, რომელი ელემენტი არის *Property*, რომელი *Value* და *Selector*-ი.



პირველ რიგში შევხებით სელექტორებს ქართულად ითარგმნა როგორც „მონიშნული“, მაგრამ სელექტორი ტერმინად არის დამკვიდრებული, ამიტომაც ასე მოვიხსენიოთ ჩვენც, რაც შეეხება ამ ელემენტს არსებობს რამდენიმე სელექტორი.

ტიპობრივი ან ელემენტის სელექტორი

ასე ვუწოდოთ სელექტორს რომელიც ერთდოულად აღწერს როგორც ცალკე ისე რამდენიმე HTML-ტეგს

```
p {  
    text-align: center;  
    color: red;  
}
```

უნივერსალური სელექტორი

უნივერსალური სელექტორის აზრი ის არის, რომ რადგან ასე თუ ისე, ყველა ტეგის თუ ელემენტის განისაზღვრა მეტ-ნაკლებად ხდება, შექმნილია უნივერსალური სელექტორი, მაგალითად, გაწერილი პარაგრაფის, ცხრილის კლასების თუ *Id*-ს, გარდა ყველა დანარჩენი, უნდა იყოს განსაზღვრული რაღაცით, მაგალითად, ფერით, თუ ზემოთ თქმულიდან ყველას აქვს თავის ფერი, ზომა-წონა, სხვა დანარჩენს

ქონდეს ერთი ფერი, ეს გვადლევს საშუალებას კონკრეტული ელემენტები ჰგანვსაზღვროთ, დავუშვათ ისევ კონკრეტული ფერებით, ხოლო სხვა დანარჩენი მოვაქციოთ ერთ ფერში, მისი აღმინიშნელი სიმბოლოა *- ფიფქია.

სინტაქსი:

```
* {  
    color: #000000;  
}
```

შევქმნათ სკრიპტი, სადაც იგივე `<p>` და `<h1>` იქნება განსაზღვრული ლურჯი ფერით, ხოლო სხვა დანარჩენი წვანე ფერით.

მაგალითი 2.2

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
<style>  
h1 {  
    color:blue;  
}  
p {  
    color:blue  
}  
  
* {  
    color:green;  
}  
}  
</style>  
</head>  
<body>  
<div>  
    <h1> აქ მოცემული სელექტორი განისაზღვრება ლურჯად </h1>  
    <p> ფერები, რომლებიც ამის მერე მწვანედ წარმოჩდება, არის უნივერსალური  
</p>  
    <ul>  
        <li> CSS <em>უნივერსალური</em> სელექტორია </li>  
        <li>აქაც წარმოჩდება <em>უნივერსალური</em> სელექტორი</li>  
    </ul>  
    ვფიქრობთ, არ არის ძნელი გასაგები :)  
</div>  
</body>  
</html>
```

შედეგი

აქ მოცემული სელექტორი განისაზღვრება ლურჯად

ფერები, რომლებიც ამის მერე მწვანედ წარმოჩდება, არის უნივერსალური

- CSS უნივერსალური სელექტორია
- აქაც წარმოჩდება უნივერსალური სელექტორი

ვფიქრობთ არ არის ძნელი გასაგები :)

დესცენდენტის სელექტორი

დესცენდენტი სელექტორი ანუ „დადმავალი“ სელექტორი, გამოიყენება როდესაც ტეგში ტეგია მოცემული და მისი აღწერა გვინდა, ილუსტრირებისათვის ხშირია გამოიყენება `<em>` მისი გამოყენება `<ul>` ტეგში.

```
ul em
{
  color: #000000;
}
```

ამით ვეუბნებით, რომ `ul` ტეგში მოთავსებული `em` ელემენტი იყოს შავი, მაგალითად ავიღოთ რაიმე ტექსტი, რომელიც იქნება წითლად, ავიღოთ მაგალითი 2.2 და მასში ჩაწერილი ტექსტი „ვფიქრობ არ არის ძნელი გასაგები“ წარმოვაჩინოთ სიტყვა „ძნელი“ წითლად, მოვახდინოთ `<style>` სკრიპტის დახვეწა, პირველადი ფორმა სახე გვაქვს შემდეგი სახის:

```
<style>
h1 {
  color:blue; }
p {
  color:blue
}
* {
  color:green;
}
</style>
```

შეგვიძლია `h1` და `p` ერთად აღვწეროთ, რადგან ორივე ერთ ფერში გვსურს, რომ წარმოვაჩინოთ, ამის საშუალებას „მძიმე“ გვაძლევს, საერთოდ, თუ ორ ან მეტ ელემენტის აღწერა ერთი და იმავე სიდიდით ხდება, შეგვიძლია ერთად წარმოვაჩინოთ, რაც შეეხება ჩვენს მაგალითს, ჩავამატოთ წითელი ფერი.

```
h1,p {
  color:blue;
}
* {
  color:green;
}
h1 em {
  color:red;
```

}

მთლიანად სკრიპტს კი ასეთი სახე აქვს:

მაგალითი 2.3

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
h1,p {
color:blue;
}

* {
color:green;

}
h1 em
{
color:red;
}
</style>
</head>
<body>
<div>
<h1> აქ მოცემული სელექტორი განისაზღვრება ლურჯად</h1>
<p>ასევე ლურჯად მაგრამ, ფერები, რომლებიც ამის მერე მწვანედ წარმოჩდება,
არის უნივერსალური</p>
<ul>
<li>CSS <em>უნივერსალური</em> სელექტორი</li>
<li>აქაც წარმოჩდება <em>უნივერსალური</em> სელექტორი</li>
</ul>
<H1> ვფიქრობთ არ არის :<EM>ძნელი</EM> გასაგები</H1>
</div>
</body>
</html>
```

კლასის სელექტორი

კლასის სელექტორი განსაზღვრავს კლასში არსებული ატრიბუტებს და წარმოაჩენს სხვადასხვანაირად, თუ გვაქვს სამი კლასი სამივე კლასის წარმოჩენა შეგვიძლია სხვადასხვა ფრად, *div class* აღწერილი, კლასები CSS-ში აუცილებლად მიეთითება წერტილი ანუ (.), რომ მასთან დაკავშირება მოხდეს.

მაგალითად 2.4

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```

<style>
p.ცენტრი
{
    text-align: center;
    color: red;
}
p.დიდად
{
    font-size: 200%;
}
h1.მწვანედ {
color:green;
}
</style>
</head>
<body>
<h1 class="მწვანედ">გამოჩნდება ტექსტი მწვანედ</h1>
<p class="ცენტრი"> გამოჩნდება შუაში და წითლად</p>
<p class="ცენტრი დიდად"> ტექსტი გამოჩნდება დიდად და წითლად.</p>
</body>
</html>

```

შედეგი

გამოჩნდება ტექსტი მწვანედ

გამოჩნდება შუაში და წითლად

ტექსტი გამოჩნდება დიდად და წითლად.

სურ 2.2

<p class-ში თუ ერთზე მეტი სახელი ჰქვია, როგორც ჩვენს მაგალითში ხშირად აღწერენ, ერთსა და იმავე დასახელების კლასს HTML კოდში title-ში, მაგალითად, სკრიპტის კონკრეტული სტრიქონი ასე იქნებოდა:

```

<p class="ცენტრი დიდად" title="ტესტ">
    ტექსტი გამოჩნდება დიდად და წითლად.</p>

```

ხოლო სტილში კი შემდეგნაირად ჩაიწერებოდა:

```

p.დიდად [title^="ტესტ"]
{
    font-size: 200%;
}

```

და შედეგი იგივე იქნება, დაისმის ლოგიკური კითხვა, თუ იმავე შედეგს გამოიტანს, რაც სურათზე 2.2-ზე ნაჩვენებია, მაშინ რატომ არ წარმოიქმნა სინტაქსური შეცდომა ან კიდევ, რა საჭიროა ამის დამატება? როდესაც დიდ კოდს ვწერთ, ისეთი საიტისა, როგორიც არის facebook ან სხვა სოციალური საიტი, თითოეული კლასისათვის გვიწევს ერთსა და იგივე სახელის დარქმევა, რომ შემდეგ პროგრამისტს

არ აერიოს, შემოაქვთ განმასხვავებელი სიდიდეები, ამატებენ სახელში სხვა სიტყვას ან სიმბოლოს, ჩვენს შემთხვევაში, ილუსტრირებისათვის ჩავამატეთ `[title^="ტესტ"]`, რაც შეეხება ჩვენს პირველად სახეს, როდესაც წერია ერთსა და იმავე სახელის კლასი:

```
<p class="ცენტრი">
<p class="ცენტრი დიდად">
```

აღწერილი გვაქვს „ცენტრი“ მეორეჯერ ვუთითებ „დიდად“ , რადგან პირველი შემთხვევა მისთვის გასაგებია, ხოლო მეორე შემთხვევისას წარმოქმნება სინტაქსური შეცდომა, რადგან კლასში ჩამატებული სახელი უკვე იგებს, როგორც განსხვავებულ კლასს და არაფერს გამოიტანს, როდესაც ამ განსხვავებას, როგორია „დიდად“ აღწერთ, მისთვის გასაგებია, ამიტომაც უთითებენ *HTML* სკრიპტში აღწერილ კლასში არსებულ ატრიბუტებს, რომ *CSS*-მა გაიგოს, რომელი სტრიქონი როგორ წარმოაჩინოს.

ID სელექტორი

`<div id=""`, რომელიც გვაქვს მოცემული, აღიწერება *ID* სელექტორის საშუალებით, *CSS*-ში მასთან დასაკავშირებლად გამოიყენება დიეზი (#).

მაგალითი 2.5

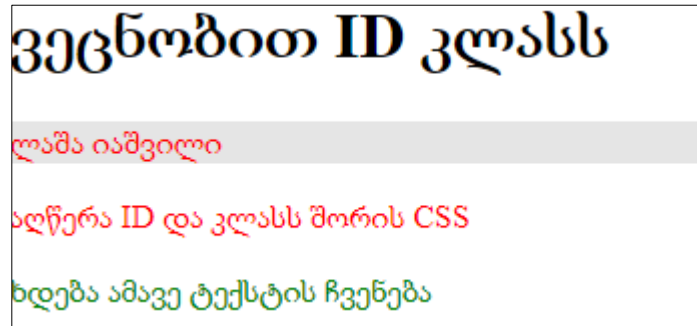
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
#სახელი
{
background-color: #e5e5e5;
color:red;
}
#გაცნობა
{
color:Red;
}
div.ტესტ
{
color:green;
}
</style>
</head>
<body>
<h1> ვეცნობით ID კლასს </h1>
<div id="გაცნობა">
  <p id="სახელი"> ლაშა იაშვილი </p>
  <p id="აღწერა"> აღწერა ID და კლასს შორის CSS </p>
```

```

</div>
<div class="ტესტ">
<p> ხდება ამავე ტექსტის ჩვენება </p>
</body>
</html>

```

შედეგი



შვილობილი სელექტორი

შვილობილი სელექტორი არის ერთგვარი „დადმავალი“ სელექტორი, რომელიც ტეგში მოიცავს სხვა ტეგს და ამავე ტეგის გაფორმება ხდება, სხვანაირად, მას კომბინირებულ სელექტორსაც უწოდებენ და „>“ სიმბოლოთი აღნიშნავენ. მისი სინტაქსი ასე გამოყენდება:

```
selector1 > selector2 { აღწერა }
```

მაგალითად 2.6

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
ol > li
{
    color: red;
}
</style>
</head>
<body>
<body>
<div>
<ol>
<li>მოინიშნება</li>
<li>მოინიშნება</li>
<ul>
<li>არ მოინიშნება</li>
<li>არ მოინიშნება</li>
</ul>
<li>მოინიშნება</li>
</ol>

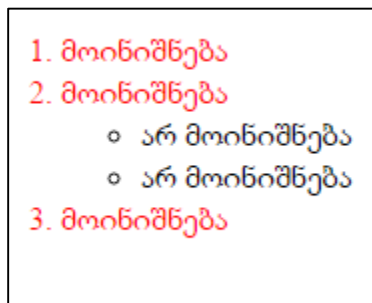
```

```

</div>
</body>
</html>

```

შედეგი



ატრიბუტის სელექტორები

ატრიბუტებს, რომელნიც შეიცავს *HTML* დოკუმენტები აღიწერება, ხდება კონკრეტული ელემენტის ატრიბუტის (ან ატრიბუტების) აღწერა.

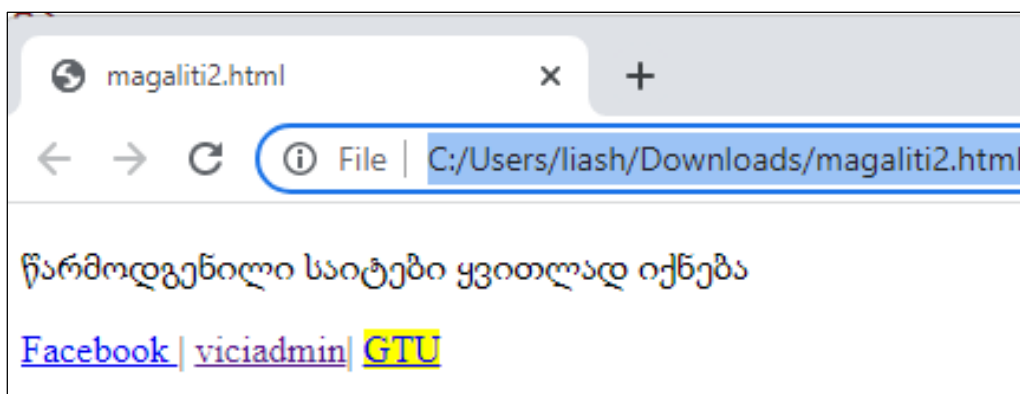
მაგალითად 2.6

```

<!DOCTYPE html>
<html>
<head>
<style>
a[target]
{
    background-color: yellow;
}
</style>
</head>
<body>
<p>წარმოდგენილი საიტები ყვითლად იქნება
</p>
<a href="http://www.facebook.com"> Facebook
</a>|
<a href="https://viciadmin.gtu.ge/#/login/login">
viciadmin</a>|
<a href="http://www.gtu.ge" target="_top">GTU
</a>
</body>
</html>

```


შედეგი



ამის გარდა, ატრიბუტთან დაკავშირებით, არსებობს კიდევ ჩაწერის რამდენიმე წესი, რომელიც უნდა აუცილებლად ვიცოდეთ, ატრიბუტების სელექტორებს ჩვენ კიდევ დავუბრუნდებით, მაგრამ თვალნათლივ წარმოჩენისათვის ამ ეტაპზე რამდენიმე მაგალითი განვიხილოთ:

	<code>[target]</code>	მონიშნავს ყველა ელემენტის <code>target</code> ატრიბუტს;
	<code>[target=_blank]</code>	მონიშნავს მხოლოდ <code>target="_blank"</code> ატრიბუტს;
	<code>[title~=მანქანა]</code>	<code>title</code> -ში აღწერილი საკვანძო სიტყვა „მანქანა“ იპოვის;
	<code>[Lang =ka]</code>	მონიშნავს ყველა იმ ენას, სადაც იქნება <code>ka</code> ნახსენები;
	<code>a[href^="https"]</code>	მონიშნავს ყველა <code><a></code> ელემენტს <code>href</code> ატრიბუტის სიდიდეს, რომელიც იწყება <code>"https"</code> .
	<code>a[href\$=".pdf"]</code>	მონიშნავს ყველა <code><a></code> ელემენტის ყველა ატრიბუტს, რომელიც მთავრდება <code>".pdf"</code> ;
	<code>a[href*=gtu.ge]</code>	მონიშნავს ყველა ელემენტს, სადაც ნახსენები იქნება <code>gtu.ge</code> ;

დაჯგუფებული სელექტორები

ასეთი ტიპის სელექტორებს მიეკუთვნება ყველა სელექტორი, რომელიც ერთ ბლოკში ხდება მათი აღწერა, თუ ყველა ელემენტს აქვს ერთი და იგივე ზომა და წონა, სკრიპტისა და ასევე პროგრამისტისათვის ადვილად წასაკითხი რომ გახდეს, ერთად ხდება მათი დაჯგუფება და აღწერა, პრაქტიკაში სელექტორების მსგავსი წარმოდგენა ხშირად გვხვდება და უფრო პრაქტიკულია.

მაგალითი 2.7

```
#header, #footer, #aside
{
    position: absolute;
    left: 510px;
    width: 200px;
}
```

CSS შეტანის მეთოდები

CSS რომ ავლწეროთ სამი მეთოდი პირველი მეთოდია როდესაც ხდება სხვა ბმულზე გადამისამართება, სადაც CSS ფაილი არსებობს და ხშირად ვხდებით, როგორც *style.css*, რატემა უნდა, სასურველი სახელის დარქმევა შეგვიძლია, მაგრამ ფაილის ფართობას უცვლელად ტოვებს, მხოლოდ სახელი იცვლება:

```
<Link rel="stylesheet" type="text/css" href="style.css">
```

მეორე მეთოდი არის თვითონ HTML დოკუმენტში სტილის აღწერა, რომელიც `<style>` `</style>` ტეგებში აღიწერება, რომლის წინა მაგალითებში შევხვდით, ხოლო მესამე სტრიქონული გაფორმება, რომელიც ასევე შიგ დოკუმენტში ფორმდება, მაგალითად, ასე:

```
<html>
<head>
<meta charset="utf-8">
</head>
<body>
  <h1 style = "color:#36C;"> სტრიქონული CSS </h1>
</body>
</html>
```

`<Link>` საშუალებით ხდება ფაილის მისამართის გაწერა ასევე და გასაგებია, რომ არსებობს სხვა მეთოდიც. ასევე ხდება `@import` საშუალებით და შესაძლებელია, რომ დიზანის ფაილის მიმაგრდეს, მოცემული მეთოდი უფრო მარტივი და დახვეწილია.

სინტაქსი:

```
<head>
@import url(მისამართი, სადაც არსებობს ფაილი)
</head>
```

CSS საზომი ერთეულები

სიგრძის საზომის ერთეულებად, გარდა ცნობილი ერთეულებისა, როგორიცაა: *em, ex, %, px, cm, mm, in, pt, pc, ch, rem, vh, vw, vmin, vmax* ხოლო ისეთი სიგრძის საზომი ერთეულები, როგორიცაა სანტიმეტრი ან მილიმეტრი, რატემა უნდა, მათ არ განვიხილავთ.

სტანდარტულად საზომ ერთეულად აღებულია *16px* და მის გარშემო ხდება საზომი ერთეულების კონვერტაცია მაგალითად, *0.75em* ერთეული არის იგივე *12/16px*, ანუ ეს ის შემთხვევა, როდესაც 12-ტიანი შრიფტით ვწერთ, თუ გვინდა პიქსელებში გადაყვანა, მაშინ $0.75em * 16px = 12$

აბსოლუტურ სიდიდეებში შედის ისეთი სიდედეები, რომელიც ეკრანის მიმართ არ არიან მოქნილები და ზუსტად აღწერენ მოცემულ ზომას, ძირითადად ასეთი სიდიდეები გამოყენება მაგალითად, ბეჭდვის ფანჯრის გამოძახების დროს:

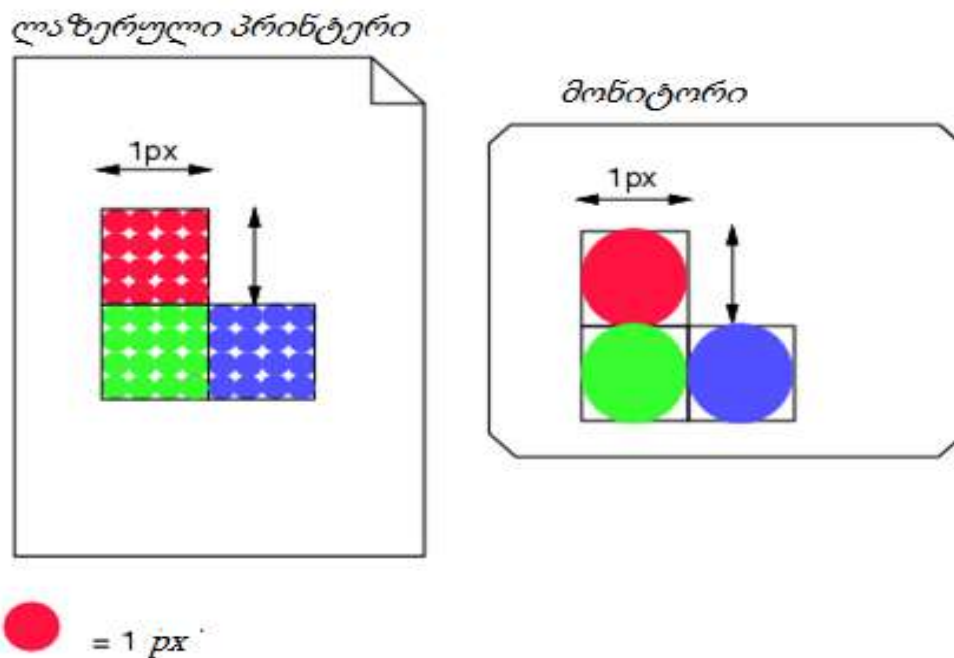
- ✓ cm(სანტიმეტრი)
- ✓ mm (მილიმეტრი)
- ✓ in (ინჩი, რომელიც არის $1in = 96px = 2.54cm$)
- ✓ px (პიქსელი, რომელიც არის $1px = 1/96th$)
- ✓ pt ($1pt = 1/72$ დან $1in$)
- ✓ pc (picas $1pc = 12 pt$)

ცხრილიში მოცემულია:

PX	EMs	Pt
6px	<i>0.375em</i>	<i>5pt</i>
7px	<i>0.438em</i>	<i>5pt</i>
8px	<i>0.500em</i>	<i>6pt</i>
9px	<i>0.563em</i>	<i>7pt</i>
10px	<i>0.625em</i>	<i>8pt</i>
11px	<i>0.688em</i>	<i>8pt</i>
12px	<i>0.750em</i>	<i>9pt</i>
13px	<i>0.813em</i>	<i>10pt</i>
14px	<i>0.875em</i>	<i>11pt</i>
15px	<i>0.938em</i>	<i>11pt</i>
16px	<i>1.000em</i>	<i>12pt</i>
17px	<i>1.063em</i>	<i>13pt</i>
18px	<i>1.125em</i>	<i>14pt</i>

ცხრილი 2.1 მოცემულია აბსოლუტური და რელატიური სიდიდეები

px ანუ იმავე პიქსელის გასაცნობად და თვითონ ამდენი საზომი ერთეულების აღსაქმელად საჭიროა, გავეცნოთ თვითონ პიქსელს, სურათზე 2.6 ნაჩვენებია, თუ ერთი პიქსელი რამდენ წერტილს მოიცევას სხვადასხვა მოწყობილებაზე, ჩვენს შემთხვევაში ნაჩვენებია დაბალი რეზოლუციის შემთხვევა.



სურათი 2.6 დაბალი რეზოლუციის დროს საჭიროა მეტი წერილი, რათა შეივსოს 1px ხოლო მაღალი რეზოლუციისას კი ნაკლები წერტილებია საჭირო

რელევანტურ სიდიდეებში შედის შედის ყველა დანარჩენი სიდიდე.

ცხრილი 2.2 რელევანტური სისიდიდეები

სიდიდე	რელევანტური
<i>em</i>	ფონტის ზომა
<i>ex</i>	ფონტის სიმაღლე
<i>ch</i>	ფონტის ზომის სიგანე 0 ნიშნიდან
<i>rem</i>	ფონტის ზომა
<i>vw</i>	სიგანე (<i>viewport's width</i>)
<i>vh</i>	სიმაღლე <i>viewport's height</i>
<i>vm</i>	მინიმუმი სიმაღლე და სიგანე

ადვილად აღსაქმელი, რომ იყოს ისეთი საზომი ერთეული როგორცაა *ex* მაჩვენებია სურათი 2.6

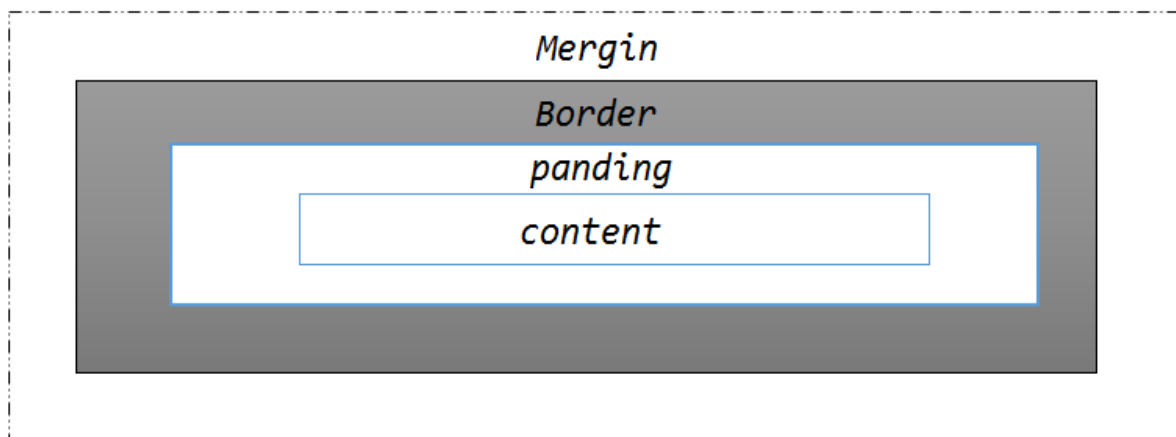
The quick brown fox x ↕ x-height ↕ font-size

სურ 2.6

Box model

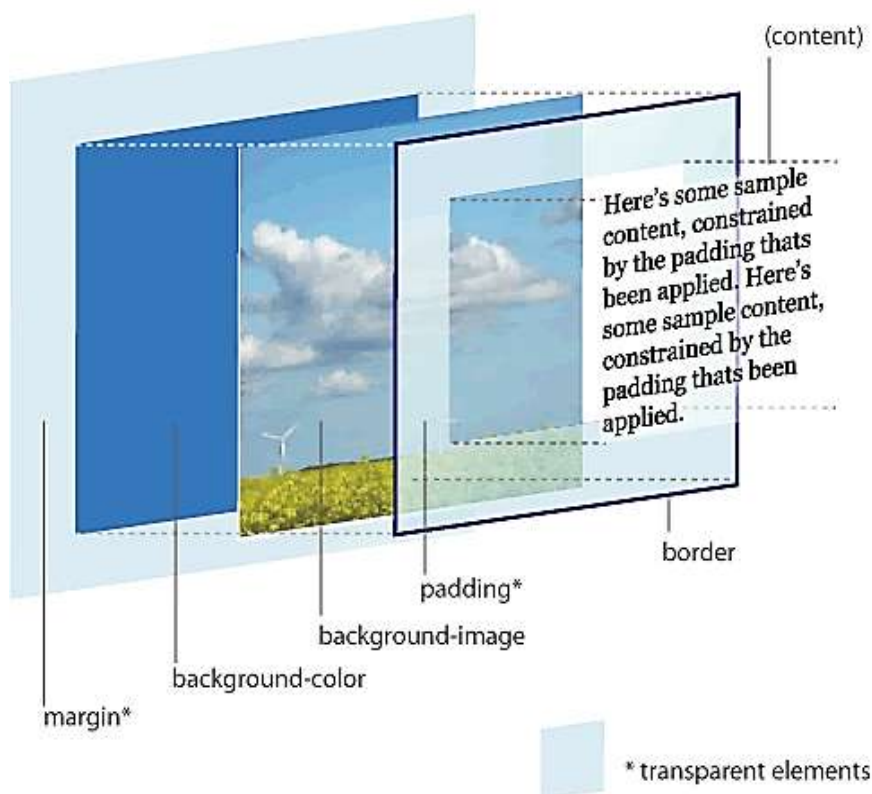
CSS-ში საზღვრების ადვილად აღსაქმელად შემუშავებულია box model-ი, თუ რომელი ელემენტი, რომელ საზღვარს მიეკუთვნება და სად იმყოფება, მისი

საშუალებით შესაძლებელია ელემენტებს შორის სივრცეების მანძილის კონტროლი, თუ სად ვიმყოფებით და რასთან ვმუშაობთ სურათ 2.7-ზე დემონსტრირებულია *box model*.



სურ. 2.7 *box model* დემონსტრირება

როგორც ვხედავთ, *Box model*-ს მოიცავს *Margin*, *border*, *padding* და კონტენტი ანუ ინფორმაცია, რაც წერია, ხოლო რეალობაში ეს გვერდება რომელიც *hicksdesign*-მა წარმოადგინა 3D-ში და ნაჩვენებია, სურათ 2.8-ზე.



სურ 2.8 3D მოდელირება *box model*-ის

რა თქმა უნდა, აქ მოცემულ ელემენტებს, როგორიცაა: *background*, *border*, *padding*-ს და *Content*, ქვემოთ შევხებით.

სხვა სიდიდეები

არსებობს სხვა სიდიდეები CSS-ში მაგალითად, როგორც კუთხე *<angle>*, დრო *<time>*, სიხშირე *<frequency>*, რომელიც გამოყენება სხვადასხვა ფორმის, თუ დახრილობებისათვის, დროითი ინტერვალისათვის.

კუთხეში შედის *deg*(გრადუსი-360° სრული ბრუნია), *Grad*(გრადიანი 400 რადიანი სრული ბრუნია), *rad* (რადიანი სრული ბრუნო-2π, რომელიც თითქმის 6.2832rad), *turn* (ბრუნი-1 არის სრული ბრუნი)

დროით სიდედებში შედის წამი და მილი წამის და ისინი აღინიშნება *s-Seconds* (წამი) და *ms-Milliseconds*(მილიწამები), სადაც 1000 მილიწამია=1 წამი სიხშირე კი განისაზღვრება ჰერცებში (*Hz*) და *kHz*(კილოჰერცებში).

backgrounds

Backgrounds-ანუ უკანა, ფონი რომელიც განსაზღვრავს გვერდის უკანა ფონს, მისი თვისებებია *background-color* (უკანა ფონის ფერი), *background-image* (უკანა ფონი სურათით, *background-repeat* (უკანა ფონის გამეორება), *background-attachment* (უკანა ფონის დანართი) და *background-position* (უკანა ფონის პოზიცია).

ფერები აღიწერება CSS3-ში 4 საშუალებით ესენია:

- ✓ **RGBA**, იგივეა რაც *RGB (red, green, blue)* ხოლო *A*-იგივე *alpha* არხად მოიხსიენებენ, რომელი პასუხისმგებელია *opacity* ფერებზე.
- ✓ **HSL**, არის ფერების, წარმოჩენდება როგორც გაჯერება და სიმსუბუქით, თუ ფერები რამდენად უნდა იყვეს გაჯერებული მისი მაქსიმალური სიდიდეა 100%, ხოლოს სიმსუბუქეც პროცენტულად გამოისახება, სადაც 0% წარმოჩენდება როგორც მუქი ფერი, ხოლო 100% კი თეთრად.
- ✓ **HSLA** იგივეა, რაც **HSL** უბრალოდ დამატებულია აქაც ალფა არხი, რომელიც პასუხისმგებელია გამჭვირვალობაზე ანუ *opacity* ფერებზე.
- ✓ **Opacity** გამჭვირვალობა მინიმუმი სიდიდე 0,0 დან 1-ამდე, 1 არის მისი მაქსიმალური სიდიდე, მაგალითად, თუ დავწერთ *opacity:0.5* ეს ნიშნავს, რომ ფერის გამჭვირვალობა 50% იქნება.

მაგალითი 2.8

მაგალითში ნაჩვენებია ოთხივე სიდიდე, თუ როგორ იწერება:

```
#p1{background-color: rgba(0, 255, 0, 0.3);}
/*მწვანე ფერი 0,3 გამჭვირვალეა */
#p1{background-color: hsl(120, 100%, 75%);} /* ღია მწვანე */
#p1{background-color: hsla(120, 100%, 75%, 0.3);}/* ღია მწვანე 0.3
გამჭირვალე*/
#p1 {background-color:rgb(0,255,0); opacity:0.7;} /* მწვანე ფერი რომელის
გამჭვირვალეა 0,7 ა */
```

თითოეულ *background*-ს აქვს თავისი, თვისებები, რომელიც მოცემულია ცხრილის სახით, მაგრამ უნდა აღინიშნოს, რომ CSS3-ში დამატებულია ელემენტები, როგორცაა:

background-size:	განსაზღვრავს ზომას, ხოლო მისი სიდიდებით განისაზღვრება არის: <i>auto</i>, <i>length</i>, <i>cover</i>, <i>contain</i>, <i>initial</i>, <i>inherit</i>, <i>percentage</i> ➤ auto: სურათის რეალური ზომა; ➤ length: სიგრძე და სიგანის მინიჭება; ➤ cover: სურათი გახდება დიდი, რამდენდაც უკანა ფონი ზომას მისცემს ამის შესაძლებლობას; ➤ contain: მაქსიმალურად გაზრდის ერთნაირად სიგრძე-სიგანეს; ➤ initial: მინიჭებს საწყის მნიშვნელობას ➤ inherit: მემკვიდრეობით გადმოცემული კლასის ან რაიმე სიტყვის აღწერა შესაბამისად ➤ percentage: პროცენტულად გამოხატავს სიდიდეს; მაგალითი 2.9 მოცემული ყოველა ელემენტი
background-origin:	განსაზღვრავს სურათის პოზიციას, ხოლო მისი დამხმარე თვისებების სიდედეებია <i>padding-box</i>, <i>border-box</i>, <i>content-box</i>, <i>initial</i>, <i>inherit</i> (მაგალით 2.10 მოცემული) ➤ padding-box: განთავსებულია მარცხენა ზედა კუთხეში და არ ანიჭებს რაიმე მნიშვნელობას. ➤ border-box: ასევე იწყება საზღვრის მარცხენა კუთხეში ➤ content-box: ანიჭებს მნიშვნელობას საზღვრის მარცხენა ზედა კუთხეში.
background-clip:	განსაზღვრავს ფერებს და არა სურათს, თუ როგორ უნდა გამოჩნდეს, მისი თვისებებია: <i>padding-box</i>, <i>border-box</i>, <i>content-box</i>, <i>initial</i>, <i>inherit</i> მაგალითი 2.11

მაგალითი 2.9

```
<head>
<style>
div
{
    background-image:url( 'ჩასვით თქვენთვის სასურველი ლინკი');
    background-repeat:no-repeat;
    background-size:auto;
}
</style>
</head>
```

ხოლო *background-size:auto*; -ში *auto*-ს მაგივრად მის ყველა თვისება ჩასვით, ბრაუზერშია ნახავთ სხვაობას.

მაგალითი 2.9

```
<head>
<style>
div {
    padding:25px;
    border:2px solid #000000;
    background-color:red;
    background-clip:border-box;
```

```

}
</style>
</head>

```

მაგალითი 2.10

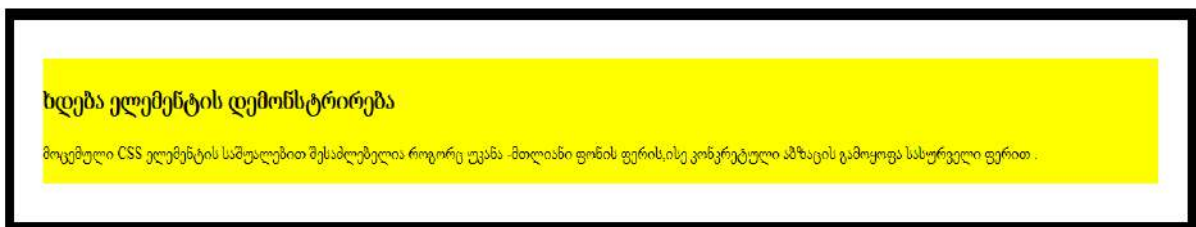
```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS ელემენტები </title>
<style>
#მაგალითი
{
    border: 10px solid black;
    padding: 33px;
    background: red;
    background-clip: content-box;
}
</style>
</head>
<body>
<p>background-clip-ს დემონსტრირება </p>
<div id="მაგალითი">
<h2>ხდება ელემენტის დემონსტრირება</h2>
<p>მოცემული CSS ელემენტის საშუალებით შესაძლებელია როგორც უკანა -
მთლიანი ფონის ფერის, ისე კონკრეტული აბზაცის გამოყოფა სასურველი ფერით
.</p>
</div>
</body>
</html>

```

შედეგი

background-clip-ს დემონსტრირება



მაგალითი 2.11

```

<!DOCTYPE html>
<html>
<head>
<style>

```



```

#მაგალითი
{
    border: 10px solid black;
    padding: 35px;
    background: url(სურათის ლინკი);
    background-repeat: no-repeat;
}</style>
</head>
<body>
<div id="მაგალითი">
<h2> სასურველი სურათი </h2>
<p> უკანა ფონის დემონსტრირება .</p>
</div>
</body>
</html>

```

background-repeat

უკანა ფონის განმეორება შესაძლებელია როგორც x , ასევე y კოორდინატი სიბრტყისათვის, წარმოიდგინეთ სურათი, რომელიც ფონად აქვს და გვინდა კონკრეტული მხარეს გამეორდეს, მიუთითებს x ან y , ხოლო თუ ორივე მხარეს მაშინ $x+y$ პრაქტიკაში ხდება მსგავსი ფორმულირება.

მისი სიდიდეებია:

repeat- გაიმეორებს უკანა ფონს ორივე მიმართულებით;
repeat-x უკანა ფონი გაიზრდება ჰორიზონტალურად;
repeat-y უკანა ფონს გაიმეორებს ვერტიკალურად;
no-repeat უკანა ფონის სურათი არ გამეორდება;

მაგალითი 2.12

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
body {
    background-image: url("http://scripts.ge/wigni/background-repeat.png ");
    background-repeat: repeat-y;
}
</style>
</head>
<body>
<p> შეცვალეთ სიდიდეები და შესაბამისადაც შეიცვლება</p>
</body>
</html>

```

background-attachment

უკანა ფონის კონტექსტის შეიძლება გადაადგილდეს ან უცვლელედ დარჩეს, მისი მიზანია, რომ განისაზღვროს ყოველივე მისი სინტაქსის შემადგენელი სიდიდეები: *Scroll, fixed, local, initial* და *inherit*, თვალსაჩინოებისათვის მაგალითში შეუცვალეთ სიდიდეები.

```
body
{
background-image:url("http://scripts.ge/wigni/background-repeat.png");
background-repeat: no-repeat;
background-attachment: fixed;
}
```

background-position

განსაზღვრავს უკანა ფონის პოზიციას, თუ სად უნდა განთავსდეს უკანა ფონი, მისი შემადგენელი სინტაქსის სიდიდეებია.

<i>left top</i>	განსაზღვრავს პოზიციებს
<i>left center</i>	
<i>left bottom</i>	
<i>right top</i>	
<i>right center</i>	
<i>right bottom</i>	
<i>center top</i>	
<i>center center</i>	
<i>center bottom</i>	
<i>x% y%</i>	
<i>xpos ypos</i>	განსაზღვრება პიქსელებში

მაგალითი 2.13

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-image:
url("რაიმეღინკრასკით");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: 40% 40%;
}
</style>
</head>
<body>
</body>
</html>
```

შეუცვალეთ სხვადასხვა სიდიდე ალწერთ *background-position: 40% 40%*; მაგივრად *background-position: right top*; ან *background-position: 20px 15px*; გავუწერეთ.

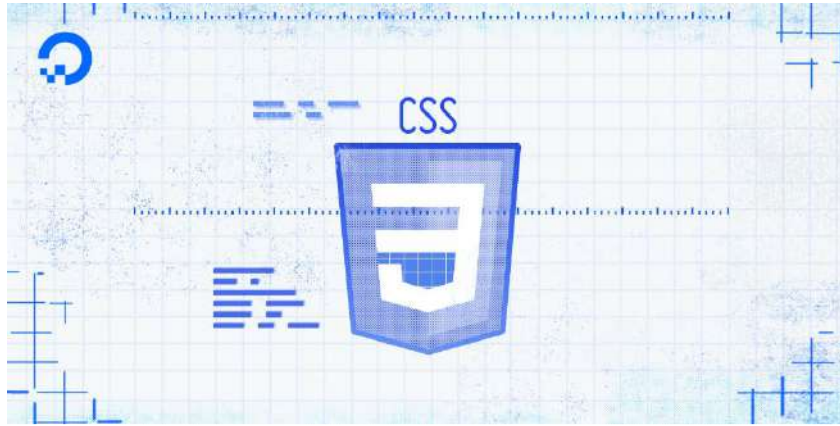
background-blend-mode

განსაზღვრავს უკანა ფონის სიმკვეთრეს, თუ როგორ წარმოჩნდეს მაგალითად, სურათი სინტაქსში მისი შემადგენელი ელემენტებია: *normal*, *multiply*, *screen*, *overlay*, *darken*, *lighten*, *color-dodge*, *saturation*, *color*, *luminosity*.

მაგალითი 2.14

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
background-color: green;
}
Div
{
width: 200px;
height: 200px;
background-size: 200px 200px;
background-repeat: no-repeat;
background-image: linear-gradient(to right, black 0%, white 100%),
url("სასურველი ბმული ");
background-blend-mode: lighten;
}
</style>
</head>
<body>
<div>
</div>
</body>
</html>
```

ბრაუზერების სპეციფიკა



ბრაუზერს კოდი მარტივად წაეკითხა, ბრაუზერის მწარმოებელმა CSS-ში მოიფიქრეს და შეიმუშავეს ერთგვარი ინდექსები, რითაც უფრო მარტივად აღიქვამდა კოდს, ხოლო პროგრამისტები რომლებიც ყოველივე ამას წერენ, უწევთ სხვადასხვა ბრაუზერისათვის ცალკეულად გაწერა, შეიძლება ასევე, ზოგადად CSS სკრიპტის აღწერა ყველა ბრაუზერისათვის, რომელსაც შემდეგ ყველა ბრაუზერი დაარენდერებს, მაგრამ უფრო დიდ დროს ანდომებს, ბრაუზერი გვერდის ჩატვირთვისას, ასე რომ პროგრამისტს უწევს დაწეროს კოდი ყველა ბრაუზერის შესაბამისი ინდექსებით, პროგრამისტისათვის მარტივი დასაწერი რომ ყოფილიყო, ძირითად CSS ელემენტებს დაუმატეს სხვადასხვა ბრაუზერის ინდექსი, მაგალითად, ინდექსები განისაზღვრება ორ ტიპად:



-მწარმოებელი-*id*-სახელი

ან



მწარმოებელი*id*_სახელი

ცხრილი 2.3 მოცემულია ბრაუზერების, პროგრამებისა და მოწყობილობების აღქმა

პრეფიქსი	შინაარსი
-atsc-	<i>Advanced Television standards Committee</i>
-ms-	<i>Microsoft</i>
mso-	<i>Microsoft office</i>
-moz-	<i>Mozilla Foundation</i>
-o-	<i>Opera</i>
-webkit-	<i>Safari, Google chrome (და სხვა WebKit-based browsers)</i>
-khtml-	<i>Konqueror browser</i>

დავუშვათ, თუ გვინდა რომ მარცხენა საზღვრის აღწერა (*border-left-colors*) რომელიც ყველა ბრაუზერი არენდერებს და ხოლო თუ გვსურს დავუშვათ *Mozilla*-ს დაარენდოს იგივე სკრიპტი მაშინ -*moz-border-left-colors* დავწერთ, მაგალითი 2.15-ში ნაჩვენებია სკრიპტი, რომელიც ნებისმიერი ბრაუზერიდან მარცხენა

კიდე იქნება წითლად, ხოლო Mozilla-ში კი მწვანედ, დეტალურად *border*-ებს მოგვინებით შევხებით:

მაგალითი 2.15

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
p {
    border-style: solid;
    border-left-width: 25px;
    border-left-color:red;
    -moz-border-left-colors:green;
}
</style>
</head>
<body>
<p>
<b>აქ:</b> მოცემულია ტექსტი </p>
</body>
</html>
```

შეინახეთ და ერთი და იგივე ფაილი სხვადასხვა ბრაუზერში გახსენით, მათ შორის mozilla-შიც და დაინახავთ სხვაობას, უნდა აღინიშნოს, რომ W3C მხარდაჭერა არ აქვს ზემოთ ნახსენებ ელემენტებს.

border

ნებისმიერი ელემენტი, რომელიც აღიწერება, მას აქვს საზღვრები, სურათი, ტექსტის ფორმა, ავტორიზაცია და ა.შ კიდეები ფორმდება *border* ელემენტის საშუალებით და იგი *box model*-ის შემადგენილი ნაწილია, შეგიძლიათ იხილოთ სურ2.8 ზე.

როგორც ყველა საზღვარს, *box model*-ი განისაზღვრება მხარებით, მარცხენა, მარჯვენა, ზედა და ქვედა, თანამიმდევრობით იქნება, შემოკლებით სამ თვისებას გამოვყოფთ, ესენია:

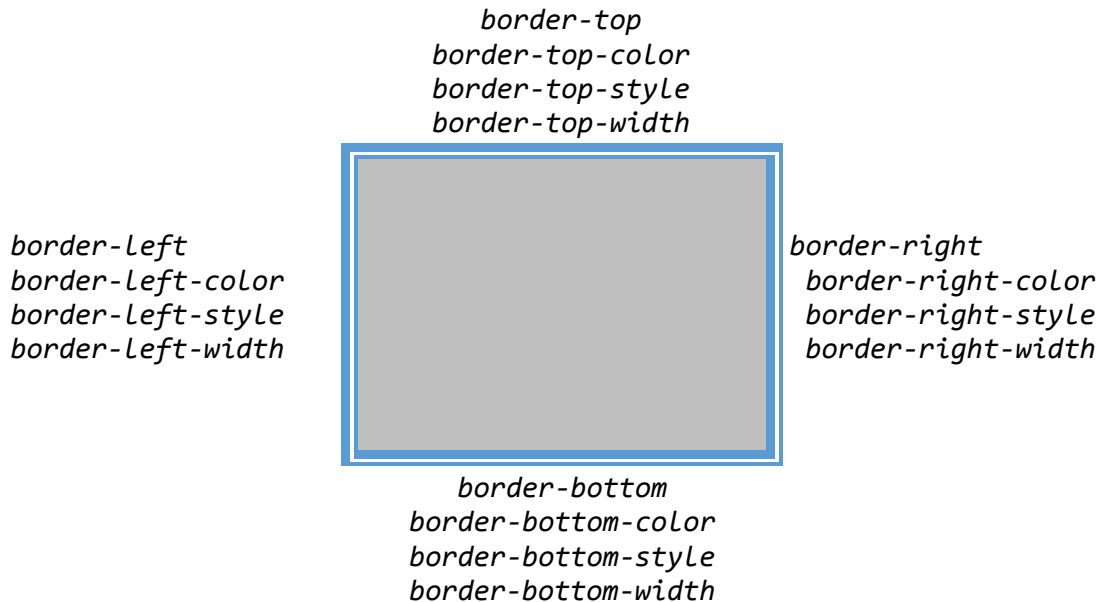
border-color

border-style

border-width

border-width, *border-style* (აუცილებლად უნდა მიეთითოს) და *border-color*, უნდა აღინიშნოს, რომ თითოეული გვერდის, მაგალითად მარცხენა საზღვრის აღწერისას რა თვისებები და პარამეტრების ექნება, იგივე ელემენტები ექნება მარჯვენა, ზედა თუ ქვედა საზღვრები.

Border-მოიცავს ყველა ელემენტს, ხოლო როგორც აღვნიშნეთ, რა პარამეტრები არის *border-bottom*-თვის იგივეა *border-left* და ა.შ, ფერი, სტილი და სისქე.



განვიხილოთ მარცხენა საზღვარი და შესაბამისად რა პირობებიც იქნება მარცხენა მხარისათვის ანალოგიურად იქნება დანარჩენებისათვის, დასაწყისათვის გავეცნოთ ზოგად ელემენტებს.

border-width აღწერს სისქეს, როგორც შესაძლებელია პიქსელებში ასევე წარმოვადგინოთ სიტყვიერადაც, *Medium* (საშუალო), *thin* (თხელი), *thick* (სქელი), და *Length* (გრძელი).

border-color როგორც ცალკეულად ერთი ფერისა ასევე შეგვიძლია ოთხივე საზღვარს სხვადასხვა ფერი მივანიჭოთ, თანამიმდევრობა კი საათის ისრის მოძრაობის მიმართულებითა: *top* (ზემოთ), *right* (მარჯვნივ), *bottom* (ქვემოთ) და *left* (მარცხნივ), რომელიც თავის მხრივ *color* აღწერს ფერს, ხოლო *transparent* კი გამჭვირვლე იქნება.

border-style რაც შეეხება სტილისტკას, მაში მოიაზრება ის, რომ თვითონ საზღვარი როგორიც უნდა იყოს: უწყვეტილი, წვეტილი და ა.შ მისი სინტაქსი შემადგენელი თვისებებია: *none*, *hidden*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset* *initial* და *inherit*.

მაგალითი 2.16

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
p.yvela {
border-style: dotted solid dashed double;}
p.samive{
border-style: dotted solid dashed;}
p.mxolodori {
border-style: dotted solid;
}
p.yvelaerti {
```

```
border-style: dotted;
}
</style>
</head>
<body>
<p class="yvela">მოცემულია ოთხი სხვადასხვა საზღვარი.</p>
<p class="samive">მოცემულია სამი სხვადასხვა საზღვარი.</p>
<p class="mxolodori">მოცემულია მხოლოდ ორი სხვადასხვა საზღვარი</p>
<p class="yvelaerti">როცა ავტომატურად ყველა გვერდს ერთი საზღვარი აქვს </p>
</body>
</html>
```

შედეგი

მოცემულია ოთხი სხვადასხვა საზღვარი.
მოცემულია სამი სხვადასხვა საზღვარი.
მოცემულია მხოლოდ ორი სხვადასხვა საზღვარი
როცა ავტომატურად ყველა გვერდს ერთი საზღვარი აქვს

რაც შეეხება ყველას ვიზუალიზაცია, თუ რა სხვაობა და რომელი თვისება როგორ წარმოჩნდება, შეგვიძლიათ სკრიპტში მოკლე ცვლილება განვახორციელოთ, მაგალით 2.16 გამოვეყენოთ პირველი კლასი, (`<p class="yvela">`) და ორი საზღვარი სხვადასხვა ფერში წარმოვაჩინოთ, ხოლო დანარჩენი კლასები უბრალოდ წაშალოთ, მაშინ სკრიპტი ჩაიწერება შემდეგნაირად:

მაგალითი 2.17

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
p.yvela {
border-top: 4px dotted #000000;
border-bottom: 1px solid #ff0000;
}
</style>
</head>
<body>
<p class="yvela">მოცემულია ოთხივე სხვადასხვა საზღვარი.</p>
</body>
</html>
```

შედეგი:

.....

მოცემულია ორი სხვადასხვა

როგორც სკრიპტიდან ჩანს ზედა მხარე არის წყვეტილი შავი, ხოლო ქვედა, სოლიდური და ისიც წითლად, ხოლო სურათი 2.10 ზე ნაჩვენებია ყველა საზღვრის ტიპები.



სურ 2.10 ნაჩვენებია ყველა საზღვრის ტიპები

border-radius

საზღვრების მომრგვალება, და გარკვეული ფორმების მიცემა, დაიზანიტაც და სტილისტურადაც ლამაზი და დახვეწილი ხდება, საიტი, როგორც CSS ყველა ელემენტი აქვს თავის თვისებები და სიდიდეები, *border-radius* ჩაიწერება როგორც:

```
{
border: 2px solid;
border-radius: 15px;
}
```

რაც შეეხება მისი სინტაქსს მოიცავს *length*, რომელიც *default*-ზე ყოველთვის უდრის 0 და %, რომელიც განისაზღვრება პროცენტულად.

border-radius: 15px;; ეს იმავს,რაც:

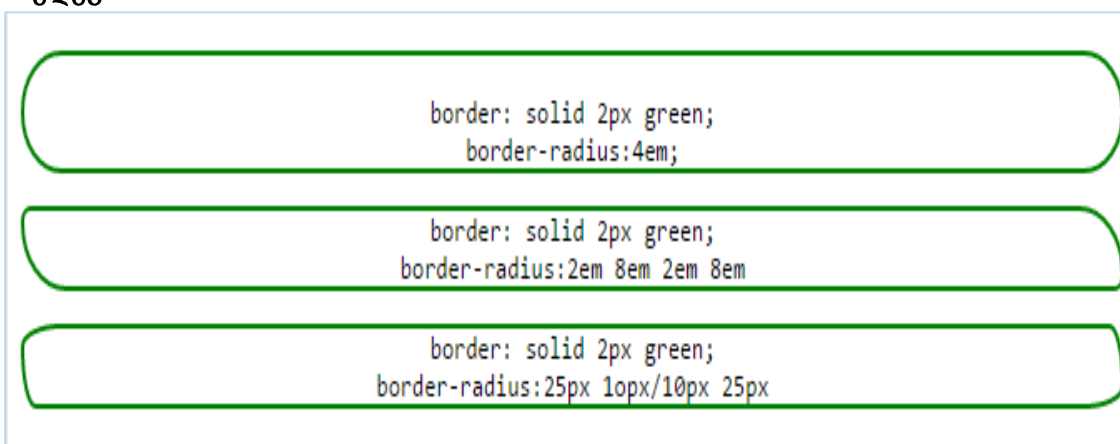
```
{
border-top-left-radius:15px;
border-top-right-radius:15px;
border-bottom-right-radius:15px;
border-bottom-left-radius:15px;
}
```

როგორც ვხედავთ, ოთხივე კუთხე ანიჭებს ერთსა და იმავე მნიშვნელობას, რა თქმა უნდა აქაც შეგვიძლია, რომ მოვახდინოთ ჩასწორება და სხვადასხვა კუთხე სხვადასხვაგვარად მომრგვალდება განვიხილოთ რამდენიმე ვარიანტი მაგალითი 2.18

მაგალითი 2.18

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>საზღვრები</title>
</head>
<body>
<br>
<br>
<pre style="border: solid 2px green;
border-radius:4em;
text-align:center;">
border: solid 2px green;
border-radius:4em;
</pre>
<pre style="border: solid 2px green;
border-radius:2em 8em 2em 8em;
text-align:center;">
border: solid 2px green;
border-radius:2em 8em 2em 8em
</pre>
<pre style="border: solid 2px green;
border-radius:25px 10px/10px 25px;
text-align:center;">
border: solid 2px green;
border-radius:25px 10px/10px 25px
</pre>
</body>
</html>
```

შედეგი



border-image

CSS3-ში სურათები ჩამატებული საზღვრებში უკვე სურათები ანუ ფრაგმენტი სურათებით შესაძლებელია სასურველი ფორმის საზღვრების შექმნათ, როგორც სხვა ელემენტებს, მასაც აქვს *-webkit-*, *-moz-*, და *-o-* ბრაუზერის პრეფიქსის მხარდაჭერა. აქაც როგორც ზემოთ ვთქვით, როგორც ცალკეულად კუთხეები და გვერდების განსაზღვრა შეიძლება ასევე შესაძლებელია ყველა საზღვარის ერთიანად აღწერა

მაგალითად:

```
img
{
  border:1px solid #b5cadcd;
}
```

<i>border-image</i>	საზღვრების სურათი რომლის სინტაქსისი თვისებებია: <i>source</i> , <i>slice</i> , <i>width</i> , <i>outset</i> , <i>repeat</i> , თუ დააკვირდებით, სხვა ელემენტების მსგავსი თვისებები აქვს
<i>border-image-outset</i>	რაოდენობა სურათისა და საზომი ერთეული, <i>length</i> და <i>number</i>
<i>border-image-repeat</i>	იმეორებს სურათს <i>stretch</i> (საზღვარი შევსებულია) <i>repeat</i> (უბრალოდ იმეორებს) და <i>round</i> უბრალოდ ამრგვალებს.
<i>border-image-slice</i>	სურათის ჭრილების სიდიდე განისაზღვრება რიცხვებით, % (პროცენტით) და <i>fill</i> (შევსებით)
<i>border-image-source</i>	განსაზღვრავს სურათს, როგორც საზღვარს, ელემენტის გარშემო მისი ელემენტის თვისებებია: <i>none</i> (სურათი არ გამოიყენება) და <i>image</i> (სურათი გამოიყენება როგორც საზღვრად)
<i>border-image-width</i>	სურათი სიგანე მისი საზომი, განისაზღვრება შემდეგი თვისებით <i>number</i> (რიცხვით), % (პროცენტულად) და <i>auto</i> .

უნდა აღინიშნოს, რომ ყველა ელემენტის თვისება შეიცავს *initial* და *inherit*, რაც შეეხება ქვემოთ მოყვანილ მაგალითის 2.19, რომელშიც დემონსტრირებულია ყველა ელემენტი:

მაგალითი 2.19

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
#საზღვარი
{
  border: 15px solid transparent;
  padding: 15px;
  border-image: url(http://scripts.ge/wigni/border.png);
  border-image-slice: 30;
```

```

border-image-repeat: round;
border-image-outset: 2px;
}
</style>
</head>
<body>
<p id="საზღვარი">border ყველა ელემენტის ჩვენება </p>
</body>
</html>

```

შედეგი:



Border-collapse

მოცემული ელემენტი გამოყენება ცხრილის სვეტებთან მიმართებით მისი სინაქსის შემადგენელი ნაწილებია: *separate*, *collapse*, *initial* და *inherit*

Table

```

{
border-collapse: collapse;
}

```

collapse ცხრილში საზღვრები და სივრცე უძრავია, ხოლო *separate* დროს კი იყოფა.

Box-shadow

CSS-ში ძალიან პოპულარულია ჩრდილების დასმა და გაფორმება, ჩრდილი როგორც კონკრეტულ მხარეს ისე ოთხივე მხარეს შეგვიძლია დავსვათ, ასევე ძალიან პოპულარულია *text-shadow*, რომელიც მონათესავე ელემენტია.

<i>none</i>	არანაირი ჩრდილი არ იქნება;
<i>h-shadow</i>	სიმაღლის მიმართ იქნება ჩრდილი;
<i>v-shadow</i>	ვერტიკალურად იქნება დასმული ჩრდილი;
<i>blur</i>	ბუნდოვნად დასვამს ჩრდილს;
<i>spread</i>	ჩრდილის გავრცელების არეალი;
<i>color</i>	ფერის მინიჭება;
<i>inset</i>	შიგ სხვა ფერის ჩასმა, ხოლო გამომავალი სხვა ფერი იქნება;

მაგალითი 2.20

```

<!DOCTYPE html>
<html>
<head>
<style>
div
{

```

```

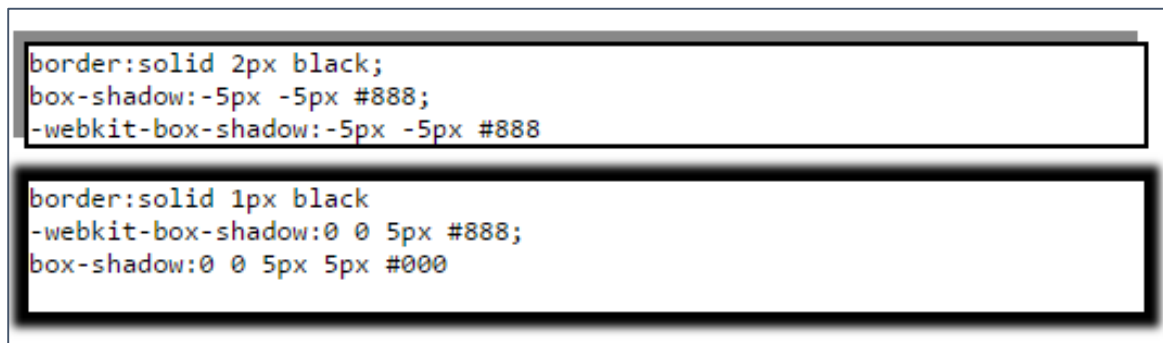
width: 300px;
height: 300px;
background-color: grey;
box-shadow: 10px 10px 5px green;
}
</style>
</head>
<body>
<div></div>
</body>
</html>

```

შედეგი



ასევე ნაჩვენებია ყველა მხარის ჩრდილი



text-shadow

CSS3-ტექსტის ეფექტების დადება, *text-shadow* ხდება, ხოლო მისი მონათესავე ელემენტია *box-shadow*, რომელთან ერთი და იგივე თვისება აქვთ, აქედან გამომდინარე, მის თვისებებს აღარ განვიხილავთ და მაგალითის საფუძველზე ვნახოთ შედეგი, ხოლო სკრიპტში თუ არ არის აღწერილი ფონტის ფერი, მაშინ იქნება სტანდარტულ შავ ფერზე, ხოლო ეფექტი იქნება იმ ფერში, რასაც მივუთითებთ, ასევე შეგვიძლია მივუთითოთ ფერი მაგალითი 2.21-ში, ფონტის ფერი ლურჯია, ხოლო უკანა, ეფექტი კი წითელია:

მაგალითი 2.21

```
<!DOCTYPE html>
```

```

<html>
<head>
<meta charset="utf-8">
<style>
    h1 {
        color: blue;
        text-shadow: 0 0 3px #FF0000;
    }
</style>
</head>
<body>
    <h1>ტექსტის ეფექტი</h1>
</body>
</html>

```

შედეგი

ტექსტის ეფექტი

ხოლო CSS სკრიპტში, თუ წავშლით *color:blue*; ხაზს, შესაბამისად სტანდარტულად შავი ფერი იქნება და ეფექტი კი უცვლელად წითლად, CSS-ში ტექსტებს ჩვენ კიდეც დავუბრუნდებით.

Outline

Border-ელემენტის გარეშემო კიდევ შესაძლებელია ერთგვარი საზღვრის ანუ გარეხაზის შემოვლება, მინიჭება ფერები, სიგანი და სტილის, მოცემული ელემენტი ხასიათდება:

<i>outline-color</i>	ფერით, რომელიც მოიცავ, <i>invert</i> და <i>color</i> თვისებებს
<i>outline-style</i>	სტილით, ყველა იმ სტილით, რაც <i>Border</i> -ს აქვს და მოცემულია სურათ 2.10-ზე
<i>outline-width</i>	სიგანით, რომელიც ასევე ხასიათდება <i>border</i> -ის თვისებებით: <i>Medium</i> , <i>thin</i> <i>thick</i> და <i>Length</i>

მოცემულ ოთხივე ელემენტს აქვს *initial* და *inherit*

მაგალითი 2.22

```

<!DOCTYPE html>
<html>
<head>
<style>
p {
    border: 1px solid red;
    outline-style: dotted;
    outline-color: #353535;
}
</style>

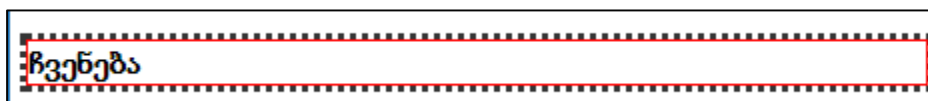
```

```

</head>
<body>
<p><b>ჩვენება </p> </body>
</html>

```

შედეგი



Overflow

CSS2-ში შემუშავებული ელემენტია, რომელიც დაიხვეწა CSS3-ში და მისი საშუალებითაც შესაძლებელია, კონტენტის ისეთი თვისებების მინიჭება როგორიცაა, დამალვა, დაყოფა და ა.შ, *Overflow* თვისებების სიდიდეებია *visible*, *scroll*, *hidden*, და *auto*. შემდეგ შემუშავდა, როგორც ვერტიკალური ისევე ჰორიზონტალური მიმართულებით შემდეგი სიდიდეები:

```

overflow-x:hidden;
overflow-x:visible;
overflow-y:auto;
overflow-y:scroll;

```

ახლა რაც ვისაუბრეთ მაგალითის სახით წარმოვაჩინოთ და დავყოთ, მაგალითი 2.23 ჩვენება მოცემულია, როგორც სკრიპტი, ისე შედეგიც.

მაგალითი 2.23

```

<!DOCTYPE html>
<html>
<head>
<title>ტექსტის დაყოფა</title>
<meta charset="utf-8">
<style>
.columns
{
text-align:justify;
font-size:14px;
-moz-column-count:3;
-moz-column-gap :1em;
-moz-column-rule :1px solid black;
-webkit-column-count:3;
-webkit-column-gap :1em;
-webkit-column-rule :1px solid black;
column-count :3;
column-gap :1em;
column-rule :1px solid black;
}

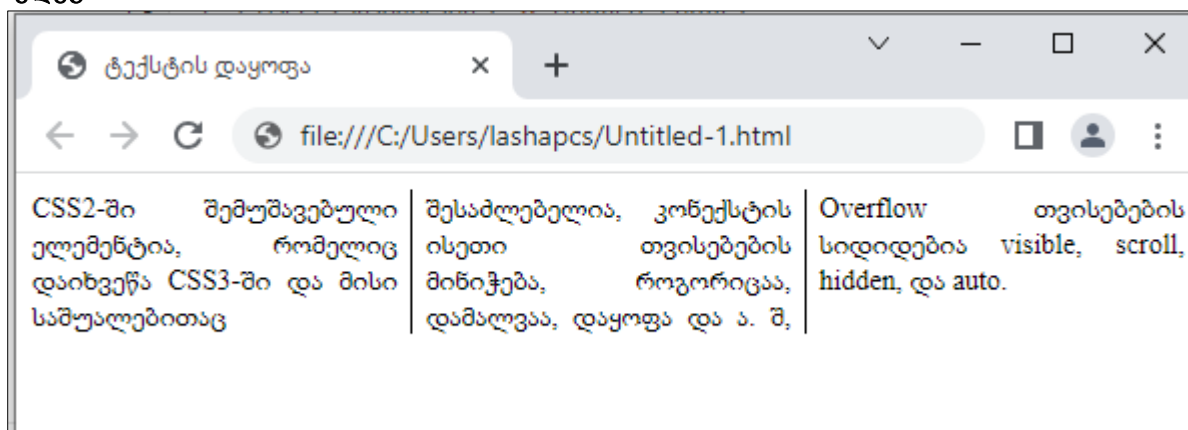
```

```

</style>
</head>
<body>
<div class='columns'>
    CSS2-ში შემუშავებული ელემენტია, რომელიც დაიხვეწა CSS3-ში და მისი
    საშუალებითაც შესაძლებელია, კონექსტის ისეთი თვისებების მინიჭება, როგორიცაა,
    დამალვა, დაყოფა და ა.შ, Overflow თვისებების სიდიდებია visible, scroll,
    hidden, და auto.
</div>
</body>
</html>

```

შედეგი



column

ყურადღება მიაქციეთ მაგალითი 2.23 ის, სვეტებს, რომელიც დაყოფილია, ასევე ისეთ, ელემენტებს, როგორიცაა:

column-count რომელიც, განსაზღვრავს სვეტების რაოდენობას;

column-gap: სვეტებს შორის სიგრძეს;

column-rule: თუ როგორი უნდა იყოს საზღვრები;

column-span მისი მიზანია სვეტებ, შორის ინტერვალი, სტანდარტულად 1-ია, თუ ALL მითითებული, მაშინ ყველა სვეტი ერთმანეთს გადაკვეთს;

column-width: განსაზღვრავს თითოეული სვეტის სიგანეს, მისი თვისებებია *auto* და *length*-გარკვეული სიგრძე.

column-rule ელემენტის თვისებები განისაზღვრება სამი სხვა ელემენტის თვისებით: *column-rule-color*, *column-rule-style* და *column-rule-width* სხვაგვარად რომ ვთქვათ, განისაზღვრება ფერით, სტილითა და სიგანით.

 *column-rule-width* თავის მხრივ ხასიათდება ისეთი თვისებით, როგორიცაა: *thin*, *thick*, *medium* და სიგრძით, რომელსაც რიცხვით ვწერთ მაგალით 2.23 ტექსტი უცვლელი დავტოვოთ და მხოლოდ CSS სკრიპტი ჩავანაცვლოთ მოვათავსოთ სამივე თვისება სკრიპტში.

 *column-rule-style* ხასიათდება ყველა იმ სტილით, რასაც უკვე ზემოთ გავეცანით იხილეთ სურ. 2.10.

 *column-rule-color* ასევე ჩვენთვის ნაცნობი ფერის ტიპებს.

როგორც ვხედავთ, ფერი სტილი და სიგანე უცვლელი სიდიდეებია და რა ელემენტში არ უნდა გამოიყენებოდეს, ყველგან უცვლელი რჩება, რაც შეეხება ელემენტის მაგალითს 2.24-ში ჩაიწერება:

მაგალითი 2.24

```
<!DOCTYPE html>
<html>
<head>
<title>ტექსტის დაყოფა</title>
<meta charset="utf-8">
<style>
.columns
{
    column-count:2;
    -webkit-column-count:2;
    -moz-column-count:2;

    column-rule-color:#0000ff;
    -webkit-column-rule-color:#0000ff;
    -moz-column-rule-color: #0000ff;

    column-rule-style: outset;
    -webkit-column-rule-style:outset;
    -moz-column-rule-style:outset;

    column-rule-width: 6px;
    -webkit-column-rule-width:6px;
    -moz-column-rule-width:6px;
}
</style>
</head>
<body>
<div class='columns'>
    CSS2-ში შემუშავებული ელემენტია, რომელიც დაიხვეწა CSS3-ში და მისი
    საშუალებითაც შესაძლებელია, კონტენტის ისეთი თვისებების მინიჭება, როგორიცაა,
    დამალვა, დაყოფა და ა.შ, Overflow თვისებების სიდიდეები: visible, scroll,
    hidden, და auto.
</div>
</body>
</html>
```

შედეგი

CSS2-ში შემუშავებული ელემენტია, რომელიც დაიხვეწა CSS3-ში და მისი საშუალებითაც შესაძლებელია, კონტენტის ისეთი თვისებების

მინიჭება, როგორიცაა, დამალვა, დაყოფა და ა.შ, Overflow თვისებების სიდიდეები: visible, scroll, hidden, და auto.

გაფართოებები

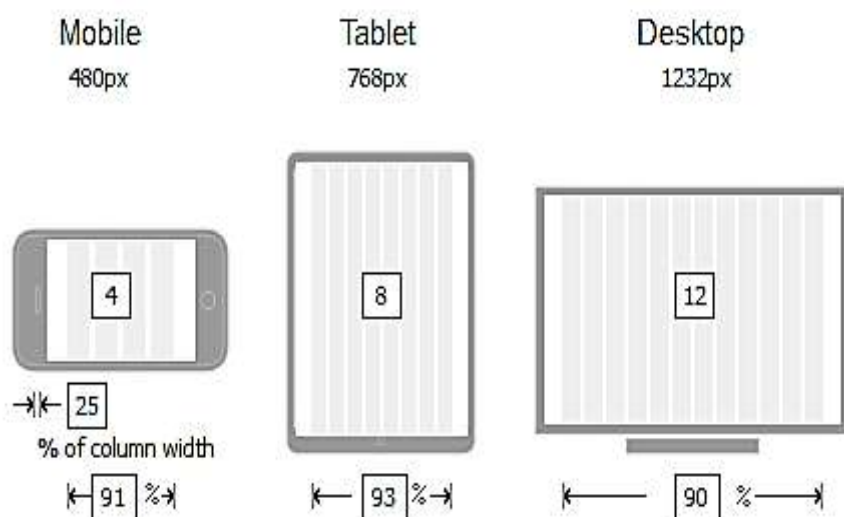
გაფართოების ისეთი ელემენტები, როგორიცაა *width* და *height*, რა თქმა უნდა ჩვენთვის უკვე ცნობილია, მაგრამ CSS ში არის დამატებული ელემენტები, ისეთი როგორცაა:

<i>max-height</i>	მაქსიმალური სიმაღლე, რომელიც შეიძლება გაფართოვდეს და მას აქვს <i>height</i> ყველა თვისება, როგორიცაა: none, სიგრძე (px, cm და ა. შ) და %-რომელიც განსაზღვრავს მაქსიმალურ სიმაღლეს, რა თქმა უნდა, ეს ელემენტი არ მოიცავს <i>border</i> , <i>panding</i> და <i>margin</i> -ს
<i>min-height</i>	მინიმალური სიმაღლე, რა თქმა უნდა, თვისებებში 0 იქნება ხოლო დანარჩენი ყველაფერი იგივე რჩება, რაც <i>max-height</i> -ს აქვს
<i>max-width</i>	მაქსიმალური სიგანე ანალოგიური თვისებები აქვს, რაც <i>max-height</i> -ს
<i>min-width</i>	იგივე თვისებები აქვს რაც <i>max-height</i> -ს, მინიმალური სიგანე იქნება 0.

ყველა ამ ელემენტის დანიშნულებაა მაქსიმალური და მინიმალური საზღვრების დაწესება, და ეს დაწესება პირველ რიგში სხვადასხვა ზომის კომპიუტერების ეკრანის ან სხვა მოწყობილობის შემთხვევაში საიტით წარმოჩენა, ეს ყოველივე შემუშავებულია იმისათვის რომ საიტი იყოს უფრო მეტად მოქნილი, მაგალითად;

```
div
{
min-width : 200px;
max-width : 500px;
padding : 5px;
border : 1px solid #000000;
}
```

☞ 200px ნაკლები რეზოლუციის ეკრანის შემთხვევაში გვერდი ჩაიტვირთება ამ მინიმალური გაფართოებით, სურათ 2.11-ში მოცემულია ზოგადად აღებული რეზოლუცია პიქსელებში.



სურ 2.11

align-content

მოქნილი ჩანართების კონტეინერი, სადაც არ გამოყენება ყველა თავისუფალი სივრცე, მისი სინტაქსია.

<i>stretch</i>	გაწელილია რომელიც მოიცავს მთელ კონტეინერს და ავსებს.
<i>center</i>	ჩანართები მდებარეობს ცენტრში;
<i>flex-start</i>	ჩანართები მდებარეობს კონტეინერის დასაწყისში;
<i>flex-end</i>	ჩანართები მდებარეობს კონტეინერის დასასრულს;
<i>space-between</i>	ჩანართების სივრცე მდებარეობს ხაზებს შორის;
<i>space-around</i>	ჩანართების სივრცე მდებარეობს ხაზამდე, ხაზებ შორის და ხაზების შემდეგ.

align-items

სპეციფიკა ამ ელემენტის ისაა, რომ მისი მითითების შემდეგ საწყისი პოზიცია ჩანართებშია მოთავსებული და თავსებადია კონტეინერის მიმართ, იგი მოიცავს *stretch*, *center*, *flex-start* და *flex-end* თვისებებს და ასევე აქვს *baseline* თვისება, რომელიც ჩანართებს კონტეინერის ძირითად ხაზე გამოიტანს.

flex

flex თვისების სპეციფიკა ისაა, რომ განსაზღვრავს ჩანართის სიგრძეს, რომელიც უკავშირდება ყველა დანარჩენ მოქნილ ჩანართს შიგ იმავე კონტეინერს, *flex* ელემენტის სინტაქსის თვისებებია:

<i>grow</i>	ციფრი, თუ რაოდენობრივად გვიჩვენებს, რამდენად გაიზრდება ყველა დანარჩენი მოქნილი ჩანართის მიმართებით.
<i>shrink</i>	ციფრი, თუ რაოდენობრივად გვიჩვენებს, რამდენად მცირდება ყველა დანარჩენ მოქნილ, ჩანართის მიმართებით.
<i>basis</i>	ჩანართის სიგრძე, რომელიც განისაზღვრება: <i>auto</i> , <i>inherit</i> და ციფრებით %, <i>px</i> , <i>em</i> ან სხვა სიგრძის ერთეულები.

შმაგალითი 2.25-ში ძირითადად ნაჩვენებია, კოდი როგორ იწერება, ფერების სახელწოდებებით გამოვყოთ თითოეული ჩანართი, მაგრამ ეს არ ნიშნავს თითოეული ჩანართი იმ ფერის იქნება.

მაგალითი 2.25

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
.კონტეინერი
{
    display: flex;
    display: -webkit-flex;
    border: solid 1px green;
}
.ჩანართი
```

```

{
margin:5px;
height:52px;
padding:8px;
border-radius:4px;
background-color:#3a87ad;
color:white;

}

.მწვანე {
    flex:1;
    -webkit-flex:1;
}

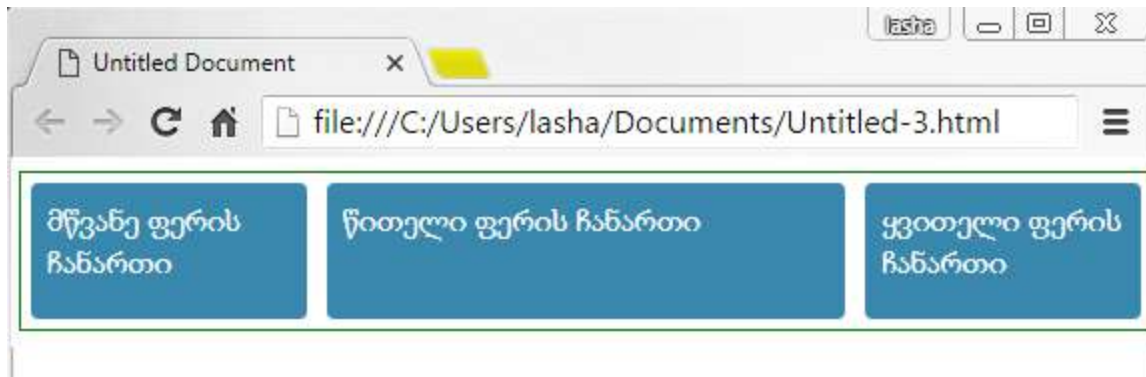
.წითელი {
    flex:2;
    -webkit-flex:2;
}

.ყვითელი
{
flex:1;
-webkit-flex:1;
}

</style>
<title>flex დემონსტრირება</title>
</head>
<body>
<div class="კონტეინერი">
<div class="მწვანე ფერის ჩანართი"> მწვანე ჩანართი</div>
<div class="წითელი ფერის ჩანართი"> წითელი ჩანართი</div>
<div class="ყვითელი ფერის ჩანართი"> ყვითელი ჩანართი</div>
</div>
</body>
</html>

```

შედეგი



flex-basis

flex-ელემენტის მონათესავე ელემენტები, რომლებიც მისი თავსებადია, თავისი სიგრძით, სხვა მოქნილ ჩანართებთან.

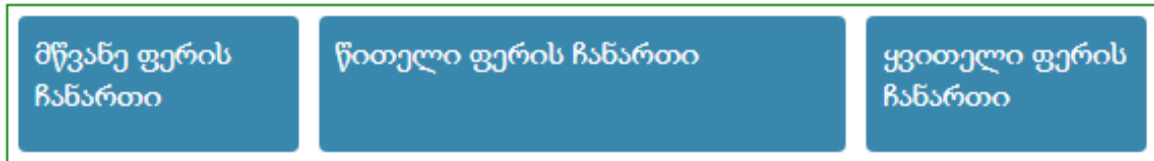
რიცხვი	სიგრძე, პროცენტი და ყველა სიგრძის საზომი ერთეული
<i>auto</i>	(<i>default</i>) ავტომატურად ენიჭება, რომლის სიგრძე ავტომატურად ტოლია მოქნილი ჩანართის სიგრძისა

მაგალითი 2.25 -ში ჩავანაცვლოთ სკრიპტი:

```
<style>
.კონტეინერი
{
    display: flex;
    display: -webkit-flex;
    border: solid 1px green;
}
.ჩანართი {
margin: 5px;
height: 52px;
padding: 8px;
border-radius: 4px;
background-color: #3a87ad;
color: white;
}
.მწვანე {
    flex-basis: 90px;
    -webkit-flex-basis: 90px;
}
.წითელი {
    flex-basis: 50px;
    -webkit-flex-basis: 50px;
}
.ყვითელი {
    flex-basis: 70px;
    -webkit-flex-basis: 70px;
}
```

```
}
</style>
```

შედეგი



flex-direction

განსაზღვრავს ჩანართების მიმართულებას და მას თვისებები განისაზღვრება *flex-direction*.

<i>row</i>	მოქნილი ჩანართების წარმოჩენა, როგორც რიგით
<i>Row-reverse</i>	იგივეა, რაც <i>row</i> უკუ მიმართულებით.
<i>column</i>	მოქნილი ჩანართების წარმოჩენა, როგორც სვეტებით
<i>column-reverse</i>	იგივეა, რაც <i>column</i> უკუ მიმართულებით.

სინტაქსის თვისებები დემონსტრირება მოვახდინოთ და მაგალით 2.25 ში ჩავამატოთ:

```
.კონტეინერი
{
  display:flex;
  display:-webkit-flex;
  border:solid 1px green;
  flex-direction:column;
  -webkit-flex-direction:column;
}
```

flex-flow

flex-flow თვისებები შემოკლებით *flex-direction* და *flex-wrap* თვისებებს მოიცავს:

<i>direction</i>	მიმართულება მოიცავს, ისეთ შესაძლებლობებს როგორიცაა: <i>row</i> , <i>row-reverse</i> , <i>column</i> და <i>column-reverse</i>
<i>wrap</i>	ხვეულა უნდა გაუჩნდეს თუ არა გვერდით, მისი სიდიდებია: <i>nowrap</i> , <i>wrap</i> , <i>wrap-reverse</i> , საწყის მნიშვნელობად <i>nowrap</i> ენიჭება

მაგალით 2.25-ში ჩავამატოთ:

```
.კონტეინერი {
  display:flex;
  display:-webkit-flex;
  border:solid 1px green;
```

```
flex-flow:row wrap;
-webkit-flex-flow:row wrap;
}
```

flex-grow

ერთი და იგივე კონტეინერი ჩანართი, რამდენად გაიზრდება და რამდენად მოქნილი იქნება სხვა ჩანართები, ერთადერთი თვისება აქვს, თუ რამდენი ჩანართი გაიზრდება ყველ მოქნილ ჩანართთან ერთად, საწყისი საზომი ერთეული 0 ია, მაგალითად 2.26-ში ილუსტრირება მოვახდინოთ ფერების საშუალებით.

მაგალითი 2.26

```
<!DOCTYPE html>
<html>
<head>
<style>
#text
{
width: 350px;
height: 200px;
display: -webkit-flex; /* Safari */
display: flex;
}
#text div:nth-of-type(1) {flex-grow: 1;}
#text div:nth-of-type(2) {flex-grow: 8;}
#text div:nth-of-type(3) {flex-grow: 1;}
#text div:nth-of-type(4) {flex-grow: 1;}
#text div:nth-of-type(5) {flex-grow: 1;}
</style>
</head>
<body>
<div id="text">
<div style="background-color:coral;">1</div>
<div style="background-color:lightblue;">8</div>
<div style="background-color:khaki;">1</div>
<div style="background-color:pink;">1</div>
<div style="background-color:lightgrey;"></div>
</div>
</body>
</html>
```

შედეგი



Flex-shrink

Flex-wrap

Shrink საშუალებით ელემენტები იკუმშება იმავე კონტეინერში ხოლო, **flex-wrap** მოიცავს გარკვეულ თვისებებს, ისეთებს, როგორიცაა **nowrap**, რომელიც საწყისი მნიშვნელობად ენიჭება, შემდეგ მოდის **wrap** და **wrap-reverse**.

```
#text
```

```
{
```

```
width:150px;
```

```
height: 150px;
```

```
border:1px solid black;
```

```
display: flex;
```

```
flex-wrap:nowrap;
```

```
}
```

მოცემული ელემენტში მოახდინოთ თითოეული თვისების შეცვლა და ვიპოვოთ ვიზუალური სხვაობა.

font

ფონტის საშუალებით შესაძლებელია საიტზე მოხდეს სასურველი ფონტის დეკლარირება, სხვადასხვა სტილის და სხვადასხვა ფორმის ფონტის წარმოსაჩენად ამისათვის შემუშავებულია **font** ელემენტი, რომელიც თავის მხრივ მოიცავს:

font	ფონტის თვისებების მთლიანობაში აღწერა
font-family	ფონტის ტიპები
font-size	ფონტის ზომა
font-style	ფონტის სტილი
font-variant	მისი საშუალებით შესაძლებელია თუ როგორ წარმოჩნდეს ფონტი, დიდი ასოებით, პატარა ასოებით და ა. შ
font-weight	ფონტის წონა (ზომა)

ცალკეულად თუ განვიხილავ **font** ელემენტში, შედის სხვა ელემენტებით

caption	გამოყენება ფონტი ისეთ კონტროლიერებზე, როგორიცაა ღილაკი, ჩამოშლისას მენიუ და ა. შ
icon	ქდეში იკონების გამოტანისას
menu	მენიუს შექმნისას
message-box	შეტყობინების ველისას
small-caption	პატარა ასოებით
status-bar	სტატუსის ველში

მაგალითი 2.27

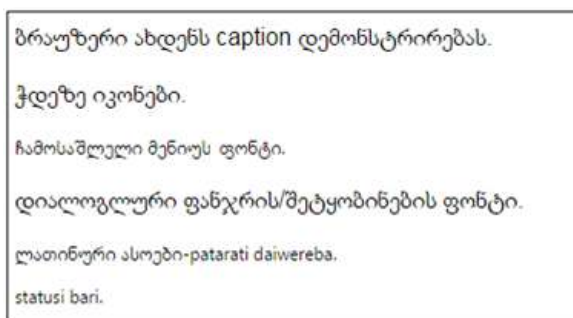
```
<!DOCTYPE html>
<html Lang="ka">
  <head>
```

```

<meta charset="utf-8" >
</head>
<body>
<p style="font:caption">ბრაუზერი ახდენს caption ჩვენებას.
</p>
<p style="font:icon">ჭდეზე იკონები. </p>
<p style="font:menu">ფონტი ჩამოსაშლელი მენიუს. </p>
<p style="font:message-box">დიალოგური ფანჯრის/შეტყობინების ფონტი.
</p>
<p style="font:small-caption">ლათინური ასოები-patarati daiwereba.
</p>
<p style="font:status-bar">სტატუსი ბარი.
</p>
</body>
</html>

```

შედეგი



ფონტის ოჯახი შეიძლება დავყოთ ორ ტიპად *generic family*, რომელიც ზოგადად სტანდარტად არის მიღებული და ყველა ბრაუზერი წაიკითხავს მას, ისეთ ფონტებს, როგორიცაა *Serif* ან *Monospace*, *serif*, *sans-serif*, *cursive*, *fantasy*, ხოლო მეორე ეტაპი გახლავთ *font family*, სადაც შესაძლებელია სასურველი ფონტის გაწერა.

მაგალითად, რა განსხვავება *Serif* და *Sans-serif* ფონტებს შორის, დააკვირდით სურათი 2.12 ასოების წვერობზე იპოვით სხვაობას.



სურ. 2.12

font-family

Font-family ელემენტში გაწერილი ფონტები რა რაოდენობისაც არ უნდა იყოს, თუ პირველი ფონტი ვერ წაიკითხავს, გადავა მეორეზე, შემდეგ მესამეზე და ა. შ შეგვიძლია ქართულ მაგალითზე რამდენიმე ქართული ფონტი მივუთითოთ, *AcadNusx*, *LitNusx* და *Sylfaen* ფონტები ავარჩიოთ.

მაგალითი 2.27

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<style>
p{
font-family:AcadNusx, LitNusx, Sylfaen;
}
</style>
</head>
<body>
<p> qarTulad warmoCndeba pirvelive teqsti</p>
</body>
</html>
```

შედეგი

ქართულად წარმონდება პირველივე ტექსტი

font-size

ფონტის ზომას განსაზღვრება პროცენტულად, პიქსელებში თუ სხვა საზომი ერთეულებით ასევე განისაზღვრება სიტყვიერად, *font-size* ელემენტში შემავალი თვისებებია: *large*, *larger*, *x-large*, *xx-large*, *medium*, *small*, *smaller*, *x-small* და *xx-small*.

```
P
{
font-size: xx-large;
}
```

font-size-adjust

font-size მონათესავე ელემენტია, მისი ზომა განისაზღვრება ფონტის ფართობით, ხოლო ფართობი განისაზღვრება *x-larger* გაყოფილი ფონტის ზომაზე ანუ *font-size*-ზე, მისი მხარდაჭერა ბრაუზერებიდან აქვს მხოლოდ *mozilla Firefox*-ს.

სკრიპტი ჩაიწერება :

```
P
{ font-size-adjust:0.70;
}
```

font-stretch

CSS3-ში შემუშავდა ახალი ელემენტი, რომელის საშუალებითაც ფონტის დაშორება შევიწროება და ა.შ არის შესაძლებელი.

font-stretch: *condensed, expanded, extra-condensed, extra-expanded, inherit, narrower, normal, semi-condensed, semi-expanded, ultra-condensed, ultra-expanded* და *wider*.

უნდა აღინიშნოს, რომ ამ ელემენტის მხარდაჭერა არცერთ ბრაუზერს არ აქვს და ბრაუზერის მწარმოებულები მოცემულ ელემენტზე მუშაობს, რომ ბრაუზერმა აღიქვას, ალბათ, უახლოვეს მომავალში, ვიხილავთ მის შესაძლებლობებს.

font-style

ფონტის სტილში შედის ფონტის სტილი და ტექსტი მისი ელემენტის შემავალი თვისებებია *normal, italic* და *oblique*.

მაგალითი 2.28

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
p{
font-size:18px;
}
</style>
</head>
<body>
<p style="font-style:oblique"> text oblique</p>
<p style="font-style:italic">text italic</p>
<p style="font-style:normal">text normal</p>

</body>
</html>
```

შედეგი

text oblique

text italic

text normal

font-variant

ფონტის ვარიანტი ლათინურ ასოებზე გვაძლევს შესაძლებლობას, წარმოაჩინოს ასოები პატარა ან დიდ ასოებად, მოცემული ელემენტის თვისებებში შემავალი ელემენტებია:

font-variant: normal და small-caps

small-caps მითითების შემდეგ ლათინურად დიდი ასოებით, რომ იყოს მაინც პატარად წარმოაჩენს.

font-weight

განსაზღვრავს ფონტის წონას, როგორი უნდა იყოს, წვრილი, მსხვილი, თვითონ ასოების სიმკვეთრს განსაზღვრავს სინტაქსი:

font-weight: 100, 200, 300, 400, 500, 600, 700, 800, 900, bold, bolder და lighter

როგორც ვხედავთ, სინტაქსიდან 100-დან 900-ამდე შეგვიძლია ფონტის სიდიდე განისაზღვრა.

@font-face

CSS3 Web Fonts მოიხსენება ინტერნეტში ფონტები, რომლის დამატება მომხმარებლებს სურთ ვებ გვერდებზე, ახლა სხვაობა რაშია? მაგალითად მომხმარებელმა შეიმუშავა სასურველი დიზანი, მოქნილობის და დახრილობის ფონტი, რომელიც ინტერნეტში არაა და მითუმეტეს ოპერაციულ სისტემას თუ ბრაუზერს არ აქვს მისი მხარდაჭერა, ანუ სხვაგვარათ, რომ ვთქვად არასტანდარტული ფონტია, როგორ მოვახდინოთ მისი ჩამატება და განთავსება სასურველ საიტზე? ამისთვის საჭირო ვიცოდეთ რა ფორმატის უნდა იყოს ფონტი და შემდეგ მოვახდინოთ მისი ატვირთვა ჰოსტინგზე-სადაც საიტის ფიალებია და ატვირთული შემდეგ მოვახდინოთ CSS სკრიპტში *@font-face* ელემენტის საშუალებით მისი ჩამატება.

მაგალითად 2.29

```
<!DOCTYPE html>
<html>
<head>
<style>
@font-face
{
    font-family: ჩემი-ფონტი;
    src: url(ღია_ფონტი.woff);
}
@font-face
{
    font-family: ჩემი-ფონტი;
    src: url(მუქი_ფონტი.woff);
    font-weight: bold;
}
```

```
div {
  font-family: ჩემი-ფონტი;
}
</style>
</head>
<body>
<div>
CSS3 text <b>text</b> fontebis gamoyeneba.
</div>
</body>
</html>
```

შედეგი:

CSS3 text **text** fontebis gamoyeneba.

TrueType Fonts (TTF)

TrueType ფონტის (შრიფტის) ფორმატი შემუშავებულია 1980-იანებში Apple და Microsoft კომპანიების მიერ, ამ ფონტის ფორმატი აღიქმება, როგორც საერთო ფორმატი და შემუშავებულია Mac OS და Microsoft Windows ოპერაციული სისტემებისათვის.

OpenType Fonts (OTF)

OpenType ფორმატი შემუშავებულია TrueType ბაზაზე, Microsoft-ს მიერ და გამოიყენება ზოგადად, ყველ ტიპის კომპუტერში.

The Web Open Font Format (WOFF)

WOFF არის ფონტის ფორმატი, რომელიც გამოიყენება Web გვერდებზე და იგი შემუშავდა 2009 წელს W3C-ს მიერ.

ძირითადი ფონტის(შრიფტის) ფორმატი რჩება TTF/OTF და WOFF, რომელიც ყველა ბრაუზერისათვის გასაგებია.

text

ტექსტთან ვიზუალური გაფორმებისათვის და სამუშაოდ გამოიყენება მისი ეფექტები ზემოთ ზოგადად, შევხეთ text-shadow განხილვისას, მისი სინტაქსში შემავალი ელემენტებია:

- color—ფერები
- direction—განისაზღვრება ტექსტის მიმართულება ltr და rtl თვისებები საშუალებით, მათ შევხეთ HTML5 განხილვისას, ჩაიწერება:

```
p {
  direction:ltr;
}
```

- Letter-spacing- რომელიც ზრდის და ამცირებს სიმბოლოებს შორის ინტერვალს, ელემენტის თვისებები განისაზღვრება normal და სიგრძის საზომი ერთეული პიქსელებით, ეს ის შემთხვევა, როდესაც სიგრძე შეიძლება

უარყოფილი სიდიდეც იყოს და ეს უარყოფილია მაშინ, როდესაც სიმბოლოებს შორის ინტერვალის შეკუმშვა გვსურს, მაგალითად:

```
P{
    Letter-spacing:-3px;
}
```

- **Line-height**- ელემენტი განსაზღვრავს აზრებს შორის ინტერვალს, მისი სინტაქსში შემავალი თვისებებია: *normal*, რიცხვი, *სიგრძე* და *%*, ოთხივე თვისების დემონსტრირება ხდება:

```
p {
    Line-height:normal;
    Line-height: 30px;
    Line-height: 2;
    Line-height:50%;
}
```

- **text-align**- ტექსტის პოზიცი, მისი თვისებებია: *left*, *right*, *center* და *justify*.
- **text-decoration**- ტექსტის დეკორაცია განისაზღვრება ხაზის გავლებით, მაგრამ CSS3-ში მას დაემატა *text-decoration-color*, რომელიც განსაზღვრავს ხაზის ფერს.

text-decoration none, *underline*, *overline* და *line-through*

text-decoration-line, *text-decoration-style*, და *text-decoration-color* ისეთივე თვისებები აქვს, როგორც სტილს და ფერს.

- **text-indent**- ტექსტის აზრადი, ეთითება სიგრძის საზომი ერთეულებში ასევე *%*-ებში.

text-indent: 20px;

- **text-shadow**- ტექსტურ ჩრდილებზე ზემოთ შევხეთ.
- **text-transform**- ტექსტების გარდაქმნა, ლათინური ასოები წარმოჩნდეს როგორც დიდი ან პატარა ასოებით.

none	ისევე წარმოჩნდება, როგორიცაა
capitalize	ყველა პირველი სიმბოლო დიდაა
uppercase	გარდაქმნის ყველა სიმბოლოს ყველა დიდად
lowercase	გარდაქმნის ყველა სიმბოლოს ყველა პატარა.

მაგალითად, დავწეროთ სამ სტრიქონიანი ტექსტი ლათიურად ერთნაირად და მიუთითოთ სამივე მნიშვნელობა.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase
{
    text-transform: uppercase;
```

```

}
p.lowercase
{
    text-transform: lowercase;
}
p.capitalize
{
    text-transform: capitalize;
}
</style>
</head>
<body>
<p class="uppercase">raime texti.</p>
<p class="lowercase">raime texti</p>
<p class="capitalize">raime texti</p>
</body>
</html>

```

შედეგი

RAIME TEXTI.

raime texti

Raime Texti

- ✓ *unicode-bidi*-მისი მონათესავე ტექსტია *direction* ელემენტი, რომლის საშუალებითაც შესაძლებელია ტექსტის არეულად დაწერა და განსაზღვრა მიმართულების, მისი სინტაქსი მოიცავს თვისებებს:

normal, *embed* და *bidirectional-override*

მაგალითი 2.30

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
div.მაგალითი
{
    direction: rtl;
    unicode-bidi: bidi-override;
}
</style>
</head>
<body>
<div>რაიმე ტექსტი.</div>
<div class="მაგალითი">რაიმე ტექსტი.</div>
</body>
</html>

```

შედეგი

რაიმე ტექსტი.

.იტსქეტ ემიარ

- ✓ **vertical-align**-განსაზღვრავს ელემენტის პოზიციას, ელემენტში შეიძლება მოთავსებული იყოს ტექსტი, სურათი ან სხვა. მისი თვისებები განისაზღვრება შემდეგი სიდიდეებით:

baseline, სივრძით, sub, super, top, text-top, middle, bottom, text-bottom, initial და inherit;

☞ მაგალით 2.31-ში მოცემული ორი თვისების დემონსტრირება, დანარჩენი თვისებების უკეთ აღსაქმელად, სკრიპტში ჩაანაცვლეთ და ნახეთ შედეგი

```
vertical-align: baseline;
vertical-align: sub;
vertical-align: super;
vertical-align: text-top;
vertical-align: text-bottom;
vertical-align: middle;
vertical-align: top;
vertical-align: bottom;
```

```
/* სივრძის საზომი ერთეულები */
vertical-align: 10em;
vertical-align: 4px;
```

```
/* პროცენტები */
vertical-align: 20%;
```

```
/* გლობალური ცვლადები */
vertical-align: inherit;
vertical-align: initial;
vertical-align: unset;
```

მაგალითი 2.31

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>vertical-align</title>
</head>
<body>
  <div style="font-size: 2em"> ჩვენ ვსწავლობთ <span
style="vertical-align: sub">CSS</span>3<span
```

```

style="vertical-align: 5px; font-size: 0.8em"> ენას</span>
</div>
</body>
</html>
შედეგი

```

ჩვენ ვსწავლობთ CSS3 ენას

- ✔ **white-space** ცარიელი სივრცის განსაზღვრა, რომელიც არის ელემენტებს შორის წარმოქნილი, მისი სინტაქსის შემავალი თვისებებია :
normal-ნორმა ტექსტი ჩვეულებრივად წარმოჩნდება.
nowrap-არ შექმნის ხვეულას და ტექსტი იქნება გაწეილი.
pre-არ ხდება წყვეტა ტექსტში მხოლოდ შესაძლებელია **
** ელემენტის საშუალებით
pre-wrap-ასევე ივსება ტექსტის ველი და არ ხდება წყვეტა;
pre-line- სივრცე თანმიმდევრობით იშლება.

სკრიპტი გამოიყურება :

```

code
{
  white-space: pre;
}

```

- ✔ **word-spacing:** სიტყვებს შორის სივრცის განსაზღვრება, მისი სინტაქსის შემადგენილი ნაწილია *normal* და სივრცე.

```

p
{
  word-spacing: 10px;
}

```

ეს ნიშნავს იმას. რომ პარაგრაფში დაწერილი ყოველივე სიტყვა **10px** დაშორებული ერთმანეთისაგან იქმნება.

- ✔ **word-break:** სიტყვებს შორის დაშორება და წყვეტა, ეს ელემენტი მუშაობს CJK სიმბოლოების გარდა, ყველა სიმბოლოზე, *CJK* შედის კორეული დამწერლობა, ჩინური და იაპონური.

normal ავტომატურად ენიჭება და წარმოადგენს ისევე ტექსტს, როგორც წერია.

break-all ორ სიმბოლოს შორის შეიძლება მოხდეს წყვეტა.

keep-all წყვეტა არ ხდება წყვილ სიმბოლოებს შორის.

მაგალითი 2.32

```

<!DOCTYPE html>
<html>
<head>

```

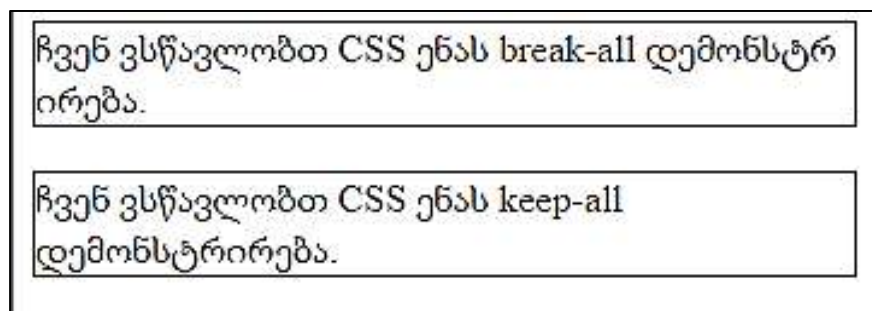


```

<meta charset="utf-8">
<title> word-break</title>
<style>
div.ტესტ1
{
    word-break:break-all;
    border: 1px solid #151515;
    width: 340px;
}
div.ტესტ2
{
    word-break:keep-all;
    border: 1px solid #151515;
    width: 340px;
}
</style>
</head>
<body>
<div class="ტესტ">
ჩვენ ვსწავლობთ CSS ენას. break-all დემონსტრირება.
</div><br>
<div class="ტესტ2">
ჩვენ ვსწავლობთ CSS ენას keep-all დემონსტრირება. </div>
</body>
</html>

```

შედეგი



უნდა აღინიშნოს, რომ *word-break* ელემენტი, ახალი ელემენტი რომელიც შემუშავებულია CSS3-სთვის დღეს ყველა ბრაუზერს მისი წაკითხვა შეუძლია.

განვიხილოთ CSS3-ის *Text* ელემენტის სხვა თვისებები, რომლებიც გვაქვს ესენია:

- ✓ *text-align-last*-ბოლოს აბზაციის პოზიცია.
auto, left, right, center, justify, start, end, initial
 და *inherit*;
 კოდის დემონსტრირება :

```
div
{
  text-align: justify;
  -moz-text-align-last: center;
  text-align-last: center;
}
```
- ✓ *text-justify*-ყველა ტექსტის კიდეები გასწორდება, ამ ელემენტის მხარდაჭერა აქვს მხოლოდ *microsoft edge* ბრაუზერს.
auto, inter-word, inter-ideograph, inter-cluster, distribute, kashida, trim, initial და *inherit*;
- ✓ *text-overflow* - განსაზღვრავს, თუ როგორ მოთავსდეს ტექსტი არეალში.
- ✓ მისი სინტაქსი განისაზღვრება:
clip, ellipsis, string initia და *inherit*;
- ✓ *text-emphasis* მოკლედ აღწერს *text-emphasis-style* და *text-emphasis-color*, ამ ელემენტის მხარდაჭერა არცერთ ბრაუზერს არ აქვს, მისი კოდი კი ასე ჩაიწერება:

```
p {
  text-align: justify;
  text-justify: inter-word;
}
```

Margin

Box-model-ის შემადგენელი ნაწილია, რომელიც საზღვრის თვისებას ოთხი ელემენტით განსაზღვრავს: *top, right, bottom* და *left*, ოთხივე ელემენტს აქვს ერთი და იგივე თვისება და ოთხივე განისაზღვრება: სიგრძით (პროცენტულად ან სხვა ერთეულში) და *auto*

```
margin-top: 20px;
margin-bottom: 10px;
margin-right: 10px;
margin-left: 80px;
```

margin ჩაწერთ ოთხი საშუალებათა, როგორ შეიძლება, დაიწეროს ელემენტი მათი თვისებით.

margin: 15px 25px 35px 10px- განისაზღვრება ოთხივე მხარე სხვაასხვა სიდიდებით
margin: 25px 30px 25px-როდესაც ხდება ორი გვერდის განსაზღვრა :
top-25px, right და *left* არის *30px*, ხოლო *bottom-25px*
margin: 25px 30px-top და *bottom* არის *25px*, ხოლო *right* და *left 30px*.

margin: 15px- ოთხივე მხარე იქნება *15px*.

Padding

Padding განსაზღვრავს სივრცეს კონტენტის გარშემო (იხ სურ. 2.8), რომელიც შიგ საზღვრების შიგნით განისაზღვრება, მასაც როგორც *margin*-ს საერთო თვისებები აქვთ, და ეს თვისებები განისაზღვრება *top*, *right*, *bottom* და *left*, ოთხივე ელემენტს აქვს ერთი და იგივე თვისება და ოთხივე განისაზღვრება: სიგრძის ერთეულით და პროცენტულად.

padding-top
padding-right
padding-bottom
padding-left

padding ისევე განისაზღვრება ოთხივე შემთხვევით, როგორც *margin*, მაგალითი 2.33-ში ნაჩვენებია ორი ელემენტის დამოკიდებულებას ერთმანეთის მიმართ.

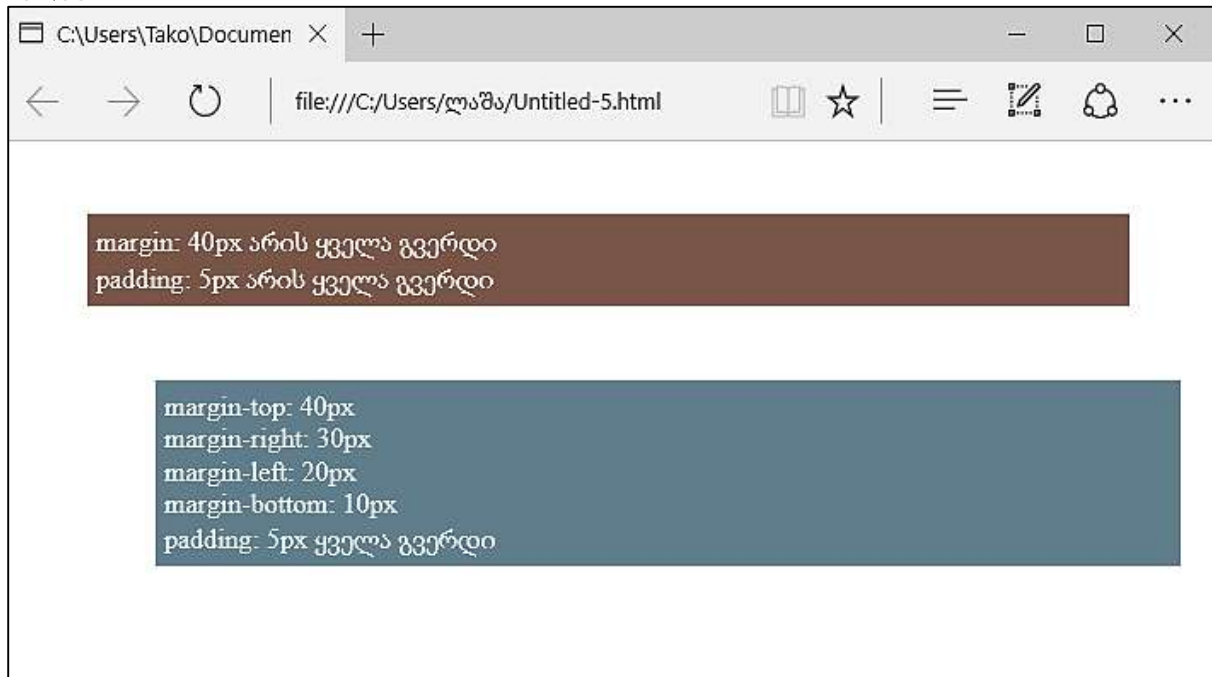
მაგალითი 2.33

```
<html>
<head>
<meta charset="utf-8">
<style>
  div.text1 {
    color:white;
    background-color: #795548;
    border: 1px solid;
    padding: 5px;
    margin: 40px;
  }

  div.text2
{
  background: #607D8B;
  color: azure;
  border: 1px solid;
  padding: 5px;
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 40px;
  margin-left: 80px;
}
</style>
</head>
<body>
<div class="text1">
  margin: 40px არის ყველა გვერდი <br>
  padding: 5px არის ყველა გვერდი
</div>
<div class="text2">
  margin-top: 40px<br>
  margin-right: 30px<br>
  margin-left: 20px<br>
```

```
margin-bottom: 10px<br>
padding: 5px ყველა გვერდი
</div>
</body>
</html>
```

შედეგი



უფრო გასაგები რომ იყოს, *margin* აღწერს ელემენტებს საზღვრებს გარეთ, ხოლო *padding*-საზღვრების შიგნით.

position

ელემენტის დასახელებიდანაც ვხდებით, რომ საქმე გვაქვს პოზიციასთან, რომელიც ენიჭება ელემენტის, ელემენტის პოზიცია ხაისათდება *static*, *absolute*, *fixed*, *relative*, *initial* და *inherit*, მას საწყის მნიშვნელობად ენიჭება *static*.

<i>relative</i>	განსაზღვრავს ნორმალურ პოზიციას, რომელიც ოთხივე მხარის მიმართ არის რელევანტური.
<i>absolute</i>	შემთხვევაში წინა ელემენტთან არის მიახლოებული
<i>fixed</i>	შემთხვევაში ელემენტი ყოველთვის დგას ერთსა და იმავე ადგილას და არ იცვლება.
<i>static</i>	არანაირი ცვლილება არ ხდება ოთხივე მხარის მიმართ.

ელემენტის პოზიციაში შედის სხვა ელემენტებიც რომელიც განვიხილოთ, ისეთი როგორიცაა: *top*, *right*, *bottom* და *left*, ოთხივე ელემენტი განისაზღვრება სიგრძით, % და *auto*-ით, ასევე შედის *overflow*-ს ელემენტი თავის მონათესავე ელემენტებით *overflow-x* და *overflow-y* ელემენტებით, რომელიც ზემოთ განვიხილეთ.

<i>clip</i>	ამცირებს ელემენტის ობიექტს
<i>cursor</i>	წარმოაჩენს კურსორს სხვადასხვა გვარად.
<i>z-index</i>	ელემენტის წარმოჩენას

clip

რა მოხდება იმ შემთხვევაში, თუ სურათი დიდია, მისი ჩამოჭრა დაშესაბამისად განთავსება, რომ არ ვიწვალთ, რაც დიდ დროს და ენერგიას წაიღებს, შეგვიძლია **CSS**-ს საშუალებით გადავიწყვიტოთ მსგავსი პრობლემა.

clip ელემენტის თვისებები განისაზღვრება :

Auto და *shape*

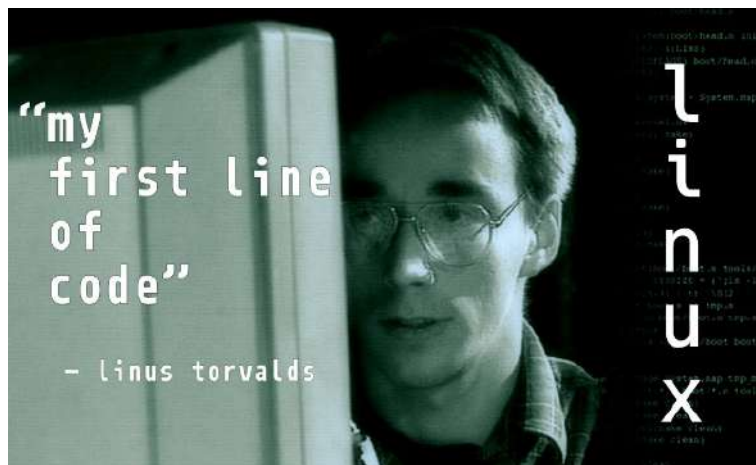
თავის მხრივ *shape* ელემენტი მოიცავს ოთხივე მხარის მიმართ დამოკიდებულებას.

მაგალითი 2.34

```
<!DOCTYPE html>
<html>
<head>
<style>
img
{
    position: absolute;
    clip: rect(0px,60px,200px,0px);
}
</style>
</head>
<body>

</body>
</html>
```

მაგალითად ავიღოთ ლინუქს თოვარდის სურათი, რომლის სახელს უკავშირდება ოპერაციული სისტემის *Linux*-ის შექმნას.



მოცემული სურათი ჩავსვათ მაგალით 2.34-ში, მივიღეთ :



cursor

წარმოაჩენს სხვადასხვა ზომის კურსორს.

auto, crosshair, default, e-resize, grab, help, move, n-resize, ne-resize, nw-resize, pointer, progress s-resize, se-resize, sw-resize, text, w-resize, wait, not-allowed და no-drop.

მაგალითი 2.35

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<p> მაუსის სხვადასხვა ზომა </p>
<span style="cursor:auto"> auto</span>
<br>
<span style="cursor:crosshair">crosshair</span>
<br>
<span style="cursor:default" >default</span>
<br>
<span style="cursor:e-resize"> e-resize</span>
<br>
<span style="cursor:grab"> grab</span>
<br>
<span style="cursor:help"> help</span>
<br>
<span style="cursor:move">move</span>
<br>
<span style="cursor:n-resize">n-resize</span>
<br>
<span style="cursor:ne-resize">ne-resize</span>
<br>
<span style="cursor:nw-resize">nw-resize</span>
<br>
<span style="cursor:pointer">pointer</span>
<br>
<span style="cursor:progress">progress</span>
<br>
<span style="cursor:s-resize">s-resize</span>
```

```

<br>
<span style="cursor:se-resize">se-resize</span>
<br>
<span style="cursor:sw-resize">sw-resize</span>
<br>
<span style="cursor:text">text</span>
<br>
<span style="cursor:w-resize">w-resize</span>
<br>
<span style="cursor:wait">wait</span>
<br>
<span style="cursor:not-allowed">not-allowed</span>
<br>
<span style="cursor:no-drop">no-drop</span>
<br>
</body>
</html>

```

შეინახეთ და შედეგი ნახეთ ბრაუზერში.

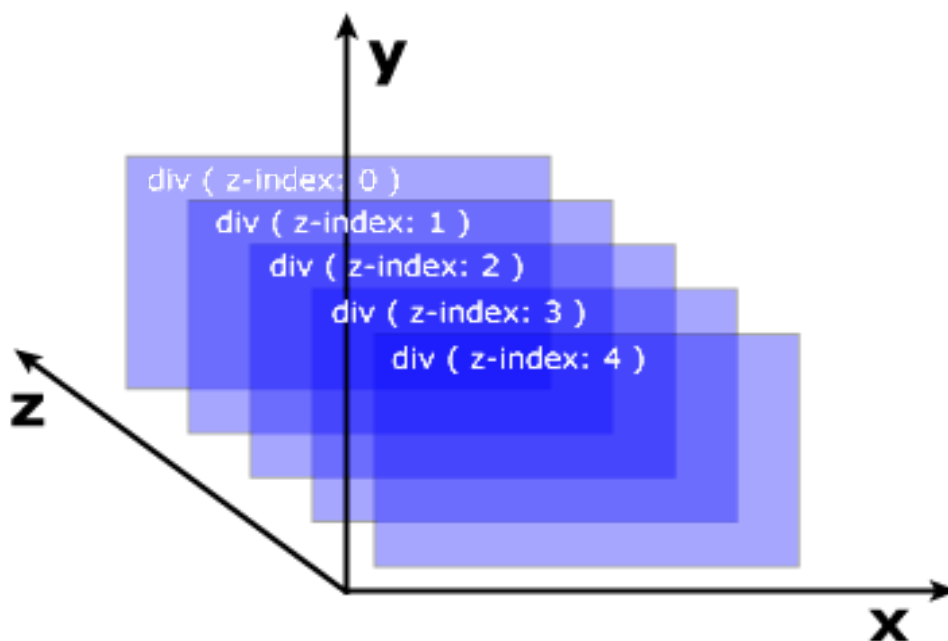
z-index

z-index ელემენტი მუშაობს იმ შემთხვევაში, თუ პოზიციის ელემენტები აღწერილი იქნება. *z-index* ელემენტის თვისებებია *auto* და რიცხვები ეთითება. კოდი ჩაიწერება შემდეგნაირად:

```

{
  position: absolute;
  background-color: grey;
  z-index: 2;
}

```



სურ.2.13 ნაჩვენებია *z-index*-ს დემონსტრირება, როგორ ხდება გვერდების გადაადგილება

Links

ლინკები-ბმულები შეიძლება წარმოვადგინოთ ოთხნაირად:

a:link - ლინკი, რომელიც არ დაგვიჭერია;

a:visited - ლინკი, რომელსაც დავაჭირეთ;

a:hover - ლინკი, რომელზეც მაუსს გადავატარებთ;

a:active - ლინკი, რომელიც მოცემულ მომენტში აქტიურია.

ამ ოთხი წარმოდგენიდან უნდა ვიცოდეთ, რომ ამ ელემენტების წარმოდგენა ხდება შემდეგნაირად :

a:link და *a:visited* შემდეგ უნდა დაიწეროს *a:hover*, ხოლო *a:hover* მერე იწერება *a:active* .

```
<style>
a:link {
    background-color: blue;
}
a:visited {
    background-color: cyan;
}
a:hover {
    background-color: green;
}
a:active {
    background-color: red;
}
</style>
```

როგორც, ხედავთ, ორი წერტილით ხდება ელემენტის აღწერა, მათ ფსევდო კლასები ჰქვია, ამ კლასს ცოტა ქვემოთ შევხებით.

List

სიის ელემენტის საშუალებით შესაძლებელია გადანომვრა, ნუმერაციი მაგივად შესაძლებელია სურათის ჩასმა.

ოთხი ელემენტი შედი *list* ელემენტში:

- ✓ *list-style*-სტანდარტული სტილით გადანომვრა რომელიც მოიცავს ყველა ელემენტს:

```
ul {
    list-style: square url("სურათის ლინკი");
}
```

- ✓ *list-style-image* სურათის ლინკის ჩასმა:

```
ul {
    list-style-image: url('surati');
}
```

- ✓ *list-style-position* - სურათის პოზიცია, რომელიც მისი სინტაქსი განისაზღვრება: *inside* და *outside* .

```
ul {
    list-style-position: inside;
}
```


- ✓ *list-style-type*-შედის, თუ როგორ უნდა მოხდეს გადანომვრა, წვრილად, ოთხკუთხედებით, ნუმერაციით თუ ალფავიტით, უნდა აღინიშნოს, რომ ქართული ასოებითაც ხდება გადანომვრა და ამის მხარდაჭერა CSS-ს აქვს.

მისი სინტაქსია:

list-style-type: ცვლადი;

მაგალით 2.36-ში მოცემულია ყველა ცვლადი, თუ როგორ აღიწერება მათი შედეგები

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a
{
list-style-type: circle;}
ul.b {
list-style-type: disc;
}ul.c
{
list-style-type: square;
}
ol.ქართული
{
list-style-type: georgian;
}
ol.სომხური
{
list-style-type: armenian;
}
ol.e
{
list-style-type: cjk-ideographic;
}
ol.f
{
list-style-type: decimal;
}
ol.g
{
list-style-type: decimal-leading-zero;
}
ol.i
{
list-style-type: hebrew;
}
ol.j {
list-style-type: hiragana;
}
ol.k
{
```

```

list-style-type: hiragana-iroha;
}
ol.l
{
list-style-type: katakana;
}
ol.m
{
list-style-type: katakana-iroha;
}
ol.n
{
list-style-type: lower-alpha;
}
ol.o
{
list-style-type: lower-greek;
}
ol.p
{ list-style-type: lower-latin;
}
    ol.q
    {
    list-style-type: lower-roman;
    }
    ol.r {
    list-style-type: upper-alpha;
    }
    ol.s {
    list-style-type: upper-latin;
    }
    ol.t {
    list-style-type: upper-roman;
    }
    ol.u {
    list-style-type: none;
    }
    ol.v {
    list-style-type: inherit;
    }
</style>
</head>
<body>
<ul class="a">
  <li>Circle type</li>
  <li>β&α</li>
  <li>γ&β</li>
</ul>
<ul class="b">
  <li>Disc type</li>

```

```

    <li>ჩაი</li>
    <li>ყავა</li>
</ul>
<ul class="c">
    <li>Square type</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ul>
<ol class="ქართული">
    <li>ქართული ტიპი</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="d">
    <li>ხომბური</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="e">
    <li>Cjk-ideographic type</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>

<ol class="f">
    <li>Decimal type</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="g">
    <li>Decimal-leading-zero type</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="ქართული">
    <li>ქართული ტიპი</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="i">
    <li>Hebrew type</li>
    <li>ჩაი</li>
    <li>ყავა</li>
</ol>
<ol class="j">
    <li>Hiragana type</li>
    <li>ჩაი</li>

```

```

    <li>ㇿ</li>
</ol>
<ol class="k">
    <li>Hiragana-iroha type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="l">
    <li>Katakana type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="m">
    <li>Katakana-iroha type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="n">
    <li>Lower-alpha type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="o">
    <li>Lower-greek type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="p">
    <li>Lower-latin type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="q">
    <li>Lower-roman type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="r">
    <li>Upper-alpha type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>
<ol class="s">
    <li>Upper-latin type</li>
    <li>ㇿ</li>
    <li>ㇿ</li>
</ol>

```

```

<ol class="t">
  <li>Upper-roman type</li>
  <li>ჩაი</li>
  <li>ყავა</li>
</ol>
<ol class="u">
  <li>None type</li>
  <li>ჩაი</li>
  <li>ყავა</li>
</ol>
<ol class="v">
  <li>inherit type</li>
  <li>ჩაი</li>
  <li>ყავა</li>
</ol>
</body>
</html>

```

შედეგი

- o Circle type

- o ჩაი

- o ყავა

- Disc type

- ჩაი

- ყავა

- Square type

- ჩაი

- ყავა

ა. ქართული ალფავიტი

ბ. ჩაი

გ. ყავა

1. სომხური ალფავიტი

2. ჩაი

3. ყავა

— Cjk-ideographic ალფავიტი

二. ჩაი

三. ყავა

1. Decimal ტიპი

2. ჩაი

3. ყავა

01. Decimal-leading-zero

02. ჩაი

03. ყავა

א. Hebrew

ב. ჩაი

ג. ყავა

あ. Hiragana

い. ჩაი

う. ყავა

い. Hiragana-iroha

ろ. ჩაი

は. ყავა

ア. Katakana e

イ. ჩაი

ウ. ყავა

イ. Katakana-iroha

ロ. ჩაი

ハ. ყავა

a. Lower-alpha

b. ჩაი

c. ყავა

α. Lower-greek

β. ჩაი

γ. ყავა

a. Lower-latin

b. ჩაი

c. ყავა

i. Lower-roman

ii. ჩაი

iii. ყავა

A. Upper-alpha

B. ჩაი

C. ყავა

A. Upper-latin

B. ჩაი

C. ყავა

I. Upper-roman

II. ჩაი

III. ყავა

None

ჩაი

ყავა

- inherit

- ჩაი

- ყავა



ჩვენ შედგასა წარმოგაჩინებოლთქვეწროვასკროპტ აკრფავთცაღმარეს მთულსიგრძეზე წარმოწნდება

Gradients

გრადიენტების საშუალებით შესაძლებელია ორი ან მეტი ფერის გამოჩენა, ადრე ამ სურათებს ხატავდნ და მისი საშუალებით ხდებოდა წარმოჩენა, შემდეგ შემუშავდა ელემენტი, რომელიც ორი ან მეტი ფერის ურთერთობისას წარმოაჩენს ფერებს.

გრადიენტი ტიპისაა:

Linear Gradients-რომელიც მოძრაობს ქვემოთ, ზემოთ, მარცნივ, მარჯვნივ და დიაგონალურად.

Radial Gradients- რომელიც ცენტრიდან განისაზღვრება.

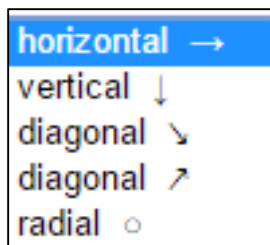
როგორც სხვა ელემენტს მასაც აქვს ბრაუზერების მხარდაჭრა -webkit-, -moz-, და -o-, ჯერ განვიხილოთ **Linear Gradients**, **რომელიც ასე იწერება:**

`background: linear-gradient(to blue, green ,grey);`

დავწეროთ დიაგონალური

`background: -webkit-linear-gradient(left top, blue, green);`

გრადიენტების ფერთა გადანაწილება ხდება ჰორიზონტალურად, ვერტიკალურად, დიაგონალურად



ასევე შესაძლებელია კუთხე მივუთითოთ:

`linear-gradient(-90deg, red, yellow);`

ავიღოთ სტანდარტული სინტაქსი, რომელშიც მოცემულია გრადიენტები;

```
<html>
<head>
<style>
  div.ტექსტ
  {
    color:white;
    background: linear-gradient(red,blue);
  }
</style>
</head>
<body><div class="ტექსტ">
გრადიენტი
</div>
</body>
</html>
```

შედეგი



ხოლო, თუ *background*-ხაზე ჩავამტებთ 90deg ანუ გრადუს, მივანიჭებთ, ფერები გადაინაცვლებს:

```
background: linear-gradient( 90deg, red,blue);
```

სხვა შედეგს მოგვცემს:



ჩავამატოთ კოდში ჰორიზონტალური გრადიენტის კოდი

```
<style>
div.ტექსტ
{
color:white;
background: #f79621;
background:-moz-linear-gradient(left, #f79621 0%, #4e8bba 55%, #f9c667
100%);
background:-webkit-linear-gradient(left, #f79621 0%,#4e8bba 55%,#f9c667
100%);
background: linear-gradient(to right, #f79621 0%,#4e8bba 55%,#f9c667
100%);
}
</style>
```

შედეგი



რაც შეეხება კოდს დააკვირდით

```
<head>
<style>
div.ტექსტ {
color:white;
height:250px;
background: #f79621;
background:-moz-linear-gradient(top, #f79621 0%, #4e8bba 55%, #f9c667
100%);
background:-webkit-linear-gradient(top,#f79621 0%,#4e8bba55%,#f9c667
100%);
background: linear-gradient(to bottom, #f79621 0%,#4e8bba 55%,#f9c667
100%);
}
</style>
</head>
```

შედეგი



როგორც ხედავთ, ფერები ვერტიკალურად არის განლაგებული, ზემოდან ქვემოთაა, ხოლო ვერტიკალურადაა, რაც შეეხება პროცენტებს, ეს ფერების გამეორების სიხშირეა 100%-ცენტი ნიშნავს, რომ სრულად იმეორებს, 55% -ი კი ფერები ამ სიდიდითა, წარმოდგენილი და ა.შ.

ასევე შესაძლებელია *transparent*-ის ფერების გამოყენებაც *rgba* ფერების გამოყენებით, როგორც ვიცით გამჭვირვალობის *0*-დან *1*-ამდეა, ქვემოთ ვნახოთ როგორ ხდება სტრიქონში დემონსტრირება.

```
{  
  border-color: #c5ae87;  
  background: rgba (11, 180, 170, 0.05);  
}
```

როგორც ვხედავთ, ემატება *rgb*-ში *Transparent*-ი

Radial Gradients-ფერების აღქმა იწყება შუიდან, და ფერთა გამა წარმოჩნდება ისე ელიფსურად და შემდეგ ამ ელიფსური და წრიულ არეების გადანაწილება ხდება სხვადასხვაგვარად, ფერების აღწერისას როგორც ელფის ზომა, ასევე საწყისი და საბოლოო, ფერი უნდა დავწეროთ, მაგალითად, თანამიმდევრობით ფერებით რომელიც გვინდა იყოს, ავიღოთ ოთხი ფერი: *red*, *yellow*, *green* და *blue* .

```
background: radial-gradient(red, yellow, green,blue);
```

როგორც აღვნიშნეთ, საწყისი მნიშვნელობა, რომელიც ენიჭება *Radial*-ს, არის ელიფსური, ხოლო თუ გვსურს, რომ წრიული იყოს, მაშინ სინტაქსში ვუთითებთ *circle*-ს.

```
background: radial-gradient(circle, red, yellow, green,blue);
```

მის სინტაქსი განისაზღვრება :

```
radial-gradient( circle, ... )  
radial-gradient( ellipse, ... )  
radial-gradient( <extent-keyword>, ... )  
radial-gradient( circle radius, ... )  
radial-gradient( ellipse x-axis y-axis, ...
```


განისაზღვრება ოთხი სიდიდით: *closest-side*, *farthest-side*, *closest-corner* და *farthest-corner*.

მაგალითი 2.37

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
#closestside
{   height: 250px;
    width: 250px;
    background: radial-gradient (closest-side at 60% 55%, red, yellow,
black);
}
#farthestside
{   height: 250px;
    width: 250px;
    background: radial-gradient(farthest-side at 60% 55%, red, yellow,
black); }
#closestcorner
{   height: 250px;
    width: 250px;
    background: radial-gradient(closest-corner at 60% 55%, red, yellow,
black);
}
#farthestcorner
{
    height: 250px;
    width: 250px;
    background: radial-gradient(farthest-corner at 60% 55%, red, yellow,
black);
}
</style>
</head>
<body>
<h3>
Radial Gradients - ოთხი სიდიდეა წარმოდგენილი
</h3>
<p>
<strong>closest-side:</strong>
</p>
<div id="closestside"></div>
<p>
<strong>
farthest-side:
</strong> </p>
<div id="farthestside">
</div>
```

```

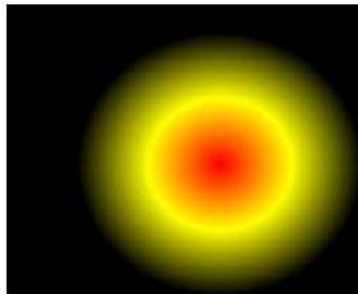
<p>
  <strong>closestcorner:</strong> </p>
<div id="closestcorner"></div>
  <p><strong>farthest-corner (არის საწყისი მნიშვნელობა):</strong> </p>
<div id="farthestcorner">
</div>
</body>
</html>

```

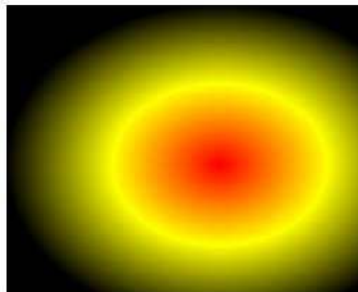
შედეგი

Radial Gradients -ოთხი სიდიდეა წარმოდგენილი

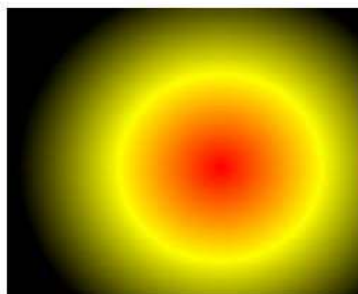
closest-side:



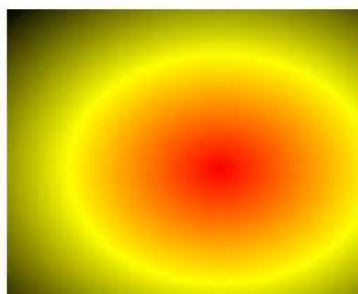
farthest-side:



closestcorner:



farthest-corner (არის საწყისი მნიშვნელობა):



ასევე შესაძლებელია ფერის გამეორება :

`background: repeating-radial-gradient(black,white 40%, green 45%);`

display

CSS-ში ელემენტების გადანაწილებისა და წარმოჩენისთვის გამოყენება, თუ როგორ წარმოჩნდეს HTML-ს ელემენტები, მას აქვს ორი მონათესავე ელემენტი, როგორიცაა *block* და *none*.

Block-ელემენტი იღებს მაქსიმალურ შესაძლებლობებს ელემენტისაგან და ყოველთვის იწყება ახალი ხაზიდან და მუშაობს ისეთ ელემენტებთან, როგორიცაა: `<div>`, `<h1>` - `<h6>`, `<p>`, `<form>`, `<header>`, `<footer>` და `<section>`.

None-ხდება *javascript*-ის ელემენტების დამალვა და ისე, რომ არ გიწევს რიამეს გარდაქმნა, წაშლა ან დამატება.

Inline-საშუალებით შესაძლებელია ელემენტების ერთ ხაზე წარმოჩენა:

მაგალითი 2.38

```
<!DOCTYPE html>
<html>
<head>
<style>
li {
    display: inline;
}
</style>
</head>
<body>
<p>სამი სხვადასხვა საიტი წარმოვაჩინოთ ერთ ხაზე: </p>
<ul>
    <li><a href="http://scripts.ge" target="_blank">SCRIPTS.GE</a></li>
    <li><a href="http://gtu.ge" target="_blank">GTU.GE</a></li>
    <li><a href="www.facebook.com" target="_blank">FACEEBOOK.</a></li>
</ul>
</body>
</html>
```

სამი სხვადასხვა საიტი წარმოვაჩინოთ ერთ ხაზე:

[SCRIPTS.GE](http://scripts.ge) | [GTU.GE](http://gtu.ge) | [FACEEBOOK.](http://www.facebook.com)

სინტაქსი:

`display: value;`

სინტაქსში მოცემულია შემავალი სხვა თვისების დემონსტრირება: *inline*, *block*, *list-item* და *none*

მაგალითი 2.39

```
<!DOCTYPE html>
```

```

<html>
<head>
<style>
body
{
}
p {
  color:white;
  background-color:blue;
  display: inline;
}
</style>
</head>
<body>
<p> display ელემენტების ჩვენება </p>
</body>
</html>

```

შედეგი

display ელემენტების ჩვენება



ანალოგიურად მოვახდინოთ დანარჩენი ელემენტების ჩასმა *display: inline-* ს მაგივრად.

<i>Flex</i>	<i>Flex-კონტეინერში გამოიტანს ელემენტებს</i>
<i>inline-block</i>	გამოიტანს ელემენტს ბლოკებად
<i>inline-flex</i>	გამოიტანს ელემენტს, როგორც flex კონტეინერს
<i>inline-table</i>	ხაზე გამოიტანს ცხრილს
<i>run-in</i>	გამოიტანს ელემენტს, როგორც block ან inline ელემენტს და დამოკიდებულია მის კონტექსტზე
<i>table</i>	იღებს <table> ელემენტის თვისებებს
<i>table-caption</i>	იღებს <caption> ელემენტის თვისებებს
<i>table-column-group</i>	იღებს <colgroup> ელემენტის თვისებებს
<i>table-header-group</i>	<thead> ელემენტის თვისებებს
<i>table-footer-group</i>	იღებს <tfoot> ელემენტის თვისებებს
<i>table-row-group</i>	იღებს <tbody> ელემენტის თვისებებს
<i>table-cell</i>	იღებს <td>ელემენტის თვისებებს
<i>table-column</i>	იღებს <col> ელემენტის თვისებებს
<i>table-row</i>	იღებს <tr> ელემენტის თვისებებს.

კომბინატორები

სხვადასხვა საიტის *Source*-ში თუ ჩავიხედავთ, დავინახავთ ისეთ ოთხ სიმბოლოს, რომელიც ჩვენში აუცილებლად ინტერესს გამოწვევს, როგორიცაა:

Space-ანუ სივრცე, როდესაც ორ ელემენტს შორის დამაკავშირებელი სიმბოლო არაა, მხოლოდ ორი სიმბოლოა *div p { }* .

შვილობილი სელექტორი *>* რომელიც მოიცავს ელემენტში არსებულ ელემენტებს და ჩაიწერება *div > p { }*.

დამატებითი ელემენტი *+* რომელიც შესაძლებელია მონიშნული ელემენტის გარდა დავუმატოთ, ელემენტი რათა არ მოგვიჩიოს ცალკე აღწერა *div+p { }*

ყველა ელემენტის მონიშვნა *~* რომელიც მოცემულია სხვა ელემენტში, მაგალითად:
div ~ p { }

მაგალითი 2.41

```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
    background-color: grey;
}
</style>
</head>
<body>
<div>
    <p>ტექსტი div-ში</p>
    <p>ტექსტი div-ში.</p>
</div>
<p>ტექსტი div-ში არაა.</p>
<p>ტექსტი div-ში არაა.</p>
</body>
</html>
```

შედეგი

ტექსტი div-ში

ტექსტი div-ში.

ტექსტი div-ში არაა.

ტექსტი div-ში არაა.



ილუსტრირებისთვის გამოვიყენეთ სხვა კომბინატორები.

ფსევდო კლასები / Pseudo-classes

ფსევდო კლასებს ზოგადად შევხებით როდესაც განვიხილეთ *Link*-ები, ფსევდო კლასების საშუალებით შესაძლებელია, განისაზღვროს ელემენტები, მაგალითად, კონკრეტულად *<p>* ელემენტი, და მისი წარმოჩენა შესაძლებელია სხვადასხვა ფერით, როდესაც დავაჭერთ ან როდესაც მაუსს გადავატარებთ და ა.შ, ერთი სიტყვით, ვესტუმრეთ თუ არა მოცემულ ელემენტს.

***Link*-ების განხილვისას შევხებით ოთხ ელემენტს:**

a:link - ლინკი, რომელზეც არ დაგვიჭერია.

a:visited - ლინკი, რომელიც დავაჭირეთ.

a:hover - ლინკი, რომელზეც მაუსს გადავატარებთ.

a:active - ლინკი, რომელიც მომენტში აქტიურია.

ფსევდო კლასებში ეს ოთხივე პირობა უცვლელი რჩება და *Link*-ში განხილული ელემენტები ფსევდო კლასის იგივე ელემენტებია, სინტაქსის ილუსტრირებისათვის *Link*-ების აღარ შევხებით და უშუალოდ შევხებით ფსევდო კლასის ისეთ ელემენტს როგორცაა *:first-child*, რომელიც შინაარსობრივად განსაზღვრავს მშობელ ელემენტს და შემდეგ მისი შვილობილი ელემენტს, მოვახდინოთ სამი შემთხვევის ილუსტრირება.

შემთხვევა I

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child
{
    color: red;
}
</style>
</head>
<body>
<p>მოცემული ტექსტი წითლად წარმოჩნდება. </p>
<p>მოცემული ტექსტი წითლად არ წარმოჩნდება. </p>
</body>
</html>
```

შედეგი

მოცემული ტექსტი წითლად წარმოჩნდება.

მოცემული ტექსტი წითლად არ წარმოჩნდება.

შემთხვევა II

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```

p i:first-child
{
    color: red;
}
</style>
</head>
<body>
<p>ტექსტში სიტყვა, რომელიც <i>მოინიშნება, </i> <i>წარმოჩნდება </i>
წითლად. </p>
<p> ტექსტში სიტყვა, რომელიც <i>მოინიშნება, </i> <i> წარმოჩნდება
</i>წითლად.</p>
</body>
</html>

```

შედეგი

ტექსტში სიტყვა, რომელიც *მოინიშნება*, წარმოჩნდება წითლად.

ტექსტში სიტყვა, რომელიც *მოინიშნება*, წარმოჩნდება წითლად.

როგორც ვხედავთ, სკრიპტიდან პარაგრაფის პირველივე <i> ელემენტი შვილობილია, ხოლო დანარჩენი იგვე ელემენტზე ამავე პარაგრაფში i:first-child-აღარ მოქმედებს.

შემთხვევა III

```

<!DOCTYPE html>
<html>
<head>
<style>
p:first-child i
{
    color: red;
}
</style>
</head>
<body>
<p>ტექსტში სიტყვა, რომელიც <i>მოინიშნება </i>
<i> წარმოჩნდება </i>წითლად.
</p>
<p>ტექსტში სიტყვა, რომელიც <i>მოინიშნება </i>
<i> წარმოჩნდება </i>წითლად.
</p>
</body>
</html>

```

შედეგი

ტექსტში სიტყვა, რომელიც **მოინიშნება წარმოჩნდება** წითლად.
 ტექსტში სიტყვა, რომელიც **მოინიშნება წარმოჩნდება** წითლად.

ცხრილ 2.3-ში მოცემულია ყველა ფსევდო კლასის ელემენტები, რომელშიც არის CSS-ში, ზემოთ განხილული სამივე შემთხვევის პრინციპით ხდება დანარჩენი ელემენტების აღწერა.

ცხრილი 2.3 ფსევდო კლასების ელემენტები

სელექტორი	მოკლე აღწერა
<code>:active</code>	ნიშნავს აქტიურ ლინკებს
<code>:checked</code>	ამოწმებს <code><input></code> ელემენტებს <code>input:checked {}</code>
<code>:disabled</code>	მალავს ყველა <code><input></code> ელემენტს <code>input[type="text"]:disabled {}</code>
<code>:empty</code>	ნიშნავს ყველა <code><p></code> ელემენტს, რომელიც არაა შვილობილი <code>p:empty {}</code>
<code>:enabled</code>	ნებადართულია ყველა <code><input></code> ელემენტი <code>input[type="text"]:enabled {}</code>
<code>:first-child</code>	ნიშნავს ყველა <code><p></code> ელემენტის მშობლის შვილობილს <code>p:first-child {}</code>
<code>:first-of-type</code>	ნიშნავს ყველა <code><p></code> ელემენტს, სადაც <code><p></code> არის ელემენტი მშობელი <code>p:first-of-type {}</code>
<code>:focus</code>	ნიშნავს <code><input></code> ელემენტი არის ფოკუსირებული
<code>:hover</code>	ლინკებზე მაუსის გადატარება
<code>:in-range</code>	ნიშნავს <code><input></code> ელემენტი, სიდიდეებს <code>:in-range {}</code>
<code>:invalid</code>	ნიშნავს ყველა <code><input></code> ელემენტს, არაა ნებადართული
<code>:Lang(Language)</code>	ნიშნავს ყველა <code><p></code> , სადაც ნახესენებია <code>Lang</code> ატრიბუტი <code>p:lang(it) {}</code>
<code>:last-child</code>	ნიშნავს ყველა <code><p></code> ბოლო შვილობილს. <code>p:last-child {}</code>
<code>:last-of-type</code>	ნიშნავს მშობლის ბოლოს <code><p></code> ელემენტს <code>p:last-of-type {}</code>
<code>:link</code>	ნიშნავს ლინკებს რომელშიც არ ვართ ნამყოფი
<code>:not(selector)</code>	მონიშნულ ელემენტს არ მონიშნავს <code>:not(p) {}</code>
<code>:nth-child(n)</code>	ყველა მონიშნული <code><p></code> არის მშობლის მეორე შვილობილი ელემენტი <code>p:nth-child(3) {}</code>

<code>:nth-last-child(n)</code>	ნიშნავს <code><p></code> ელემენტს. რომელიც არის მეორე შვილობილი მშობლის, ითვლის ბოლო შვილობილი ელემენტიდან <code>:nth-last-child(2) {}</code>
<code>:nth-last-of-type(n)</code>	ითვლის ბოლო „შვილიდან“ და ნიშნავს ყველა იმ ელემენტს, რომელიც არის მეორე ელემენტი მშობელ ელემენტებში <code>p:nth-last-of-type{}</code>
<code>:nth-of-type(n)</code>	მშობელში ნიშნავს ყველა მეორე ელემენტს
<code>:only-of-type</code>	ნიშნავს მშობლის მხოლოდ ელემენტს
<code>:only-child</code>	ნიშნავს მხოლოდ შვილობილ ელემენტს
<code>:optional</code>	ნიშნავს <code><input></code> ელემენტის <code>required</code> ელემენტს.
<code>:out-of-range</code>	საზღვრის გარეთ <code>input:out-of-range {}</code>
<code>:read-only</code>	ნიშნავს <code><input></code> ელემენტის <code>readonly</code> ატრიბუტს <code>input:read-only{}</code>
<code>:read-write</code>	ნიშნავს <code><input></code> „ <code>readonly</code> “ ატრიბუტს <code>input:read-write {}</code>
<code>:required</code>	ნიშნავს <code><input></code> ელემენტს „ <code>required</code> “ ატრიბუტს <code>input:required{}</code>
<code>:root</code>	ნიშნავს <code>root</code> ელემენტს
<code>:target</code>	ნიშნავს აქტიურ URL-ს <code>:target {}</code>
<code>:valid</code>	<code><input></code> ნიშნავს ყველა ნებადართულ ელემენტს
<code>:visited</code>	ნიშნავს ყველა ელემენტს.

ფსევდო ელემენტები

ფსევდო ელემენტები ფსევდო კლასებისაგან განსხვავებით გამოირჩევა `::`-ორი წერტილით და შინაარსობრივად ფსევდო კლასისაგან განსხვავებით ფსევდო ელემენტი აღწერს ელემენტს :

ცხრილი 2.4 ფსევდო კლასები

სელექტორი	აღწერა
<code>::after</code>	შეიტანს ელემენტს კონტენტის შემდეგ.
<code>::before</code>	შეიტანს ელემენტს კონტენტამდე.
<code>::first-letter</code>	ნიშნავს პირველ სიმბოლოს ყველა ელემენტში, ელემენტი მოიცავს ისეთ თვისებებს, როგორიცაა: <code>font properties</code> <code>color properties</code>

	<i>background properties</i> <i>margin properties</i> <i>padding properties</i> <i>border properties</i> <i>text-decoration</i> <i>vertical-align</i> <i>text-transform</i> <i>line-height</i> <i>float</i> <i>clear</i>
::first-line	ნიშნავს პირველ სტრიქონს ელემენტში
::selection	ნიშნავს მომხარებლების მიერ.

განვიხილოთ თითოეული ელემენტის სინტაქსი

::after ელემენტის განხილვისას დავაკვირდეთ ელემენტში, მოთავსებული ტექსტი უცვლელი დარჩება

```
<!DOCTYPE html>
<html>
<head>
<style>
p::after
{
    content: " - ლაშა იაშვილი";
}
</style>
</head>
<body>
<p>სახელი და გვარია:</p>
<p></p>
</body>
</html>
```

შედეგი

სახელი და გვარია: - ლაშა იაშვილი

- ლაშა იაშვილი

ლაშა იაშვილი მაუსი უძრავად იქნება, ანალოგიურად იქნება *p::before*-ს დროსაც, ახლა კი ილუსტრირებისათვის მოვახდინოთ ორი ელემენტის *before* და *first-letter*.

```
<!DOCTYPE html>
<html>
```

```

<head>
<style>
p::before
{
    content: "მონიშნება -";
    background-color: grey;
    color: white;
    font-weight: bold;
}
p::first-letter {
    font-size: 200%;
    color: white;
}
</style>
</head>
<body>
<p>ტექსტი</p>
<p>ტექსტი</p>
</body>
</html>

```

შედეგი

მონიშნება -ტექსტი

მონიშნება -ტექსტი

ნავიგაცია

Web გვერდების პროექტირებისას დიდი მნიშვნელობა აქვს ნავიგაციას, მის ელემენტებს HTML-ს განხილვისას შევხებით, მაგრამ არ შევხებივართ სტილისტურად თუ როგორ გვინდა წარმოჩნდეს, მაგალითისათვის ავიღოთ სამი სხვადასხვა საიტი:

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
</style>
</head>
<body>

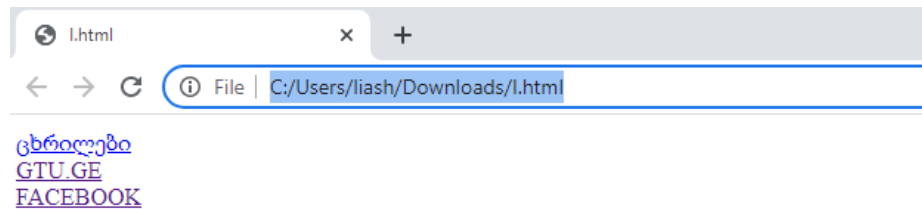
```

```

<ul>
  <li><a href="http://lector.gtu.ge">ცხრილები</a></li>
  <li><a href="http://gtu.ge">GTU.GE</a></li>
  <li><a href="http://facebook.com">FACEBOOK</a></li>
</ul>
</body>
</html>

```

შედეგი



რაც შეეხება `<style>` ტეგში მოთავსებულ ელემენტს, `list-style-type:none;` აღნიშნავს, რომ `bullet`-ს არ გამოჩნდეს ხოლო `margin:0;` და `padding:0;` აღნიშნავს, რომ ბრაუზერის `default` პარამეტრებს შლის.

მაგალით 2.42-ში წარმოდგენილია ვერტიკალური მენიუ, სადაც მაუსის გადატარების დროს, როგორც უკანა ფონი, ისე ფონტის ფერიც იცვლება.

მაგალითი 2.42

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 150px;
  background-color: #e5e5e5;
}

li a {
  display: block;
  color: #000;
  padding: 10px 16px;
  text-decoration: none;
}

/* მაუსის გადატარებისას უკანა ფონი, შავად ნაწერი ტექსტი კი თეთრი იქნება */
li a:hover
{

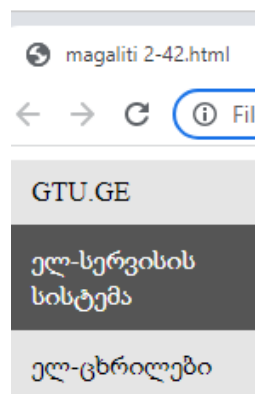
```

```

        background-color: #555;
        color: white;
    }
</style>
</head>
<body>
<ul> <li><a href="http://gtu.ge">GTU.GE</a></li>
      <li><a href="https://viciadmin.gtu.ge/#/login/login">ელ-სერვისის
სისტემა</a></li>
      <li><a href="https://leqtori.gtu.ge/">ელ-ცხრილები</a></li>
</ul>
</body>
</html>

```

შედეგი



ჰორიზონტალური მენიუს დროს, როგორც ვერტიკალურის მენიუს დროს, რაღაც ელემენტები უცვლელად რჩება, მაგრამ ძირითადი ელემენტები, რომელიც შედის **display:inline** ელემენტი, რომელიც ერთ სტრიქონზე აწყობს ყველა ელემენტს, ასევე **float** ელემენტი, რომელიც განსაზღვრავს ელემენტის ან ელემენტების პოზიციას, ასევე ხშირად გამოყოფენ ხოლმე მენიუს ერთმანეთისაგან **Border**-ებით.

მაგალითი 2.43

```

<!DOCTYPE html>
<html>
<head>
<style>
    ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
        overflow: hidden;
        background-color: #222;
    }

    li {
        float: left;
    }

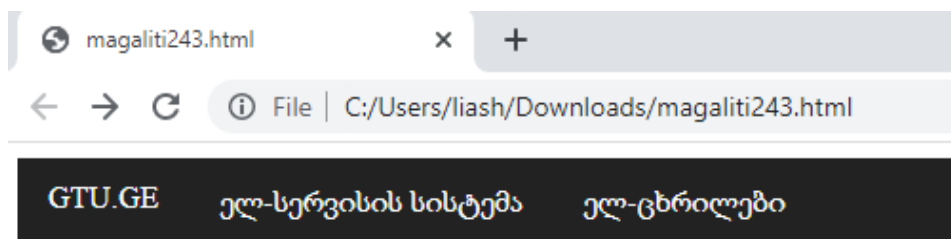
```

```

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 12px 16px;
    text-decoration: none;
}
li a:hover:not(.active) {
    background-color: rgb(179, 112, 112);
}
.active {
    background-color: #6c87a2;
}
</style>
</head>
<body>
<ul> <li><a href="http://gtu.ge">GTU.GE</a></li>
    <li><a href="https://viciadmin.gtu.ge/#/login/login">ელ-სერვისის
სისტემა</a></li>
    <li><a href="https://leqtori.gtu.ge/">ელ-ცხრილები</a></li>
</ul>
</body>
</html>

```

შედეგი



რაც შეეხება მოცემული სკრიპტის ხაზებს :

```

.active {
    background-color: #708090;
}

```

Active კლასი გამოყოფს ძირითად ან მთავარ მენიუს ჩანართს, ამ კლასის წაშლით უბრალოდ ყველა ჩანართის background-ი შავი ფერი იქნებოდა, ხოლო

```

li {
    float: left;
}

```

მაგივრად

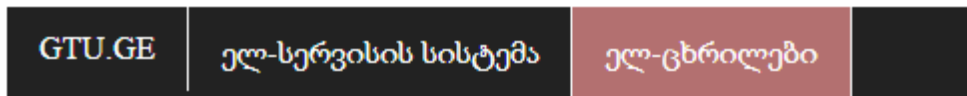
```

li {
    float: left;
}

```

```
border-right:1px solid #f1f1f1;
}
```

თუ დავწერთ, მენიუ გამოიყოფა, შედეგი გვექნება:



ჩამოსაშლელი მენიუში `javascripts`-ს საშუალებით აკეთებ, უფრო ლამაზად და ეფექტურად, მისი გვერდის ავლით სკრიპტი დავწეროთ:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title></title>
<style>

ul {
    list-style: none;
    padding: 0px;
    margin: 0px;
}
ul li {
    display: block;
    position: relative;
    float: left;
    border:1px solid #000
}
li ul {
    display: none;
}
ul li a
{
    display: block;
    background: #000;
    padding: 5px 10px 5px 10px;
    text-decoration: none;
    white-space: nowrap;
    color: #fff;
}
ul li a:hover {
    background: #f00;
}
li:hover ul {
    display: block;
    position: absolute;
}
li:hover li {
```

```

        float: none;
    }
    li:hover a {
        background: #f00;
    }
    li:hover li a:hover {
        background: #000;
    }
    #nav li ul li {
        border-top: 0px;
    }
</style>
</head>
<body>
<ul id="nav">
    <li><a href="#">ვებდეკლემენტ</a>
        <ul>
            <li><a href="#">HTML</a></li>
            <li><a href="#">CSS</a></li>
            <li><a href="#">JavaScript</a></li>
        </ul>
    </li>
    <li><a href="#">ვებძრავები</a>
        <ul>
            <li><a href="#">Joomla</a></li>
            <li><a href="#">Drupal</a></li>
            <li><a href="#">WordPress</a></li>
        </ul>
    </li>
</ul>
</html>

```

შედეგი

ვების ენები	ვებ ძრავები
HTML & CSS	
JavaScript	
PHP	

როგორც ვხედავთ, შედეგიდან როდესაც „ვების ენებს“-დავაჭერთ მაუსს ჩამოიშლება ქვე-განყოფილებები : HTML & CSS ,JAVASCRIPTS დასხვა. ანალოგიური იქნება ძრავების არჩევისას.

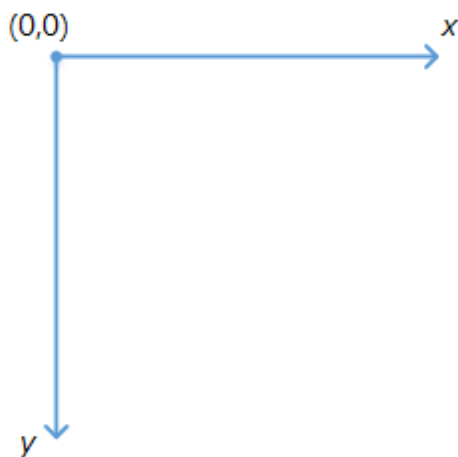
transforms

ელემენტების დაწერისას, თითოეული ელემენტის წარმოჩენისათვის როგორი დახრილობისა და მოქნილობის, რომ იყოს ელემენტი, შეიმუშავეს *transforms* ელემენტები, გარდაქმნის მათ ანიჭებს დამატებით სხვა თვისებებსაც, როგორც ორ განზომილებიანი ანუ 2D გარდაქმნა არსებობს, ასევე სამ განზომილებიანი-3D გარდაქმნები, არსებობს და თითოეული გარდაქმნას თავის თვისები აქვს, მისი სინტაქსია:

```
div {  
  -webkit-transform: scale(2.5);  
  -moz-transform: scale(2.5);  
  -o-transform: scale(2.5);  
  transform: scale(2.5);  
}
```

- ✓ **transform**-ში შედის ისეთი ელემენტები, როგორიცაა *translate*, *rotate*, *scale*, *matrix* და *skew*, მათი მეშვეობით შესაძლებელია შეიცვალოს ფორმა, ზომა და პოზიცია.
- ✓ *Translate*-საშუალებით ელემენტი იმოძრაავს, მარჯვნივ და მარცხნივ.
- ✓ *Rotate*-360° ატრიალებს ელემენტს.
- ✓ *Scale*-მოძრაობს x და y კოორდინატა სიბრტყის მიმართ, ელემენტის სიგრძეს და სიგანეს ზრდის.
- ✓ *Skew*- როგორც ცალკეული კოორდინატის მიმართ, ისე ორივე კოორდინატის სიბრტყეზე განისაზღვრება.
- ✓ *matrix()*-არის თავმოყრილი ყველა 2d transform-ის ელემენტი, ამ ელემენტს შეუძლია ექვსი პარამეტრის გათვალისწინება.

სურათი 2.12-ში ნაჩვენებია კოორდინატთა სიბრტყეზე ორ განზომილებიანობის, როგორც ვხედავთ, სურათიდან y სიბრტყე ქვემოთ არის დახრილი და არა პირიქით ზემოთ, ხოლო 2d transform-ის ელემენტები ძირითადად განისაზღვრება გრადუსებში და პიქსელებში-px



სურ 2.11 წარმოდგენილია CSS3 transform ელემენტების კოორდინატთა სიბრტყე

ცხრილი 2.4 2D transform-ის მეთოდები

<i>matrix(a,b,c,d,e,f)</i>	2-D გარდაქმნის matrix ექვს ელემენტს
<i>rotate(angle)</i>	2-D rotation კუთხის სიდიდით ატრიალებს ელემენტს
<i>scale(sx,sy)</i>	2-D scale მოქმედებს [sx,sy] არის სკალარული ვექტორი და აღიწერება ორი პარამეტრით
<i>scaleX(sx)</i>	[sx,1] არის სკალარული ვექტორი, სადაც sx არის პარამეტრი.
<i>scaleY(sy)</i>	[1,sy] არის სკალარული ვექტორი, სადაც sy არის პარამეტრი.
<i>skew(angleX,angleY)</i>	დამოკიდებულება X და Y კოორდინატებზე
<i>skewX(angle)</i>	skew გადახრა x-ღერძის მიმართ კუთხით.
<i>skewY(angle)</i>	skew გადახრა y-ღერძის მიმართ კუთხით.
<i>translate(tx,ty)</i>	2-D translation ვექტორით [tx,ty], სადაც tx არის პირველი ელემენტი გარდაქმნის, ხოლო ty მეორე სიდიდე.
<i>translateX(tx)</i>	X ღერძის გასწვრივ ხდება გარდაქმნა
<i>translateY(ty)</i>	Y ღერძის გასწვრივ ხდება გარდაქმნა.

თითოეული ელემენტის სინტაქსი:

```
transform: matrix(1.0, 2.0, 3.0, 4.0, 5.0, 6.0);
transform: translate(12px, 50%);
transform: translateX(2em);
transform: translateY(3in);
transform: scale(2, 0.5);
transform: scaleX(2);
transform: scaleY(0.5);
transform: rotate(0.5turn);
transform: skew(30deg, 20deg);
transform: skewX(30deg);
transform: skewY(1.07rad);
```

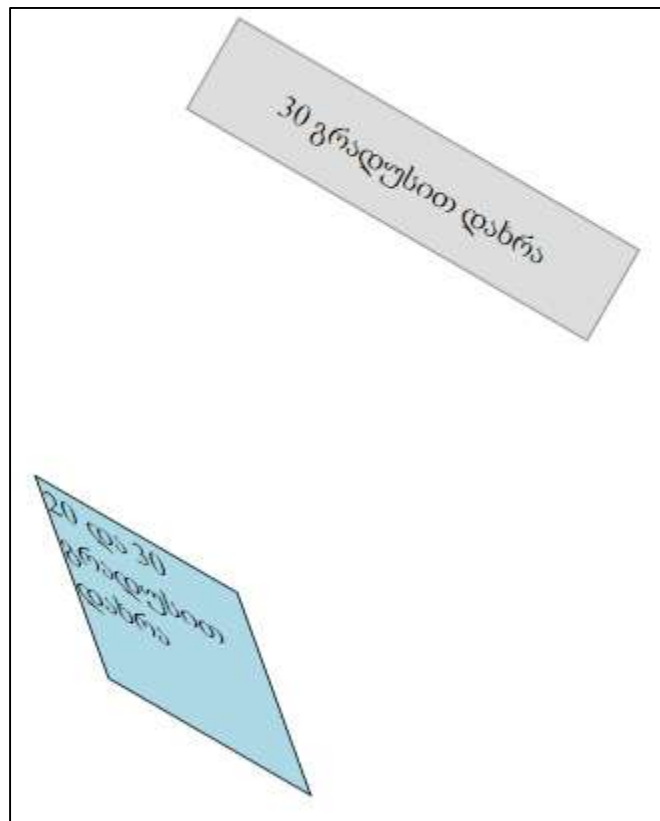
მოვახდინოთ ორი ელემენტის დემონსტრირება, rotate-ს შემთხვევაში, როდესაც დავდგებით, ელემენტი ამოტრიალდება, ხოლო skew-შემთხვევაში გვანახებს კუთხეების დახრილობს.

მაგალითი 2.44

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>
2d
</title>
<style>
#rotate
{
display: block;
width: 200px;
margin: 150px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: rotate(30deg);
-webkit-transform: rotate(30deg);
transform: rotate(30deg);
}
#rotate:hover
{
-moz-transform: rotate(120deg);
-webkit-transform: rotate(120deg);
}

#skew
{
width: 100px;
height: 100px;
margin: 50px;
background-color: lightblue;
border: 1px solid black;
transform: skew(20deg, 30deg);
}
</style>
</head>
<body>
<div id="rotate">30 გრადუსით დახრა
</div>
<div id="skew"> 20 და 30 გრადუსით დახრა
</div>
</body>
</html>
```

შედეგი



transform-origin

2D განზომილებაში შესაძლებელია ორი პარამეტრის გაწერა, სამ განზომილებისათვის კი სამი პარამეტრის, მისი სინტაქსის შემავლი ელემენტებია *top*, *left*, *right*, *bottom*, და *center*.

საწყისად 50% 50% ენიჭება, მისი მიზანია შეიცვალოს ელემენტის მდგომარეობა, სინტაქსში შედის სამი ელემენტი თავისი თვისებებით.

transform-origin: *x-ღერძი y-ღერძი z-ღერძი*

რა თქმა უდა, *z-ღერძი* არის სამ განზომილების შემთხვევაში.

ცხრილი 2.5 transform-origin თვისებები 2D და 3D განსაზღვრა

<i>x-ღერძი</i>	განსაზღვრავს <i>x</i> ღერძის მიმართ სიდიდებია <i>left</i> , <i>center</i> , <i>right</i> , <i>Length</i> და %
<i>y-ღერძი</i>	განსაზღვრავს <i>y</i> ღერძის მიმართ სიდიდებია: <i>top</i> , <i>center</i> , <i>bottom</i> , <i>Length</i> და %
<i>z-ღერძი</i>	განსაზღვრავს <i>z</i> ღერძის მიმართ სიდიდებია: <i>Length</i>

ელემენტი სხვადასხვა თვისებებით ჩაიწერება:

transform-origin: *top center*;

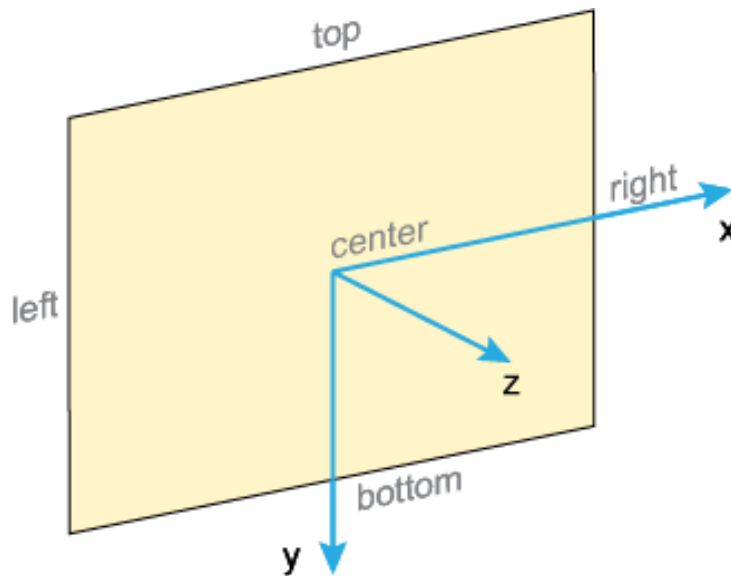
transform-origin: *100% 30px*;

transform-origin: *30px 60px*;

transform-origin: *center*;

transform-origin: *left*;

```
transform-origin: 30% 50% 0px;
transform-origin: bottom right;
```



სურ 2.12 ზე მოცემული პოზიციები *transform-origin*

ახლა მოვიყვანოთ მაგალითი, სადაც წარმოდგენილი იქნება ელემენტის ტრიალი, შევქმნათ ღილაკი რომელის დაჭერის შემდეგ ელემენტი დატრიალდება, ერთი კუთხით:

მაგალითი 2.45

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>transform-origin</title>
<style>
div
{
background: #fc0;
padding: 100px;
display: inline-block;
border: 1px solid #000;
}
div:hover
{
/* ზედა კუთხეში მოტრიალდება*/
-webkit-transform-origin: 100% 0;
-moz-transform-origin: 100% 0;
-o-transform-origin: 100% 0;
-ms-transform-origin: 100% 0;
transform-origin: 100% 0;
-webkit-transform: rotate(-30deg);
-moz-transform: rotate(-30deg);
-o-transform: rotate(-30deg);
```

```

    -ms-transform: rotate(-30deg);
    transform: rotate(-30deg);
}
</style>
</head>
<body>
    <div>მაუსის მიტანისას </div>
</body>
</html>
შედეგი

```



transform-style

3-D განზომილებაში წარმოადგენს ელემენტებს რაც სინტაქსის შემავალი ელემენტებია, მისი ორი ელემენტია ***preserve-3d***, რომელიც ჩნდება სამ განზომილებაში და ***flat***, თუ ***flat*** ია მონიშნული მაშინ, ელემენტები სამ განზომილებაში არ წარმოჩნდება, რისი დემონსტრირებაცაა მაგალითი 2.46

მაგალითი 2.46

```

<!doctype html>
<html>
<head>
<style type="text/css">

.container {
    position: relative;
    left:80px;
    top:60px;
    height: 200px;
    width: 200px;
    margin: 1px;
    border: 1px dotted firebrick;
    perspective: 250px; /* W3C */

```

```

    -webkit-perspective: 250px; /* Safari & Chrome */
    -moz-perspective: 250px; /* Firefox */
    -ms-perspective: 250px; /* Internet Explorer */
    -o-perspective: 250px; /* Opera */
}

.rotate {
padding:10px;
width: 200px;
height: 200px;
text-align:center;
background-color: gold;
opacity: 0.8;

/* W3C */
transform-style: preserve-3d;
animation: rotate 7s linear 0s infinite reverse none;

/* Safari & Chrome */
-webkit-transform-style: preserve-3d;
-webkit-animation: rotate 7s linear 0s infinite reverse none;

/* Firefox */
-moz-transform-style: preserve-3d;
-moz-animation: rotate 7s linear 0s infinite reverse none;

/* Opera */
-o-transform-style: preserve-3d;
-o-animation: rotate 7s linear 0s infinite reverse none;
}

.container:hover .rotate {
transform-style: flat; /* W3C */
-webkit-transform-style: flat; /* Safari & Chrome */
-moz-transform-style: flat; /* Firefox */
-ms-transform-style: flat; /* Internet Explorer */
-o-transform-style: flat; /* Opera */
}

/* W3C */
@keyframes rotate
{
from { transform: rotateX(0); }
to { transform: rotateX(360deg); }
}

/* Safari & Chrome */
@-webkit-keyframes rotate
{
from { -webkit-transform: rotateX(0); }
to { -webkit-transform: rotateX(360deg); }
}

```

```

}

/* Firefox */
@-moz-keyframes rotate
{
from { -moz-transform: rotateX(0); }
to   { -moz-transform: rotateX(360deg); }
}
/* Opera */
@-o-keyframes rotate
{
from { -o-transform: rotateX(0); }
to   { -o-transform: rotateX(360deg); }
}
.rotate > div
{
position: absolute;
top: 40px;
left: 40px;
width: 80px;
height: 50px;
padding: 10px;
box-sizing: border-box; /* W3C */
-webkit-box-sizing: border-box; /* Safari & Chrome */
-moz-box-sizing: border-box; /* Firefox */
-ms-box-sizing: border-box; /* Internet Explorer */
-o-box-sizing: border-box; /* Opera */
}

.rotate > :first-child
{
width:80px;
height:70px;
background-color: chartreuse;
transform: translateZ(-80px) rotateY(35deg); /* W3C */
-webkit-transform: translateZ(-80px) rotateY(35deg); /* Safari & Chrome */
-moz-transform: translateZ(-80px) rotateY(35deg); /* Firefox */
-ms-transform: translateZ(-80px) rotateY(35deg); /* Internet Explorer */
-o-transform: translateZ(-80px) rotateY(35deg); /* Opera */
}

.rotate > :last-child {
width:100px;
height:50px;
background-color: plum;

/* W3C */
transform: translateZ(50px) rotateX(50deg);
transform-origin: 50% top;

```



```

/* Safari & Chrome */
-webkit-transform: translateZ(50px) rotateX(50deg);
-webkit-transform-origin: 50% top;

/* Firefox */
-moz-transform: translateZ(50px) rotateX(50deg);
-moz-transform-origin: 50% top;

/* Internet Explorer */
-ms-transform: translateZ(50px) rotateX(50deg);
-ms-transform-origin: 50% top;

/* Opera */
-o-transform: translateZ(50px) rotateX(50deg);
-o-transform-origin: 50% top;
}
</style>
</head>
<body>
<p>დააკვირდით ელემენტების ტრიალის დროს <code>transform-style</code>
გადადის <code>preserve-3d</code> დან <code>flat</code>-ში.</p>
<div class="container">
  <div class="rotate">ელემენტების ტრიალი...
    <div>პირველი შვილობილი</div>
    <div>ბოლო შვილობილი</div>
  </div>
</div>
</body>
</html>

```



სკრიპტი არის ელემენტები რომლებიც არ განგვიჩივს, მათქვემოთ შევხებიდით ახლ კი დავტოვოთ, როგორც არს, რათა ეფექტურად დავაჩვენოთ როგორ ელემენტი.

სკრიპტი გადაწეროთ, შევინახოთ და გავუშვათ, დააკვირდით ელემენტების ტრიალის დროს *preserve-3d* დან *flat*-ში როგორ გადადის.

შედეგი



backface-visibility

ელემენტი განისაზღვრება ორი თვისებით *visible* და *hidden*, მათ შორის სხვაობა ის არის, რომ ელემენტი უკანა მხარე ჩანდეს თუ არა, ქვემოთ მოცემულია, შემთხვევა როდესაც ელემენტი ტრიალებს და უკანა მხარე მოჩანს, ხოლო თუ *hidden*-ს გავუწერთ, მხოლოდ წინა მხარე გამოჩნდება.

მაგალითი 2.47

```
<!doctype html>
<html>
<head>
<style type="text/css">
div.rotate
{
width:150px;
height:130px;
padding:20px;
text-align:center;
background:#e5e5e5;
-webkit-animation:rotate 5s linear 0s infinite normal none; /* Safari &
Chrome */
-webkit-backface-visibility: visible;
}
@-webkit-keyframes rotate {
from {
-webkit-transform: rotateY(0deg);
}
to {
-webkit-transform: rotateY(300deg);
}
}
}

</style>
</head>
<body>
<div class="rotate">ელემენტი ტრიალებს .</div>
</body>
</html>
```

შედეგი



perspective

განისაზღვრება სამ განზომილებიანი მოდელირებისას და განსაზღვრავს ელემენტს პიქსელებით ხედვის არეში მისი ორი თვისებაა: სიგრძე, რომელიც განისაზღვრება პიქსელებში და none, რომელიც Default-ია და იგივე 0-ია.

მაგალითი 2.48

```
<!doctype html>
<html>
<head>
<style>
div.container
{
padding:130px;
perspective: 200px;
    -webkit-perspective: 100px;
    -moz-perspective: 100px;
    -ms-perspective: 100px;
    -o-perspective: 100px;
}
div.rotate
{
    width:50px;
    height:30px;
    padding:50px;
    text-align:center;
    background:grey;
    transform: rotateX(30deg);
    -webkit-transform: rotateX(30deg);
    -moz-transform: rotateX(30deg);
    -ms-transform: rotateX(30deg);
    -o-transform: rotateX(30deg);
}
</style>
</head>
<body>
    <div class="container">
        <div class="rotate"> ელემენტი..
    </div> </div>
</body>
</html>
```

შედეგი



უნდა აღნიშნოს, რომ ეს ელემენტები, როგორც განზომილებისათვის ასევე სამ განზომილებისათვის არის გათვალისწინებული.

3D Transform მეთოდები

ცხრილი 2.6 3D Transform მეთოდები

ფუნქცია	აღწერა
<i>matrix3d</i> (<i>n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n</i>)	განსაზღვრავს 3D გარდამქმნებისათვის და ამ შემთხვევაში კი 4x4 მატრიცით განისაზღვრება და იწერება 16 სიმბოლო.
<i>translate3d(x,y,z)</i>	განსაზღვრავს 3D translation
<i>translateX(x)</i>	განსაზღვრავს ელემენტს X- ღერძის მიმართ
<i>translateY(y)</i>	განსაზღვრავს ელემენტს Y- ღერძის მიმართ
<i>translateZ(z)</i>	განსაზღვრავს ელემენტს Z- ღერძის მიმართ
<i>scale3d(x,y,z)</i>	განსაზღვრავს 3D scale გარდაქმნას
<i>scaleX(x)</i>	განსაზღვრავს 3D scale გარდაქმნას x ღერძის მიმართ
<i>scaleY(y)</i>	განსაზღვრავს 3D scale გარდაქმნას Y ღერძის მიმართ
<i>scaleZ(z)</i>	განსაზღვრავს 3D scale გარდაქმნას z ღერძის მიმართ
<i>rotate3d(x,y,z,angle)</i>	სამ განზომილებიანი როტაცია
<i>rotateX(angle)</i>	სამ განზომილებიანი გარდაქმნა X ღერძის მიმართ
<i>rotateY(angle)</i>	სამ განზომილებიანი, გარდაქმნა Y ღერძის მიმართ
<i>rotateZ(angle)</i>	როტაცია Z ღერძის მიმართ
<i>perspective(n)</i>	ელემენტის ხედვის არე სამ განზომილებაში

წესები/Rules

წესები- ასე უწოდებენ იმ ელემენტს, რომლის საშუალებით შესაძლებელია განისაზღვროს, ჩაინერგოს ან დაიწეროს CSS სკრიპტში.

@charset- საშუალებით შესაძლებელია განისაზღვროს უნიკოდი, დეკოდირების მეთოდიკა, როგორც ცნობილია, ძირითად გამოიყენება არა ANSI დეკოდირებისას, აქტიურია ქართული უნიკოდის წასაკითხად, ქართული ენის დეკოდირებისათვის გამოიყენება UTF-8, მისი სინტაქსი სხვადასხვა დეკოდირებისას ასე ჩაიწერება:

`@charset "UTF-8";`

`@charset 'iso-8859-15';`

@import- გამოიყენება როდესაც რაღაცის შეტანა გვსურს CSS- სკრიპტში, მაგალითად, სხვა CSS ფაილი, სადაც სხვა დიზიანია აღწერილი.

`@import url("/css/mtavari.css");`

`@import "style.css";`

@media- მედიას ტიპი, სადაც შეიძლება განისაზღვროს სიტყვიდან გამომდინარე, როგორც სტილისტური საკითხები, ისე მოწყობილობებთან მიმართებით, ბეჭდვის, რეზულუციის და სხვ.

media type...

{

წესის გაწერა

}

მედია ტეგის განხილვისას ზოგადად, შევხებით, ახლა კი ცხრილ 2.6 და ცხრილ 2.7 წარმოდგენილია ყველა ელემენტი:

ცხრილი 2.6 წარმოდგენლია ზოგადი სიდიდეები

სიდიდეები	აღწერა
<i>all</i>	ყველა მოწყობილობაზე
<i>aural</i>	საუბრის სინქრონიზაციისას
<i>braille</i>	Braille მოწყობილობა
<i>handheld</i>	პატარა ეკრანი, პატარა გაფართოება
<i>projection</i>	პროექტორი
<i>print</i>	ბეჭდის ნახვა/ბეჭდვის გვერდები
<i>screen</i>	კომპიუტერის ეკრანები
<i>tty</i>	ტერმინალები, პორტატული მოწყობილობები შეზღუდული ჩვენების შესაძლებლობებით. მათთვის, გამოიყენება როგორც <i>pixel</i> ერთეული.
<i>tv</i>	ტელევიზორის ან მსგავსი მოწყობილობები.

ცხრილი 2.7 @media თვისებები

სიდიდეები	აღწერა
<i>aspect-ratio</i>	ეკრანის სიგანის/სიმაღლის გაფართოება
<i>color</i>	თითოეული ბიტი თითოეულ ფერზე ეკრანზე/ქაღალდზე

<i>color-index</i>	ეკრანზე ფერების რიცხვი
<i>device-aspect-ratio</i>	device-width/device-height გაფართოება, ეკრანის/ქაღალდის მაგალითი: media="screen and (aspect-ratio:16/9)"
<i>device-height</i>	ეკრანის/ქაღალდის გაფართოება სიმაღლე შეგვიძლია გამოვიყენოთ მაგალითად: media="screen and (device-height:500px)"
<i>device-width</i>	მოწყობილობის სიგანე რეზოლუცია
<i>grid</i>	ინფორმაციის გამოტა, grid ან bitmap. სიდიდეები "1" grid-სთვის, "0" სხვა შემთხვევა. მაგალითი: media="handheld და (grid:1)"
<i>height</i>	სიმაღლე
<i>max-aspect-ratio</i>	ეკრანის მაქსიმალური გაფართოება.
<i>max-color</i>	მაქსიმალური ფერების რაოდენობა, რომელიც გამოდის მოწყობილობაზე.
<i>max-color-index</i>	მაქსიმალური ფერების რიცხვი, რომელიც შეუძლია ეკრანზე გამოტანოს.
<i>max-device-aspect-ratio</i>	ეკრანის მაქსიმალური გაფართოება.
<i>max-device-height</i>	მაქსიმალური სიმაღლე, როგორიცაა კომპიუტერის ეკრანი.
<i>max-device-width</i>	მაქსიმალური სიგანე, როგორიცაა კომპიუტერის ეკრანი.
<i>max-height</i>	ბრაუზერის მაქსიმალური არეალი .
<i>max-monochrome</i>	მაქსიმალური მონოქრომზე თითოეული ბიტი თითოეულ პიქსელზე.
<i>max-resolution</i>	მაქსიმალური რეზოლუცია.
<i>max-width</i>	მაქსიმალური სიგანე ბრაუზერის ეკრანზე.
<i>min-aspect-ratio</i>	მინიმალური გაფართოება.
<i>min-color</i>	მინიმალური ფერების რაოდენობა, რაც მოწყობილობაზე გამოდის.
<i>min-color-index</i>	მინიმალური ფერების რაოდენობა.
<i>min-device-aspect-ratio</i>	მინიმალური გაფართოება.
<i>min-device-width</i>	მინიმალური გაფართოება, როგორიცაა კომპიუტერის ეკრანი.
<i>min-device-height</i>	მოწყობილობის მინიმალური სიმაღლე, მაგალითად კომპიუტერის ეკრანის მინიმალური სიმაღლე.
<i>min-height</i>	მინიმალური სიმაღლე.
<i>min-monochrome</i>	მინიმალური მონოქრომზე თითოეული ბიტი თითოეულ პიქსელზე.
<i>min-resolution</i>	მოწყობილობის მინიმალური რეზოლუცია, dpi ან dpcm

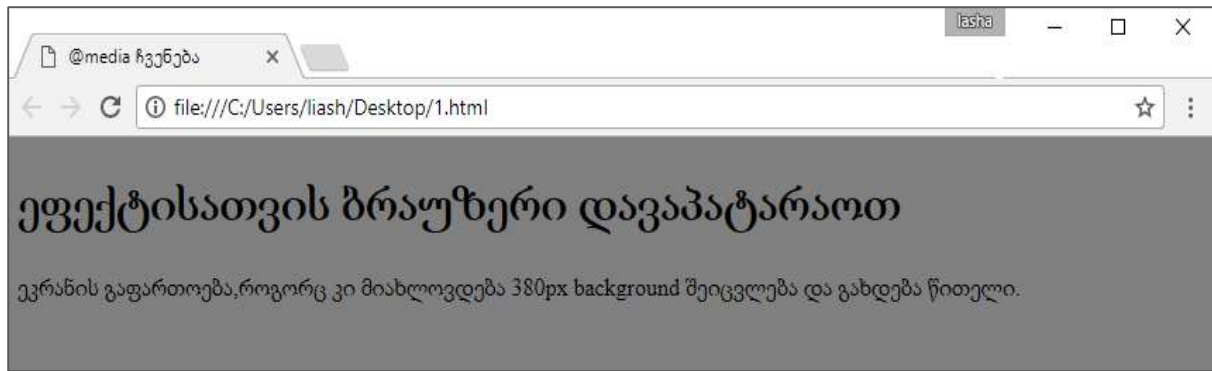
min-width	მინიმალური სიგანე
monochrome	მონოქრომზე თითოეული ბიტი თითოეულ პიქსელზე მაგალითი: <code>media="screen and (monochrome:2)"</code>
orientation	ორიენტაცია <code>landscape</code> ან <code>portrait</code> მოდიფიკაცია
overflow-block	ინფორმაცია, თუ როგორ გადმოეცემა ღერძების გასწვრივ
resolution	რეზოლუცია მოწყობილობაზე, როგორიცაა <code>dpi</code> ან <code>dpcm</code>
scan	საკნირების მეთოდი, შესაძლო სიდიდეები „progressive“ და „interlace“. მაგალითი: <code>media="tv ან (scan:interlace)"</code>
update-frequency	რამდენად ხშირად გამოდის მოწყობილობაზე კონტენტი
width	viewport სიგანე

მოვახდინოთ მაგალითი 2.49 ჩვენება `@media` ელემენტის:

მაგალითი 2.49

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> @media ჩვენება </title>
<style>
body {
    background-color: red;
}
@media screen and (min-width: 380px)
{
    body {
        background-color: grey;
    }
}
</style>
</head>
<body>
<h1> ევექტისათვის ბრაუზერი დავაპატარაოთ</h1>
<p> ევრანის გაფართოება, როგორც კი მიახლოვდება 380px background, შეიცვლება
და გახდება წითელი.
</p>
</body>
</html>
```

შედეგი



@page- განისაზღვრება გვერდები, გვერდის არეალები და საზღვრები, გვერდის ზომები, გვერდი პარამეტრებში შედის ისეთი პარამეტრები, როგორიცაა *auto*, *landscape*, *portrait* და სიგრძე, რომელიც განსაზღვრავს გვერდის ზომას.

```
<style>
  @page {
    size: auto;
    margin: 10%;
  }
</style>
ან
<style type="text/css">
  <!--
  @page { size : portrait }
  @page rotated { size : Landscape }
  table { page : rotated }
  -->
</style>
```

თვითონ @page სინტაქსია:

```
@page [ { :left | :right | :first } ] {
    margin ruleset }
```

თითოეული თვისების ჩვენება

```
@page {
  margin: 2.5cm; /* ყველა გვერდისათვის */
}
```

```
@page :left {
  margin-left: 15cm; /* მარცხნივ */
}
```

```
@page :right {
  margin-right: 15cm; /* მარჯვნივ */
}
```

```
@page :first {
```



```
margin-top: 10cm; /* ზემოთ პირველი გვერდის */
}
```

@font-face-ვისაუბრეთ ზემოთ (იხ. მაგალით 2.29), მისი დახმარებით შეგვიძლია სასურველი ფონტების ჩამატება.

```
@font-face {
  font-family: "სასურველი ფონტი";
  src: url("ფონტის მისამართი");
}

h1 {
  font-family: "ფონტი", sans-serif;
}
```

@namespace -სახელსივრცე განისაზღვრება XML სახელსივრცე CSS დოკუმენტისათვის, მისდევს რა **@import** და **@charset** დირექტივებს, ამავე დროს წინ უძღვის ყველა წეს და რიგ შემთხვევაში განისაზღვრება პრეფიქსით, მათი სინტაქსია:

```
@namespace url(XML-namespace-URL);
ან
@namespace prefix url(XML-namespace-URL);
```

@keyframes-ელემენტარის სპეციალური ელემენტის ანიმაციას, სადაც შესაძლებელია განისაზღვროს და შეიცვალოს რამდენჯერმე ელემენტი, მისი თვისებებია *from* და *to*, და სიდიდეები, რომელიც განისაზღვრება არის 0%-დან (სადაც ანიმაცია იწყება) და 100%-მდე (სადაც ანიმაცია მთავრდება), ასევე რამდენჯერმე შეიძლება გაიწელოს.

animation-ელემენტისას განისაზღვრება **@keyframe** ელემენტი, ამიტომაც ახლა კოდს თუ ავკრეფთ ელემენტი უმოძრაოდ იქნება, უნდა ვახსენეთ ისიც, რომ **animation** სახელი და **@keyframe** ცვლადის სახელი ერთი და იგივე უნდა იყოს.

სინტაქსი:

```
@keyframes სახელი:{ keyframes-სელექტორი{css-სტილი;}};
```

ჩვენება

```
/* Chrome, Safari, Opera */
@-webkit-keyframes ელემენტი
{
  0% {top: 0px; background: red; width: 100px;}
  100% {top: 200px; background: yellow; width: 300px;}
}

/* სტანდარტული სინტაქსი */
@keyframes ელემენტი
{
  0% {top: 0px; background: red; width: 100px;}
  100% {top: 200px; background: yellow; width: 300px;}
}
```

!important- ელემენტი, რომელიც პირველ რიგში უნდა შესრულდეს, თუ ფერს, ფონტს ამ სხვა ელემენტიდან წერია, პირველ რიგში, მას შეასრულებს.

მაგალითად

```
<html>
  <head>
    <meta charset="utf-8">
    <style>
      p { color: green !important; }
      p { color: black; }
    </style>
  </head>
  <body>
    <p>გამწვანდება</p>
  </body>
</html>
```

transitions

transitions მისი თვისებები, ოთხნაირი შემოკლებით განისაზღვრება: *transition-property*, *transition-duration*, *transition-timing-function* და *transition-delay*.

საწყისი დროის ინტერვალი 0 ია, ვებ დაპროგრამირებაში დროითი ინტერვალი როგორც დადებით ისე უარყოფითი შეიძლება იყოს, თუ მივუთითებთ დროითი ინტერვალში დადებით რიცხვს, *transition* ეფექტი დაიწყება შეფერხებით და ნელა, ხოლო უარყოფილი დროის ინტერვალის დროს მყისიერად იმოქმედებს ელემენტზე. სინტაქსის შემავალი თვისებებია:

property, *duration*, *timing-function* და *delay*

მაგალითი 2.50

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style>
div.text
{
background: #eee;
color:#333;
text-decoration:none;
padding:6px 11px;
border:2px solid #333;
border-radius:5px;
-moz-border-radius:5px;
-webkit-border-radius:6px;
}
.text:hover
{
background:#333;
```

```

color:#eee;
}
</style>
<title> transitions</title>
</head>

<body>
<div class="text">text </div>
</body>
</html>

```

შემდეგი



მაგალით 2.50-ში მოცემულია ჩვეულებრივი ელემენტი, რომელიც დაკლიკების მერე ფერს იცვლის, ახლა CSS-ში ჩავამატოთ შემდეგი სტრიქონები:

```

-webkit-transition-property:background color, color;
-webkit-transition-duration:600ms;
-webkit-transition-timing-function:ease-in-out;

```

სრული სკრიპტი

```

<style>
div.text
{
background: #eee;
color:#333;
text-decoration:none;
padding:6px 11px;
border:2px solid #333;
border-radius:5px;
-moz-border-radius:5px;
-webkit-border-radius:6px;
-webkit-transition-property:background color, color;
-webkit-transition-duration:600ms;
-webkit-transition-timing-function:ease-in-out;
}
.text:hover
{
background:#333;
color:#eee;
}
</style>

```

დავინახავთ, როგორ შეიცვლება მოცემული ელემენტი დროის ინტერვალში, რა თქმა უნდა უცხო ელემენტი მათი თვისებებს ქვემოთ გავცნობით.

transition-delay

transition ეს ელემენტი განსაზღვრავს დროის ინტერვალს, მის თვისებაში შემადგენილი ნაწილი დროა, რომელიც ელოდება და შემდეგ იწყებს ეფექტს.

transition-delay: 10s

*-webkit-transition-delay: 10s; /*Safari და Chrome*/*

*-o-transition-delay: 10s /*Opera*/*

transition-duration

დროითი ინტერვალი რომელიც განსაზღვრავს ელემენტს, თუ რა დრო, ჭირდება ეფექტის შესასრულებლად, მის თვისებაში შედის დროითი სიდიდეები, წამები, მილი წამები და ა.შ.

transition-duration: 15s

-webkit-transition-duration: 15s;

-o-transition-duration: 15s

inline-block

float ელემენტს გამოყენებდნენ ხოლმე, სანამ *inline-block* ელემენტი შემუშავდებოდა, მისი საშაულებით შესაძლებელია შეიქმნას *box*-ები, რომელებსაც აქვთ სიგრძე და სიგანე.

მაგალითი 2.40

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>inline block </title>
```

```
<style>
```

```
.inlineblock
```

```
{ width:250px;
```

```
height:50px;
```

```
display:inline-block;
```

```
border: 2px solid #3F83AF;
```

```
text-align:center;
```

```
margin: 55px;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="inlineblock">inline-block-ჩვენება </div>
```

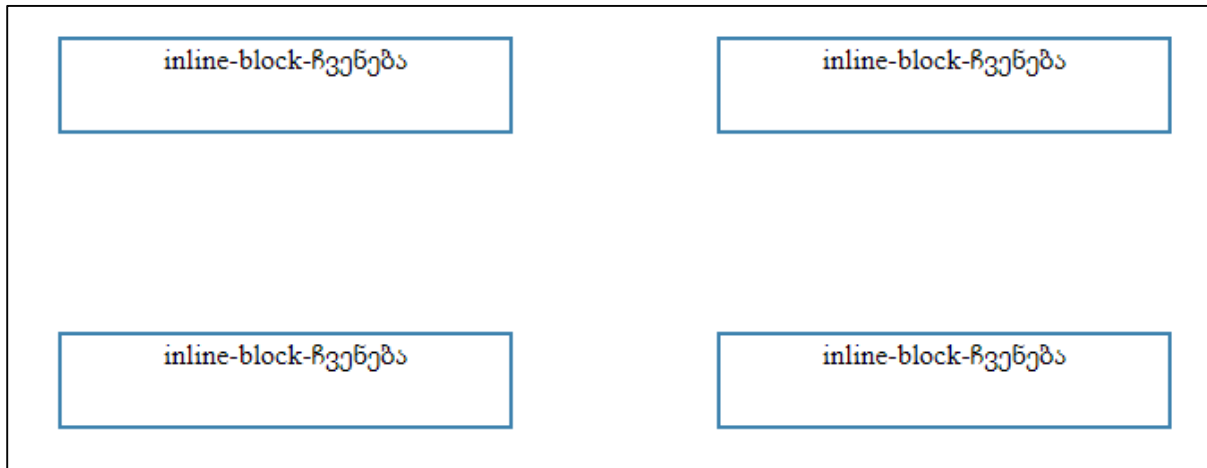
```
<div class="inlineblock">inline-block-ჩვენება </div>
```

```
<div class="inlineblock">inline-block-ჩვენება </div>
```

```
<div class="inlineblock">inline-block-ჩვენება</div>
```

```
</body>
</html>
```

შედეგი



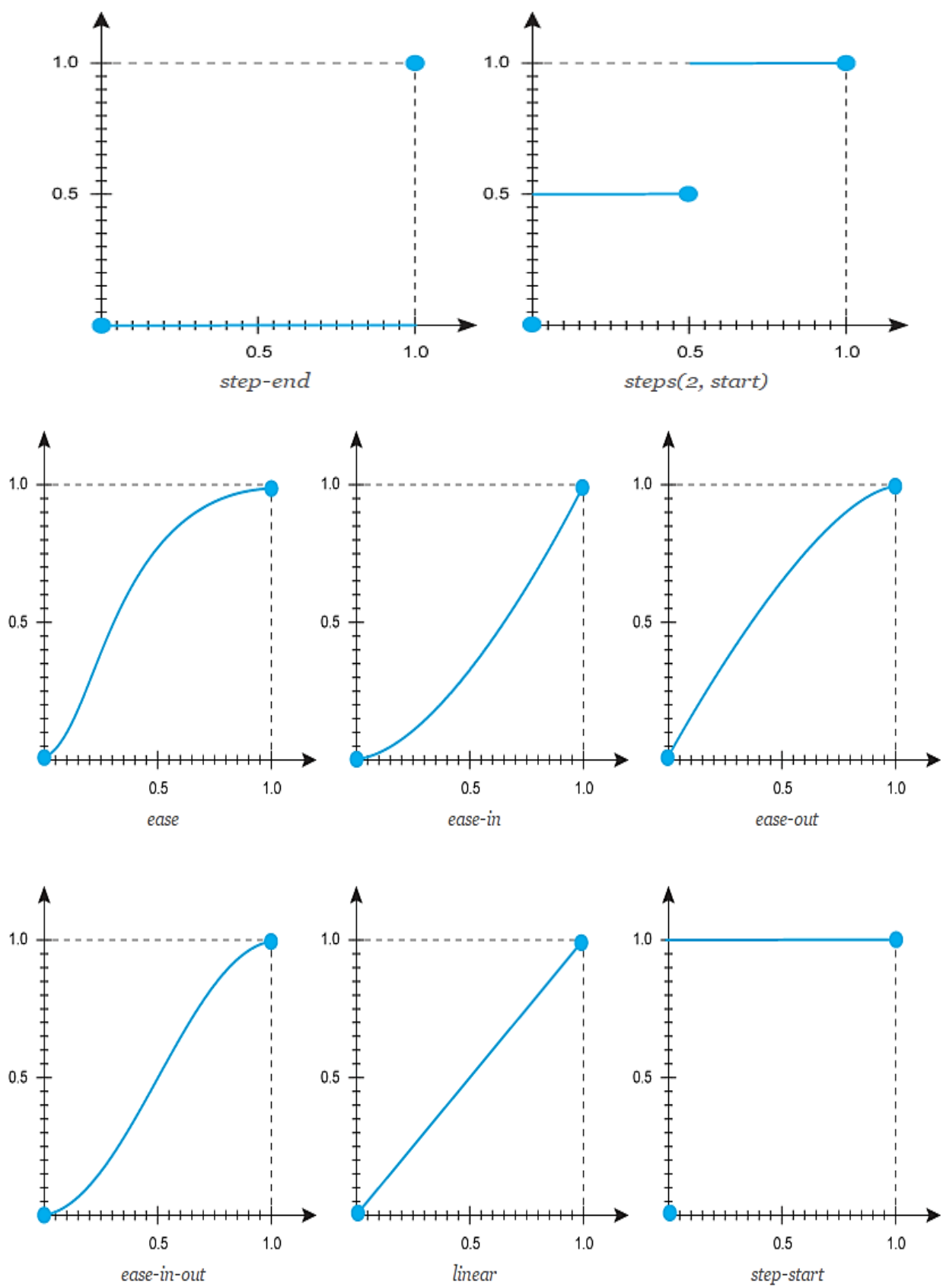
transition-property

მოიცავს სახელის ყველა თვისებას, რაც CSS თვისებებში შედის, მის თვისებაშია none, რომელიც ყველა transition ეფექტს უგულებელყოფს, all-რომელიც ყველა თვისებას არენდერებს და თვითონ თვისებები (CSS-ს ზოგადად, ყველა თვისება).

transition-timing-function

transition ეფექტის სიჩქარეს განსაზღვრავს, რომელიც დროის ინტერვალში შეიძლება შემცირდეს ან ეფექტის სიჩქარე მოემატოს, იგი მათემატიკური ფუნქციაა (Bézier curve-ხშირად გამოიყენება კომპიუტერულ გრაფიკაში), რომელიც დროის ინტერვალში იცვლება და საწყისი წერტილები არის კოორდინატთა სიბრტყეზე 0.0, 0.0, ხოლო საბოლოო - 1.0, 1.0, მის თვისებებში შედის: Linear, ease, ease-in, ease-out, ease-in-out და cubic-bezier(n,n,n,n);

- ✓ ease-თავიდან ანიმაცია ნელდება შემდეგ აჩქარდება და ბოლოს ისევ შენელდება, მოცემული თვისება ანალოგია cubic-bezier(0.25,0.1,0.25,1).
- ✓ ease-in-თავიდან ანიმაცია ნელა იწყება, შემდეგ ჩქარდება, მისი ანალოგია cubic-bezier(0.42,0,1,1).
- ✓ ease-out-ანიმაცია იწყება სწრაფად, ხოლო ბოლოს ნელდება, ანალოგია cubic-bezier(0,0,0.58,1).
- ✓ ease-in-out-ანიმაცია იწყება და მთავრდება ნელა, ანალოგია cubic-bezier(0.42,0,0.58,1).
- ✓ Linear-ერთსა და იმავე სიჩქარით იწყება და მთავრდება ანიმაცია.
- ✓ step-start-როგორც ასეთი, ანიმაცია არ არსებობს, ენიჭება სიდიდე, ეს არის დროითი ინტერვალი
- ✓ step-end-ანიმაცია არ არსებობს.
- ✓ step განსაზღვრავს ბიჯს, თუ რამდენი ერთეულის მერე შეიძლება, რომ დაიძრას, ხოლო cubic-bezier განსაზღვრავს მოძრაობის ფუნქციას.



სურ. 2.13 ზე მოცემულია სხვადასხვა მრუდი

მაგალითი 2.51

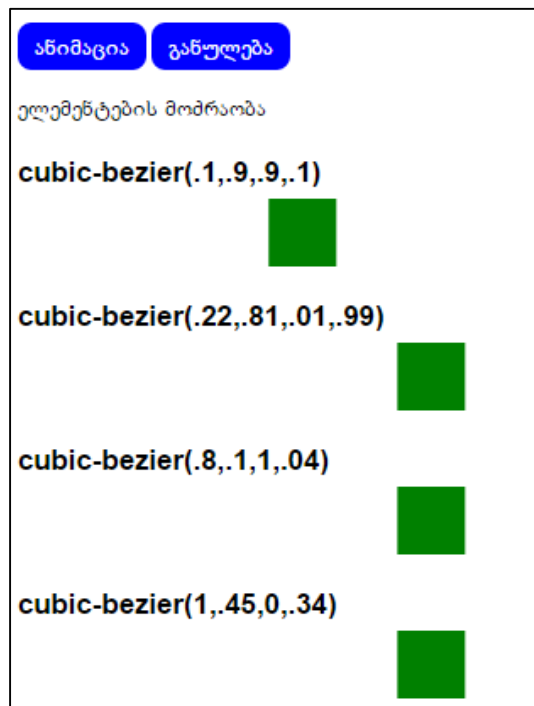
```
<!DOCTYPE html>
<html Lang="ka">
  <head>
    <meta charset="UTF-8">
    <title>ანიმაცია</title>
    <style>
      .ელემენტი
    {
      background: green;
      width: 40px;
      height: 40px;
      -webkit-transition-duration: 2s;
      transition-duration: 2s;
    }
    .ერთი .ელემენტი
    {
      -webkit-transition-timing-function: cubic-bezier(.1,.9,.9,.1);
      transition-timing-function: cubic-bezier(.1,.9,.9,.1);
    }
    .ორი.ელემენტი
    {
      -webkit-transition-timing-function: cubic-bezier(.22,.81,.01,.99);
      transition-timing-function: cubic-bezier(.22,.81,.01,.99);
    }
    .სამი.ელემენტი
    {
      -webkit-transition-timing-function: cubic-bezier(.8,.1,1,.04);
      transition-timing-function: cubic-bezier(.8,.1,1,.04);
    }
    .ოთხი.ელემენტი
    {
      -webkit-transition-timing-function: cubic-bezier(1,.45,0,.34);
      transition-timing-function: cubic-bezier(1,.45,0,.34);
    }
    #მოძრაობა:target .ელემენტი
    {
      -webkit-transform: translate(300px);
      transform: translate(300px);
    }
    body
    {
      font: .7em Arial;
    }
    h2 {
      margin: 20px 0 5px;
    }
    a {
      padding: 5px 10px;
```

```

background: blue;
color: #fff;
border-radius: 8px;
text-decoration: none;
font-size: 1.2em;
display: inline-block;
margin-top: 10px;
}
a:hover
{
  opacity: 0.6;
}
p
{
  font-size: 1.2em;
}
</style>
</head>
<body>
  <div id="მოდრობა">
    <a href="#მოდრობა">ანიმაცია</a> <a href="#reset">განულება</a>
    <p>ელემენტების მოძრაობა</p>
      <div class="ერთი">
        <h2> cubic-bezier(.1,.9,.9,.1)</h2>
        <div class="ელემენტი">
        </div>
      </div>
      <div class="ორი">
        <h2>cubic-bezier(.22,.81,.01,.99)</h2>
        <div class="ელემენტი">
        </div>
      </div>
      <div class="სამი">
        <h2>cubic-bezier(.8,.1,1,.04)</h2>
        <div class="ელემენტი">
        </div>
      </div>
      <div class="ოთხი">
        <h2>cubic-bezier(1,.45,0,.34)</h2>
        <div class="ელემენტი">
        </div>
      </div>
    </div>
  </body>
</html>

```


შედეგი



როდესაც დავაწვებით „ანიმაციის“ ღილაკს, კუბიკები დაიწყებენ მოძრაობას. ხოლო განულების დროს, დაუბრუნდება საწყის მნიშვნელობას.

ახლა დავწეროთ step- ფუნქცია, რომელიც 6 ბიჯით დაძრავს ელემენტს.

მაგალითი 2.53

```
<!DOCTYPE html>
<html Lang="ka">
  <head>
    <meta charset="UTF-8">
    <title>ანიმაცია</title>
    <style>
      .ელემენტი {
        background: grey;
        width: 100px;
        height: 100px;

        margin-top: 30px;
        -webkit-transition: 6s steps(6);
        transition: 6s steps(6);
      }

      #მოძრაობა:target .ელემენტი {
        -webkit-transform: translate(400px);
        transform: translate(400px);
      }

      body {
```

```

    font: .7em Arial;
}

a {
    padding: 5px 10px;
    background: blue;
    color: #fff;
    border-radius: 8px;
    text-decoration: none;
    font-size: 1.2em;
    margin-top: 10px;
    display: inline-block;
}

a:hover {
    opacity: 0.6;
}

```

```

p {
    font-size: 1.2em;
}

```

```

</style>
</head>
<body> <div id="მოდრაობა">

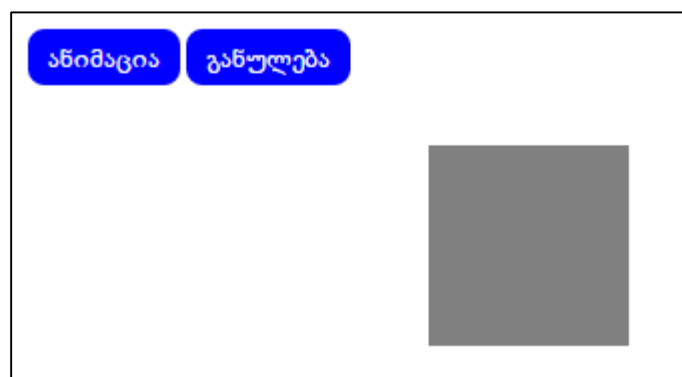
    <a href="#მოდრაობა">ანიმაცია</a> <a href="#reset">განულება</a>

    <div class="ერთი">
        <div class="ელემენტი">
            </div>
        </div>

</div>
</body>
</html>

```

შედეგი



animations

ანიმაციები შედგება ორი კომპონენტისაგან, სტილი, სადაც აღწერილია CSS ანიმაცია და კომპლექტი *keyframes*, რათა მიუთითოს დაწყებისა და დასრულების პარამეტრები, მისი საშუალებით შესაძლებელია დროითი ინტერვალში შეიცვალოს ელემენტის სიჩქარე და ეფექტურობა, როგორც ზემოთ აღვნიშნეთ *keyframes* გარჩევისას, *animation* სახელი და *keyframes* სახელი ერთი და იგივე უნდა იყოს.

ანიმაციის აღწერისას გასათვალისწინებელია ელემენტის ის ქვეთვისებება, რომლების პასუხისმგებელია: დროზე, შეფერხებზე და სხვა თვისებებზე, რომელიც *@keyframes* ეხმარება, რომ ანიმაცია, ანიმირებული იყოს.

animation თვისებებია:

- ✓ *animation-duration*- დროითი ხანგრძლივობა, რომელიც უნდა შესრულდეს და დასრულდეს, მის თვისებებში შედის დროითი სიდიდე, რომელიც უნდა დაიწეროს.

```
div {  
    -webkit-animation-duration: 10s;  
    animation-duration: 10s;  
}
```

- ✓ *animation-timing-function*- დროის ფუნქცია, დამოკიდებულია დროზე, რომელიც დიზაინის პარამეტრები იცვლება დროით ინტერვალში, მის თვისებებში შედის: *linear*, *ease*, *ease-in*, *ease-out*, *ease-in-out*, *step-start*, *step-end*, *steps(int,start/end)*, *cubic-bezier(n,n,n,n)* და , როგორც ზემოთ ვიმსჯელეთ ესენი მათემატიური ფუნქციებია *Cubic Bezier* მრუდე და ეწოდება .

```
div {  
    animation-timing-function: ease; /* ყველა თვისების ჩვენება */  
    animation-timing-function: ease-in;  
    animation-timing-function: ease-out;  
    animation-timing-function: ease-in-out;  
    animation-timing-function: linear;  
    animation-timing-function: step-start;  
    animation-timing-function: step-end;  
}
```

- ✓ *animation-delay*-ელემენტი განსაზღვრავს როცა ანიმაცია დაიწყება ან რაიმე ნაწილი ანიმაციის, დაიწყება დროითი ინტერვალი, რომელიც ნიშნავს იმას, რომ ანიმაცია დაიწყება მას მერე რაც დავაჭერთ, ანუ 5 წამი თუ გვაქვს მითითებული დაჭერის შემდეგ 5 წამის მერე დაიწყება ანიმაცია, ხოლო დროითი ინტერვალი თუ უარყოფითია, მაშინ მყისიერად შესრულდება ანიმაცია.

```
div {  
    -webkit-animation-delay:  
    animation-delay: 5s;  
}
```

- ✓ *animation-iteration-count*-განსაზღვრავს დროის რაღაც მონაკვეთში, თუ რამდენჯერ უნდა შესრულდეს ანიმაცია.

```
    animation-iteration-count: 5
```

ნიშნავს, ანიმაცია რომ ხუთჯერ შესრულდება

- ✓ *animation-direction*-განსაზღვრავს მიმართულებას, მის სიდიდეში შედის თვისებები:

<i>normal</i>	<i>Default</i> მნიშვნელობა, რომელიც ანიმაციას უკრავს ნორმალურად ანუ იმ პარამეტრებში რაც აქვს გაწერილი
<i>reverse</i>	ანიმაცია იმოდრავებს საპირისპიროდ/რევერსულად
<i>alternate</i>	ანიმაცია იმოდრავებს ნორმალურად, ოდონდაც დროის კენტ რიცხვებში (1,3,5 ...წმ), ხოლო რევერსულად ანუ საპირისპიროდ ყოველ ლუწ წამში ან წუთში
<i>alternate-reverse</i>	ანიმაცია იქნება ყოველ კენტ რიცხვში და იმოდრავებს რევერსული მიმართულებით, ხოლო, კენტ დროის რიცხვებში კი ნორმალურად.

- ✔ *animation-fill-mode*-ზუსტად ადგენს ანიმაცია, როდის ფერხდება, როდის არ სრულდება ანიმაცია .

სინტაქსი:

animation-fill-mode: none, forwards, backwards, both

none *Default* მნიშვნელობაა, სტილი არ გამოჩნდება სანამ ანიმაცია არ შესრულდება

forwards ანიმაციის მორჩენის დროს, ელემენტი გაჩერდება იმ მდგომარეობაში სადაც გაჩერდა.

backwards ანიმაციისას როგორც კი წარმოჩნდება ელემენტი, განსაზღვრული სიდიდე, რომელიც არის პირველი *keyframe* რელევანტური, პირველი რელევანტურობა დამოკიდებულია *keyframe animation-direction* სიდიდეზე:

<i>animation-direction</i>	<i>Keyframe</i> პირველი რელევანტური
<i>normal</i> ან <i>alternate</i>	0% ან <i>from</i>
<i>reverse</i> ან <i>alternate-reverse</i>	100% ან <i>to</i>

both *forwards* და *backwards* თვისებებს ითვალისწინებს და მუშაობს ორივე მიმართულებით

- ✔ *animation-play-state*-მისი აზრი ისაა, რომ ელემენტი ანიმაცია მიმდინარეობს, თუ დაპაუზებულია.

სინტაქსი

paused ან *running*

მაგალითისათვის ავიღოთ რაიმე სურათი, დავუშვათ მანქანა მოძრაობს, ხოლო სურათს, რომ დავაჭერთ გაჩერდება.

მაგალითი 2.54

```
<!DOCTYPE html>
<html Lang="ka">
<head>
<title> ანიმაცია </title>
<style>.manqana{
width: 253px;
height: 203px;
```

```

margin: 50px;
background: url("https://i.ibb.co/qYFSGzc/Picture1.png ") no-
repeat;
position: relative;
/* Chrome, Safari, Opera */
-webkit-animation-name:
-webkit-animation-duration: 6s;
-webkit-animation-play-state: running;
-webkit-animation: moveit; 6s infinite;
animation: moveit; 6s infinite;
/* Standard syntax */
animation-name: moveit;
animation-duration: 6s;
animation-play-state: running;
}
div:hover
{
/* Chrome, Safari, Opera */
-webkit-animation-play-state: paused;
animation-play-state: paused;}
/* Chrome, Safari, Opera */
@-webkit-keyframes moveit
{ from {left: 0;}
to {left: 50%;}}
/* ყველასათვის */
@keyframes moveit
{ from {left: 0;}
to {left: 50%;}}
</style>
</head>
<body>
<div class="manqana"></div>
<p> მანქანა იმოძრაებს, ხოლო თუ „დავაკლიკებთ“ გაჩერდება ან შეგვიძლია
თქვენთვის სასურველი სურათი ჩასვათ .
</p>
</body>
</html>

```

შედეგი



მანქანა იმოძრაებს, ხოლო თუ დავაკლიკებთ გაჩერდება

რა თქმა უნდა, შესაძლებელია სხვა ელემენტების ჩამატება, მაგალითად *animation-iteration-count* და სასურველი რიცხვის გავუწერის მერე ელემენტი დროის მონაკვეთში გაიმეორებს იმდენს რამდენსაც, ეს ჩვენზეა დამოკიდებული.



CSS3-ში წარმოვადგინეთ მაქსიმალურად ყველა ელემენტი თავის თვისებით და სიდიდეებით HTML5-ის უკეთ გასაგებად და ისეთი საკითხების აღსაქმელად, როგორიცაა API-ები და გრაფიკა, მოგვიწევს შევხებით *javascripts* და მას ზოგადად განვიხილავთ, ძირითად თემებს და საკითხებს.

თავი III ჯავასკრიპტი

javascripts საშუალებით კომპიუტერს ვეუბნებით, თუ რისი გაკეთება გვსურს, ხშირად დამწყებებს ან სხვა სფეროს წარმომადგენელს ერევათ ორი ენა *javascripts* და *java*, თუმცა ორივე ენაში არის საერთო ტიპები, ცვლადები, მასივები და ა.შ, მაინც განსხვავებული ენებია, განსხვავებული ფუნქციებითა და ამოცანებით.



ჯავასკრიპტი შექმნილია კომპანია Netscape-ის მიერ, და პირველად ჩართული იყო *Netscape Navigator* ბრაუზერში და ამავე ბრაუზერის ვერსიის ანუ *Netscape Navigator 2.0* შემადგენელი ნაწილი 1995 წლის დეკემბერში გახდა, ხოლო ECMA სტანდარტად 1997 წლიდან მიიღეს, ამის შემდეგ ოფიციალური სახელწოდება *ECMA-262* სას მიენიჭა, უნდა აღინიშნოს რომ *ECMAScript 6* შემუშავდა 2005 წელს და ის გახლავთ ჯავასკრიპტის ბოლო ვერსია.

რაც შეეხება მის გამოყენებას?, იგი ერთ ერთი პოპულარული და ყველაზე ხშირი გამოყენების ენაა, მისი საშუალებით შესაძლებელია ფორმების შექმნა, კალკულაცია და ა.შ ანუ მისი მოვალეობაა, რომ ელემენტი, რომელიც შექმნილია და სტილისტურად გაფორმებულია რაღაცის კეთების უნარი მიანიჭოს.

სკრიპტის წერისას უნდა გავითვალისწინოთ მისი ასოების მიმართ მგრძნობიარობა მაგალითად სახელი *index* და *Index* მისთვის სხვადასხვა სახელია, ერთი ასოც რომ იყოს დიდი, აღიქვამს როგორც სხვა სახელს, მისი წერის დროს უნდა გავითვალისწინოთ :- წერტილ მძიმე, რითაც კოდის ხაზის დასასრულია და გადადის შემდეგ ხაზზე, როგორც CSS-ს შემთხვევაში, აქაც შესაძლებელია სკრიპტის ორნაირად გამოყენება: ჩანერგვით ანუ შიგ ტექსტში `<scripts ></scripts>` ტეგების გამოყენებით და ლინკის საშუალებით.

შიგ ჩანერგილი სკრიპტი იქნება:

```
<html>
<head>
<meta charset="utf8">
<script type="text/javascript">
    document.write ("ვეცნობით javascript ან უბრალოდ HELLO WORLD");
</script>
</head>
<body>
კონტენტი
</body>
</html>
```

შედეგი მისი ისაა, რომ გამოიტანს *document.write*-ში ჩაწერილ ტექსტს, ხოლო თვითონ ბრძანება *document.write*, გამოტანის ბრძანებაა ;

```
<script src="kodi.js">
</script>
```

სადაც *kodi.js* არის ჩვენი ფაილი, რომელსაც ვუკავშირდებით, ხოლო *js* ჯავასკრიპტის გაფართოება შიგ კი წერია:

```
document.write ("ვეცნობით javascript ან უბრალოდ HELLO WORLD");
```

მისი მოთავსება ნებისმიერ ადგილას შეიძლება, ასევე ერთი ან რამდენიმე სკრიპტის დაწერაც, ჯავასკრიპტი მუშაობს :

```
<scripts> </scripts>
```

ან

```
<script type="text/javascript"> </script>
```

ოპერატორები

ოპერატორია ისაა, რითაც ოპერაციების სრულდება, ჯავასკრიპტში არის ოთხი ოპერატორი: არითმეტიკის, ლოგიკის, შედარებისა, ბიტობრივი, სტრიქონული უნდა ითქვას ისიც, რომ არსებობს ოპერანდები, რომელნიც აღნიშნავს რიცხვებს, მაგალითად, $3+2=5$ ციფრები არის ოპერანდები, ხოლო „+“ ნიშანი კი ოპერატორი

არითმეტიკული

ოპერატორი	აღწერა
+	შეკრება
-	გამოკლება
*	გამრავლება
/	გაყოფა
%	მოდული (უნაშთოდ გაყოფა)
++	ინკრიმენტი
--	დეკრიმენტი

არითმეტიკულ ოპერატორებში გაუგებარი ინკრიმენტი და დეკრიმენტი რომ არ იყოს, ვიტყვით, რომ წარმოადგენს ერთი ერთეული რაღაცის გაზრდას ან შემცირებას, მაგალითად:

$x=3$; ერთი ერთეულით გაზრდა იქნება ასე $x++$, მაგრამ დაპროგრამებაში ხშირად არის. ხოლმე, რომ იწერება შემდეგნაირად : $++x$ ამ შემთხვევა შედეგს ჩვენს შემთხვევაში 4 კი არა გამოიტანს, არამედ სამს და შემდეგ დაუმატებს ერთ და გამოიტანს 4-ს., ხოლო თუ წერია $x++$, მაშინ პირდაპირ $3+1$ და გამოაქვს 4.

შედარების ოპერატორები

შედარების ოპერატორების გამოიყენება როდესაც ორი ცვლად ვადარებ ან პირობითად ვუშვებთ, რომ ორი ცვლადი ერთმანეთის ტოლია და ა.შ:

ოპერატორი	აღწერა
==	ტოლია
!=	არ არის ტოლი

>	მეტია
>=	მეტია ან ტოლია
<	ნაკლებია
<=	ნაკლებია ან ტოლია

ლოგიკური ოპერატორები

ლოგიკური ოპერატორების მნიშვნელობა არის ის, რომ პირობა ჭეშმარიტია თუ არა, ანუ *true* და *false*.

&&	და	(<i>x</i> < 9 && <i>y</i> > 3) ჭეშმარიტია
	ან	(<i>x</i> == 5 <i>y</i> == 5) არ არის ჭეშმარიტი
!	არა	!(<i>x</i> == <i>y</i>) ჭეშმარიტია

მინიჭების ოპერატორები

მინიჭების ოპერატორებში ხშირად ხდება მოკლედ ჩაწერა, დაპროგრამებაში მიღებულია საკუთარ თავს დაუმატო ან გამოაკლო რაღაც და ისევ საკუთარ თავს უდრიდეს ანუ:

x=*x*+1 ეს იგივე, რაც *x*+=1

=	<i>X</i> = <i>y</i>	<i>x</i> = <i>y</i>
+=	<i>X</i> += <i>y</i>	<i>x</i> = <i>x</i> + <i>y</i>
-=	<i>x</i> -= <i>y</i>	<i>x</i> = <i>x</i> - <i>y</i>
*=	<i>X</i> *= <i>y</i>	<i>x</i> = <i>x</i> * <i>y</i>
/=	<i>X</i> /= <i>y</i>	<i>x</i> = <i>x</i> / <i>y</i>
%=	<i>X</i> %= <i>y</i>	<i>x</i> = <i>x</i> % <i>y</i>

სტრიქონული ოპერატორი

სტრიქონული ოპერატორებისას ხდება ერთმანეთთან ორი *string*-ის შეკრება, მაგალითად, ავიღოთ ორი ცვლადის შეკრება:

```
var saxeli = "Lasha ";
var gvari = "Iashvili";
shekreba = saxeli+ gvari;
```

კომენტარები

ობიექტზე ორიენტირებულ ენებში როგორიცაა: ჯავასკრიპტი, java, C#, PHP თუ სხვა არსებობს როგორც ერთ სტრიქონიანი კომენტარები, ისე გვაქვს მრავალ სტრიქონიანი კომენტარები.

- ✔ ერთ სტრიქონიანი კომენტარი ჩაიწერება // ორი დახრილი ხაზით
- ✔ მრავალ სტრიქონიანი კომენტარი მოთავსდება /*

მმართველი სიმბოლოები

მმართველი სიმბოლოები აქტიურად გამოიყენება, ერთი სტრიქონიდან, მეორეზე გადასვლიას, უკან დაბრუნებისას თუ სხვა ფუნქციების შესასრულებად.

მმართველი სიმბოლო	აღწერა
\'	ერთმაგი ბრჭყალი
\"	ორმაგი ბრჭყალი
\\	ირიბი დახრილი ხაზი
\0	Null (სიმბოლო, რომლის ნომერია 0)
\a	ზარი
\b	უკან დაბრუნება (BackSpace)
\f	გვერდის გადაფურცვლა.
\n	ახალ სტრიქონზე გადასვლა.
\r	სტრიქონის დასაწყისში გადასვლა.
\t	ჰორიზონტალური ტაბულირება.
\v	ვერტიკალური ტაბულირება.

ჯავასკრიპტის საკვანძო სიტყვები

ჯავასკრიპტს აქვს საკვანძო სიტყვები რითაც პირობის დაწყებამდე ეხმარება მას, რომ შესრულდეს მოქმედება, ასეთი სიტყვებია: *var*, *return*, *debug* და სხვა მრავალი რომელიც 3.1 ცხრილში მოყვანილია

ცხრილი 3.1 ჯავასკრიპტის keyword

Keyword	აღწერა
break	წყვეტს <i>switch</i> ან ციკლს
continue	ახტება ციკლს და აგრძელებს
debugger	იძახებს, თუ შესაძლებელია გაშვებული ფუნქციას.
do ... while	do ასრულებს ბლოკებში არსებულ პირობას, მანამ <i>while</i> პირობა არ იქნება ჭეშმარიტი.
for	ბლოკებში არსებულ, ასრულებს პირობას, სანამ პირობა ჭეშმარიტი არ იქნება.
function	აღწერს ფუნქციას.
if ... else	დამოკიდებულია პირობაზე (თუ..კვლავ).
return	გამოდის ფუნქციიდან .
switch	ბლოკებში აღწერს რამდენიმე პირობას.
try ... catch	შესრულდება, მაგრამ <i>Error</i> -ს დროს ამოაგდებს შეტყობინებას.
var	აღწერს ცვლადებს.

ხოლო ოპერაციები ანუ დაცული სიტყვები, რომელიც არ შეიძლება, რომ შეიცვლოს და შესაძლებელია ამა თუ იმ კოდის დასაწერად და ასევე დაპროგრამების ენა სარგებლობს მოცემულია ცხრილ 3.2-ში, უნდა ითქვას, რომ ჯავასკრიპტში მათი გამოყენება არ შეიძლება:ციკლებთან, ფუნქციებთან, მეთოდებთან და ნებისმიერი ობიექტის სახელებთან.

ცხრილი 3.2 რევერსიული სიტყვები

<i>abstract</i>	<i>arguments</i>	<i>boolean</i>	<i>break</i>	<i>byte</i>
<i>case</i>	<i>catch</i>	<i>char</i>	<i>class*</i>	<i>const</i>
<i>continue</i>	<i>debugger</i>	<i>default</i>	<i>delete</i>	<i>do</i>
<i>double</i>	<i>else</i>	<i>enum*</i>	<i>eval</i>	<i>export*</i>
<i>extends*</i>	<i>false</i>	<i>final</i>	<i>finally</i>	<i>float</i>
<i>for</i>	<i>function</i>	<i>goto</i>	<i>if</i>	<i>implements</i>
<i>import*</i>	<i>in</i>	<i>instanceof</i>	<i>int</i>	<i>interface</i>
<i>let</i>	<i>long</i>	<i>native</i>	<i>new</i>	<i>null</i>
<i>package</i>	<i>private</i>	<i>protected</i>	<i>public</i>	<i>return</i>
<i>short</i>	<i>static</i>	<i>super*</i>	<i>switch</i>	<i>synchronized</i>
<i>this</i>	<i>throw</i>	<i>throws</i>	<i>transient</i>	<i>true</i>
<i>try</i>	<i>typeof</i>	<i>var</i>	<i>void</i>	<i>volatile</i>
<i>while</i>	<i>with</i>	<i>yield</i>		

ცხრილ 3.3-ში მოთავსებული თვისებები, მეთოდები და ობიექტები

<i>Array</i>	<i>Date</i>	<i>eval</i>	<i>function</i>	<i>hasOwnProperty</i>
<i>Infinity</i>	<i>isFinite</i>	<i>isNaN</i>	<i>isPrototypeOf</i>	<i>length</i>
<i>Math</i>	<i>NaN</i>	<i>name</i>	<i>Number</i>	<i>Object</i>
<i>prototype</i>	<i>String</i>	<i>toString</i>	<i>undefined</i>	<i>valueOf</i>

გამოტანა

ინფორმაციის გამოტანას დიდი მნიშვნელობა აქვს, როგორც მის აღწერას, გამოტანის სხვადასხვა მეთოდი უფრო კონკრეტულად, ოთხი მეთოდით შეგვიძლია ინფორმაცია გამოვტანოთ.

- ✓ ***window.alert()***. როდესაც ფანჯრა ამოვარდება ბრაზერში მარტივი ილუსტრირებისთვის ორი რიცხვის შეკრება მოვახდინოთ მაგალით 3.1-ში

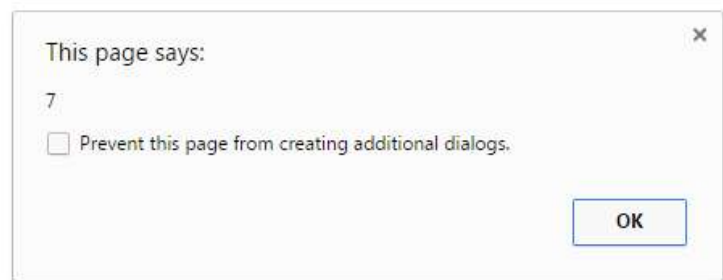
მაგალითი 3.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h1>პირველი შეკრება </h1>
<p>ფანჯრა ამოვარდება </p>
<script>
window.alert(3 + 4);
</script>
</body>
</html>
```

შედეგი

პირველი შეკრება

ფანჯარა ამოვარდება



- ✓ **document.write().** - შედეგს გამოტანს, მაგრამ ფანჯარას აღარ ამოაგდებს, შიგვე ბრაუზერში მაჩვენებს შედეგს, იგივე მაგალითი 3.2-ში აღვწერ ცვლადებს var ტიპში, მაგრამ უფრო მარტივად შეიძლება, უბრალოდ შეკრება, რომ დავწეროთ `document.write(3+ 4);`

მაგალითი 3.2

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
    <h1>პირველი შეკრება</h1>
<p>ფანჯარა ამოვარდება </p>
<script>
var x=3
var y=4
    z=x+y
document.write(z);
</script>
</body>
</html>
```

შედეგი



- ✓ **innerHTML** - შემთხვევაში განისაზღვრება HTML ელემენტის `id` ატრიბუტით, ხოლო ჯავასკრიპტში ჩაიწერება `document.getElementById(id)` მეთოდით, მაგალით 3.3-ში მოცემული მეთოდი პრაქტიკაში აქტიურად გამოყენება.

მაგალითი 3.3

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h1> პირველი შეკრება </h1>
<p> ფანჯარა ამოვარდება </p>
<p id="raime"></p>
<script>
var x=3
var y=4
z=x+y
document.getElementById("raime").innerHTML=z;
</script>
</body>
</html>
```

შედეგით იმავეს მივიღებთ

- ✓ `console.Log()` - კონსოლიერ გარემოში ეკრანზე გამოქვს ინფორმაცია, ბრაუზერში აქტივირდება F12 -ს დაწოლისას.
ანალოგიური კოდი იქნება და ცვლადებს თუ აღვწერთ, როგორც მაგალით 3.3-ში მაში ჩაიწერება:
`console.Log(z);`

ცვლადები

ჯავასკრიპტი ცვლადების აღსაწერად გამოყენებს `var` ტიპს, მაინც რა არის ცვლადები? ცვლადებია ის, რაც შეიძლება შეიცვალოს და ასევე აღწერისას მოკლეთ აღიწეროს, მაგალითად:

```
Var saxeli="";
```

ან

```
Var X=0; ან სხვა რაიმე
```

ცვლადების მინიჭებისასც უნდა ითქვას რომ ასოების მიმართ მგრძნობიარე ასევე, მაგალითად, `x` და `X` სხვადასხვა სახელია, ცვლადის მინიჭების და გამოტანის დემონსტრირება მაგალით 3.1-3.3-ში იყო, დავწეროთ სკრიპტი:

```
<html>
<body>
<p> ცვლადი </p>
<p id="rame"></p>
<script>
var saxeli= "Lasha";
document.getElementById("rame").innerHTML = saxeli;
```

```

</script>
</body>
</html>
გამოიტანს სახელს

```

ფუნქცია /function

ფუნქციები არის ბრძანებათა ჯგუფით, რომელსაც აქვს სახელი, ანუ ინდიფიკატორი, რომელთაც ბრძანებების საშუალებით შესაძლებელია ფუნქციის გამოძახება.

სინტაქსი

```

function სახელი (პარამეტრი 1, პარამეტრი 2)
{
    კოდი, რომელიც უნდა შესრულდეს
}

```

მარტივი ფუნქცია ორი ოთხკუთხედის ფართობის გამოიანგარიშებისან ასეთია, თუ ერთ გვერდს აღვნიშნავთ a -თი, მეორე გვერდს- b , ხოლო ფართობი S , ფორმულა კი არის $S=a*b$, ამის ფუნქცია იქნება:

```

function gamotvale (a,b)
{
    var s=a*b;
    return s;
}

```

ხოლო მაგალით 3.4-ში შევქმენი `textbox`-ები, მარტივი კალკულატორია და ითვლის მართკუთხედის ფართობს.

მაგალითი 3.4

```

<!DOCTYPE html>
<html lang="ka">
<head>
    <meta charset="utf-8" />
    <title>ფუნქცია
</title>
</head>
<body>
    <form name="სახელი">
        a:<input type="Text" name="a" id="a" size="12"> <br /> <br />
        b:<input type="Text" id="b" name="b" size="12" > <br/> <br />
        ფართობი <br />
        <input type="Text" name="area" id="ფართობი" size="12">
    <br><br>
        <input type="button" name="calculate" id="ფართობი" size="12"
onclick="ანგარიში()" value="ანგარიში">
    </form>
</script>

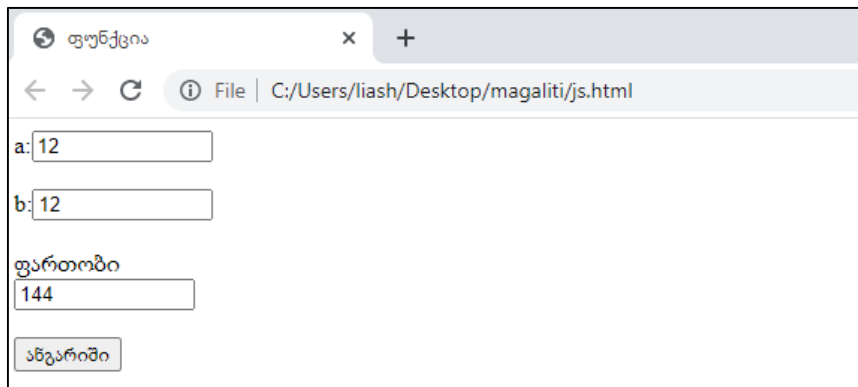
```

```

function ანგარიში()
{
    var a = +document.getElementById("a").value;
    var b = +document.getElementById("b").value;
    var ფართობი = a * b;
    return document.getElementById("ფართობი").value = ფართობი;
}
</script>
</body>
</html>

```

შედეგი



ფუნქციებზე ბევრი ითქვა, უნდა ვიცოდეთ რომ ფუნქცია განისაზღვრება *სახელით*, ნებისმიერი ცვალდით-*value* ანუ, როგორც ცნობილია, *არგუმენტით* და ფუნქციის პირობით.

return-ყურადღებას მივაქცევდით, იგი ამთავრებს ფუნქციის ტანს და აბრუნებს მნიშვნელობას, როგორც მაგალით 3.4 დან ჩანს მაგრამ ამ შემთხვევაში *return*-ს, თუ არ დაწერდით და უბრალოდ დავწერდით

```
document.getElementById("ფართობი").value = ფართობი;
```

ამის შემდეგ მნიშვნელობის გამოძახება მაინც მოხდება.

მმართველი ოპერატორები

მმართველ ოპერატორებში შედის სამი ტიპის ოპერატორი:

- *if* ოპერატორი, რომელიც შესრულდება იმ შემთხვევაში, თუ პირობა ჭეშმარიტია.
- *if..else* ოპერატორი შესრულდება, თუ ერთ ერთი პირობა მცდარია, ხოლო მეორე ჭეშმარიტი.
- *switch* ოპერატორი, რომელიც ბლოკებით არის მოცემულია სადაც აღწერილია სხვადასხვა სიტუაცია.

if ოპერატორი

if ოპერატორი ამოწმებს პირობას და ასრულებს იმ შემთხვევაში, თუ პირობა ჭეშმარიტია, მისი სინტაქსია:

```
if (პირობა)
{
    კოდი, რომელიც უნდა შესრულდეს.
}
```

რეალური მაგალითის საფუძველზე ავიღოთ პრიმიტიული მეტობა ან ნაკლებობა:

```
<script>
    var ricvi =20;
    if( ricvi > 18 )
    {
        document.write(" ricvi ტოლია 20ის ");
    }
</script>
```

თუ გვაქვს მეორე პირობაც, მაშინ შემოღებულია დამატებით else ოპერატორი

```
if (პირობა)
    პირობა, რომელიც უნდა შესრულდეს
}
Else
{
    ეს პირობა უნდა შესრულდეს, თუ if მცდარია
}
```

მაგალითი 3.5

```
<!DOCTYPE html>
<html>
<body>
<p>იმ დროს გამოიტანს, რაც მოცემული მომენტში გვაქვს</p>
<button onclick="დრო()">დააჭირეთ</button>
<p id="ტესტ"></p>
<script>
function დრო()
{
    var saati = new Date().getHours();
    var dro;
    if (saati < 12)
    {
        dro = "დღე მშვიდობისა";
    } else
    {
        dro = "სალამო მშვიდობისა";
    }
    document.getElementById("ტესტ").innerHTML = dro;
```

```

}
</script>
</body>
</html>

```

შედეგი

იმ დროს გამოიტანს, რაც მოცემული მომენტში გვაქვს

დააჭირეთ

საღამო მშვიდობისა

switch ოპერატორი

if ოპერატორისაგან განსხვავებით, *switch* ოპერატორი გვაძლევს შესაძლებლობას, რამდენიმე ვარიანტი რომ წარმოავადგინოთ. თუ ერთ ვარიანტი არ აკმაყოფილებს ჩვენს მოთხოვნებს, გადავა შემდეგ პირობაზე, ხშირად ამ ოპერატორს კალკულატორის დროს გამოიყენებენ, მისი დამახასიათებელია *case*-ები და თითოეული *case* თითო შემთხვევა, აქედან გამომდინარე, როდესაც კალკულატორს წერენ, როგორც აღვნიშნეთ თითოეულ *case*-ში თითო არითმეტიკულ ოპერაციას წერენ.

სინტაქსი

switch (გამოსახულება)

```

{ case მიმდევრობა :ოპერაცია
  break;
  case მიმდევრობა:ოპერაცია
  break;
  ...
  default: ოპერაცია
}

```

შევქმნათ მანქანების მოდელები, რომელიც თითოეულ ფირმას ჰყავს დამზადებული, როდესაც ველში ჩავწერთ ტექსტს, თუ მოდელები არ მოიძებნა, შეტყობინებას გვიწერს, რომ „მოდელები უცნობია“, დანარჩენ *case*-ებზე რომ არ გადავიდეს *break*, ოპერატორს საჭიროებს.

მაგალითი 3.6

```

<!DOCTYPE html>
<html>
<body>
<input id="ეს" type="text" value="ტექსტი">
<button onclick="shemowmeba()">მანქანა</button>
<p id="text">

```



```

</p>
<script>
function shemowmeba()
{
    var text;
    var manqana= document.getElementById("js").value;
    switch(manqana)
    {
        case "VW":
            text = "golf 3, Vento GL, jetta ";
            break;
        case "opel":
            text = "astara, astara sport";
            break;
        case "BMW":
            text = "E34, x6, x5 ";
            break;
        default:
            text = "მოდელები უცნობია";
    }
    document.getElementById("text").innerHTML = text;
}
</script>
</body>
</html>

```

შედეგი

VW მანქანა	ან	opel მანქანა
მოდელები უცნობია		astara, astara sport

ციკლები

ციკლებში შედის სამი ოპერატორი:

- *while* თუ პირობა არა არის ჭეშმარიტი ანუ *false*-ია, ხდება ციკლიდან გამოსვლა ხოლო თუ პირობა ჭეშმარიტია, სრულდება ციკლი, მისი სინტაქსი ჩაიწერება:

```

while (პირობა)
{
    ციკლის კოდი
}

```

- *do..while* ამოწმებს პირობას და შესრულდება ციკლის კოდი, შემდეგ მოწმდება პირობა, მაგალითად, მისი სინტაქსია:

```

do
{
    ციკლის კოდი
}
while ( პირობა)

```

- *for*-ციკლი შესრულდება მანამ სანამ ციკლში აქვს გაწერილი, ეს ოპერატორი, ხშირად გამოიყენება ციკლებში მისი სინტაქსია:

```

for (statement 1; statement 2; statement 3)
{
    სანამ კოდი შესრულდება
}

```

statement 1- ციკლის დაწყების წინ გამოითვლება.

statement 2-სადაც, სრულდება პირობა.

statement 3-ციკლის კოდი, სრულდება ყოველ ჯერზე.

მაგალით 3.7-ში წარმოდგენლია *do...while* ციკლი, ხოლო მაგალით 3.8-ში *for* ოპერატორი.

მაგალითი 3.7

```

<html>
<head>
    <title>ციკლი</title>
</head>
<body>
    <script>
        var count = 0;
        document.write("ციკლი" + "<br />");
        do
        {

            document.write("ციკლი: " + count + "<br />");
            count++;
        }
        while (count <10);
        document.write ("ციკლი შეჩერდა!");
    </script>
</body>
</html>

```

შედეგი

```

ციკლი
ციკლი: 0
ციკლი: 1
ციკლი: 2
ციკლი: 3
ციკლი: 4
ციკლი: 5
ციკლი: 6
ციკლი: 7
ციკლი: 8
ციკლი: 9
ციკლი შეჩერდა!

```

მაგალითი 3.8

```
<html>
  <body>
    <script>
      var count;
      document.write("ციკლი იწყება" + "<br />");
      for(count = 0; count < 5; count++){
        document.write("ციკლი: " + count );
        document.write("<br />");
      }
      document.write("ციკლი ჩერდება");
    </script>
  </body>
</html>
```

შედეგი

```
ციკლი იწყება
ციკლი: 0
ციკლი: 1
ციკლი: 2
ციკლი: 3
ციკლი: 4
ციკლი ჩერდება!
```

როგორც ვხედავთ, ციკლის ნუმერაციის დროს არასდროს არ გამოდის ბოლო რიცხვი დაისმის კითხვა რატომ? იმიტომ, რომ როგორც *for* ციკლისას 0 დან, თუ გადავითვლით ზუსტად ხუთი ელემენტია, მას ნუმერაცია კი არა არამედ ელემენტების რაოდენობა გამოაქვს 0, 1, 2, 3, 4- ჩათვლით.

ობიექტები

javascript არის ობიექტზე ორიენტირებული ენა ანუ *Object-oriented programming (OOP)*, რომელიც მომხმარებელს აძლევს საშუალებას, განისაზღვროს მომხმარებლის მიერ შექმნილი ობიექტები. ობიექტი განისაზღვრება თვისებებით და მეთოდებით.

რა არის მეთოდი ან თვისება? ავიღოთ მანქანა, მანქანა არის ობიექტი, რომელსაც აქვს, ფერი, წონა, სახელი და ა.შ, ხოლო მეთოდი არის ის, თუ რა მოქმედებია შეუძლია ობიექტს ჩვენს მანქანას: დაძვრა, დამუხრუჭება და ა.შ, კიდევ განვიხილოთ ერთი მაგალითი, ავიღოთ „პიროვნება“, რა თქმა უნდა პიროვნება არის ობიექტი, მაგრამ, ჩვენი ობიექტი რა თვისებებით განისაზღვრება? სახელით, ასაკით, წონით, წარმომავლობით და ა.შ, ანუ ობიექტს რა მოქმედებების შესრულება შეუძლია:

```
objectName.propertyName
ან
objectName["propertyName"]
```

ობიექტს, შეიძლება ჰქონდეს უამრავი ცვლადი, მაგალითად მანქანისას:

```
<script>
var manqana = {type:"VW", model:"VENTO", color:"red"};
document.getElementById("demo").innerHTML = manqana.type;
</script>
```

ფერი, მოდელი და ა.შ

მაგალითი 3.9

```
<!DOCTYPE html>
<html>
<body>
<p>javascripts ობიექტი</p>
<p id="ობიექტი"></p>
<script>
var pirovneba= {
    saxeli: "zaza",
    gvari : "iashvili ",
    asaki: 50,
    erovneba : "ქართველი"
};
document.getElementById("ობიექტი").innerHTML = pirovneba.saxeli +
pirovneba.gvari + pirovneba.asaki + " წლის, ეროვნებით"
+pirovneba.erovneba;
</script>
</body>
</html>
```

შედეგი

javascripts ობიექტი

zazaiashvili 50 წლის, ეროვნებით ქართველი

ასევე უნდა ვიცოდეთ, რომ ობიექტის შექმნისას *new* სიტყვის გამოყენებისას ხდება ობიექტის შექმნა.

მაგალითად

var saxeli = new String(); *saxeli* აღიწერება როგორც *string*-ის ობიექტი.

new-ს გამოყენებით პიროვნების ობიექტის ხაზი ჩავსვს, მაგალით 3.9-ში:

```
var pirovneba= new Object();
{
    saxeli: "zaza",
    gvari : "iashvili",
    asaki: 50,
    erovneba : "ქართველი"
};
```

რა თქმა უნდა, შედეგი იგივე იქნება, როგორც ვხედავთ, ჩვენ გავცევით ორ ვარიანტს ობიექტი თუ როგორ უნდა შეიქმნას, პირველი ლიტერალის (*ლიტერალი არის ობიექტი სახელი, მაგალითად asaki:50*) საშუალებით, ხოლო მეორე *new* საკვანძო სიტყვის გამოყენებით, და არსებობს მესამე საშუალება, სადაც ხდება **this** საკვანძო სიტყვის გამოყენება.

პრაქტიკაში ხშირია ობიექტის ტიპის გამოყენება, როგორც ობიექტად, ხოლო ამისთვის საჭიროა შეიქმნას ობიექტის ტიპის კონსტრუქტორის ფუნქცია:

მაგალითი 3.10

```
<!DOCTYPE html>
<html>
<body>
<p id="test"></p>
<script>
function piri(saxeli, gvari, profesia)
{
    this.firstName =saxeli;
    this.lastName = gvari;
    this.job= profesia;
}
var da= new piri("tako", "iashvili", "menejeri");
var mama = new piri("zaza", "iashvi", "leqtori");
document.getElementById("test").innerHTML="დის სახელი"+ da.firstName+
".მამის სახელი" +mama.firstName;
</script>
</body>
</html>
```

შედეგი

დის სახელი tako, მამის სახელი zaza

data

data ობიექტი მუშაობს დროებზე, მისი საშუალებით შესაძლებელია განისაზღვროს დროის ყველა ერთეულში ყველაფერი: წელი, თვე, წელი, წუთი, წამი და მილიწამები, ხოლო დროის ტიპი იქმნება *new Date()* კონსტრუქტორით.

ჯავასკრიპტს აქვს მეთოდები რომლის საშუალებითაც ენიჭება დროითი მახასიათებლები და გამოაქვს როგორც წლიდან დაწყებული მილი წამებით დამთავრებული, ასევე შესაძლებელია გამოვიძახოთ როგორც ლოკალური ისე სარტყლის მიხედვით *UTC* (*universal*, ან *GMT*) დროები.

მისი სინტაქსი შეიძლება ჩაიწეროს ოთხი მეთოდით:

```
new Date()
new Date(milliseconds)
new Date(datestring)
```

new Date(year, month, day, hours, minutes, seconds, milliseconds)

დროის პარამეტრების აღწერისას უნდა ვიცოდეთ ის, რომ შეიძლება არანაირი არგუმენტი არ ჰქონდეს მინიჭებული, ეს ის შემთხვევაა, როდესაც გვინდა ადგილობრივი დროის გამოტანა (ამ წამს თუ რა დროა.)

Datestring-გამოქვს დროის ინფორმაცია, რომელიც გაწერილია, არსებობს დროის გაწერის ოთხი ტიპი:

ISO	"2016-04-25" (საერთაშორისო სტანდარტი)
მოკლედ	"04/25/2016" ან "2016/04/25"
გრძლად	"sep 27 2016" ან "27 sep 2016"
სრულად	"Wednesday setember 27 2016"

დროითი ობიექტების აღწერისას, იგი განისაზღვრება ორი თვისებით, ესაა **კონსტრუქტორი** და **პროტოტიპი**, რაც შეეხება კონსტრუქტორს ესა ფუნქციაა, რაც ქმნის პროტოტიპს, ხოლო თავის მხრივ პროტოტიპი ობიექტს ანიჭებს გარკვეულ თვისებებს, ასევე დროებს აქვს თავის მეთოდები რომელიც აღწერილია (მოკლედ) ცხრილ 3.4-ში.

ცხრილი 3.4 აღწერილია მეთოდები

მეთოდები	აღწერა
<i>Date()</i>	გამოაქვს დრო და რიცხვი
<i>getDate()</i>	გამოაქვს დრო და რიცხვი
<i>getDay()</i>	გამოაქვს კვირის დღეები
<i>getFullYear()</i>	გამოაქვს სრული წელი
<i>getHours()</i>	ლოკალური დროისათვის გამოაქვს საათი
<i>getMilliseconds()</i>	გამოაქვს მილი წამები
<i>getMinutes()</i>	გამოაქვს ადგილობრივი დროის წუთები
<i>getMonth()</i>	მოცემული დროისათვის გამოაქვს თვე
<i>getSeconds()</i>	გამოაქვს წამები

სრულად მოცემულია დროითი მეთოდები:

```
Date.UTC()  
Date.now()  
Date.parse()  
Date.prototype.getDate()  
Date.prototype.getDay()  
Date.prototype.getFullYear()  
Date.prototype.getHours()  
Date.prototype.getMilliseconds()  
Date.prototype.getMinutes()  
Date.prototype.getMonth()  
Date.prototype.getSeconds()  
Date.prototype.getTime()  
Date.prototype.getTimezoneOffset()  
Date.prototype.getUTCDate()  
Date.prototype.getUTCDay()
```

```

Date.prototype.getUTCFullYear()
Date.prototype.getUTCHours()
Date.prototype.getUTCMilliseconds()
Date.prototype.getUTCMinutes()
Date.prototype.getUTCMonth()
Date.prototype.getUTCSeconds()
Date.prototype.getYear()
Date.prototype.setDate()
Date.prototype.setFullYear()
Date.prototype.setHours()
Date.prototype.setMilliseconds()
Date.prototype.setMinutes()
Date.prototype.setMonth()
Date.prototype.setSeconds()
Date.prototype.setTime()
Date.prototype.setUTCDate()
Date.prototype.setUTCFullYear()
Date.prototype.setUTCHours()
Date.prototype.setUTCMilliseconds()
Date.prototype.setUTCMinutes()
Date.prototype.setUTCMonth()
Date.prototype.setUTCSeconds()
Date.prototype.setYear()
Date.prototype.toString()
Date.prototype.toGMTString()
Date.prototype.toISOString()
Date.prototype.toJSON()
Date.prototype.toLocaleDateString()
Date.prototype.toLocaleFormat()
Date.prototype.toLocaleString()
Date.prototype.toLocaleTimeString()
Date.prototype.toSource()
Date.prototype.toString()
Date.prototype.toTimeString()
Date.prototype.toUTCString()
Date.prototype.valueOf()
Date.prototype[@@toPrimitive]
მოვიყვანოთ Date.prototype.toUTCString()

```

მაგალითი 3.11

```

<!DOCTYPE html>
<html>
<title>
დროების ჩვენება
</title>
<head>
<meta charset="utf-8">
</head>
<body>

```

```
<p id="dro"></p>
<script>
var today = new Date();
var UTCstring = today.toUTCString();
document.getElementById("dro").innerHTML =today;
</script>
</body>
</html>
შედეგი
```

Tue Sep 20 2016 15:01:40 GMT+0400 (Georgian Standard Time)

მათემატიკა

რაც შეეხება მათემატიკური ოპერაციების ჩატარების დროს, ჯავასკრიპტს აქვს მათემატიკური ისეთი ფუნქციები, როგორიცაა ახარისხება, ნეპერის რიცხვის გამოტანა და ა.შ

მაგალითად π გამოტანა

```
var numberPI = Math.PI;
console.log( numberPI );
```

მათემატიკური მეთოდები

<i>Abs(x)</i>	აბსოლუტური მნიშვნელობა
<i>Sin(x)</i>	სინუსი
<i>Cos(x)</i>	კოსინუსი
<i>Tan(x)</i>	ტანგენსი
<i>Sqrt(x)</i>	კვადრატული ფესვი
<i>Exp(x)</i>	ექსპონენტი
<i>Log(x)</i>	ლოგარითმი
<i>Log10(x)</i>	ათობითი ლოგარითმი
<i>Max(x, y)</i>	ორ რიცხვს შორის უდიდესი
<i>Min(x, y)</i>	ორ რიცხვს შორის უმცირესი
<i>Pow(x, y)</i>	ახარისხება
<i>Random()</i>	ამორჩევით გასცემს მნიშვნელობას
<i>Floor(x)</i>	დამრგვალება ნაკლებობით
<i>Ceiling(x)</i>	დამრგვალება მეტობით
<i>sqrt</i>	ახარისხება
<i>toSource</i>	გასცემს სტრიქონის Math
<i>Round(x, y)</i>	ფორმატირებული გამოტანა
<i>E</i>	2,71
<i>PI</i>	3,14
<i>Sign(x)</i>	არგუმენტის ნიშანი

მაგალით 3.12

```
<!DOCTYPE html>
<html>
<title>
    მათემატიკური ფუნქციები
</title>
<head>
<meta charset="utf-8">
</head>
<body>
<p> ფესვი </p>
<script>
    var value = Math.sqrt( 0.545 );
    document.write("0.545 ფესვია: " + value );
    var value = Math.sqrt( 81 );
    document.write("<br />81 ფესვია : " + value );
    var value = Math.sqrt( 19 );
    document.write("<br />T19 ფესვია: " + value );
    var value = Math.sqrt( 1569 );
    document.write("<br />1569 ფესვია : " + value );
</script>
</body>
</html>
შედეგი
```

ფესვი

0.545 ფესვია: 0.73824115301167
81 ფესვია : 9
T19 ფესვია: 4.358898943540674
1569 ფესვია : 39.61060464067672

მასივები

მასივები არის ელემენტის ან ელემენტთა ერთობლიობა, მასივში შეგვიძლია მოვათავსოთ ისეთი ელემენტები როგორცაა, სახელი და გვარი, პიროვნება, მანქანა, და ა.შ. მასივი საშუალებას გვაძლევს ერთი ცვლადით შევინახოთ და ვისარგებლოთ მონაცემთა მრავალი ელემენტით, მაგალითად ინსტრუმენტებში შედის: გიტარა, ვიოლინო, პიანინო და ა.შ როგორ იქნება ისინი მასივში:

```
var instrumentebi= new Array("გიტარა", "ვიოლინო", "პიანინო");
```

რაც შეეხება მასივის გამოცხადება შესაძლებელია როგორ *new* საშუალებით ასევე სახელი:

```
var instrumentebi= ["გიტარა", "ვიოლინო", "პიანინო"];
```

რომელ მეთოდს გამოვიყენებთ მასივებში ელემენტების აღსაწერად, ეგ ჩვენზე არის დამოკიდებული, მაგრამ აუცილებელი პირობა როდესაც ელემენტების რაოდენობის ათვლა ხდება, იგი იწყება 0 დან და არა 1-დან, ე.ი როდესაც მასივის გამოცხადებისას მიუთითებთ 1-ს გამოვა „ვიოლინო“ და არა „გიტარა“, რადგან მისთვის იგი მეორე ელემენტია.

მასივში ელემენტების რაოდენობის დათვალა ხდება შემდეგნაირად:

საწყის ეტაპზე შევქმნათ მასივი „ხილი“ სადაც შიგ მოთავსებული იქნება სხვადასხვა ხილ-ბოსტნეული

მაგალითი 3.13

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h1>JavaScript მასივები</h1>
<p id="masivi">
</p>
<script>
var xili= ["ბანანი", "ფორთოხალი", "კივი", "ვაშლი", "მანგო"];
document.getElementById("masivi").innerHTML = xili.length;
</script>
</body>
</html>
```

შედეგი: გამოიტანს 5, რადგანაც, 5 ელემენტია, ახლა მასივში არსებული ყველა ელემენტი გამოვაცხადოთ, ამისთვის დაგჭირდება ციკლი:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h1>JavaScript მასივები</h1>
<p id="masivi">
</p>
<script>
var xili, text, cikli, i;
xili= ["ბანანი", "ფორთოხალი", "კივი", "ვაშლი", "მანგო"];
cikli = xili.length;
text = "<ul>";
for (i = 0; i < cikli; i++)
{
    text += "<li>" + xili[i] + "</li>";
}
text += "</ul>";
document.getElementById("masivi").innerHTML = text;
</script>
</body>
</html>
```

JavaScript მასივები

- ზანანი
- ფორთოხალი
- კივი
- ვაშლი
- მანგო

მასივებზე ბევრი რამის დაწერა შეიძლება, ამიტომაც მას აღარ შევეხებით, იმასაც აღვნიშნავ, რომ ახლა ვლაპარაკობ ერთ განზომილებიან მასივზე, სადაც ელემენტები ერთ სტრიქონზეა წარმოდგენილი, მაგრამ არსებობს კიდევ ორ ან მრავალ განზომილებიანი მასივი, სადაც ელემენტები წარმოდგენილია მატრიცულად, ასევე არის დაკბილური მასივები, იგივე მრავალგანზომილებიანი მასივია განსხვავებით ის, რომ ინფორმაცია სრულად არა შევსებული, თითოეულ სტრიქონში შეიძლება ინფორმაცია არ ეწეროს, ამიტომაც უწოდებენ დაკბილული მასივის, რაც შეეხება მათ მეთოდებს წარმოდგენილია ცხრილ 3.5-ში.

ცხრილი 3.5 მასივების მეთოდები

მეთოდი	აღწერა
<i>concat()</i>	სხვა მასივში აერთიანებს მასივებს და მათ სიდიდეებს.
<i>every()</i>	იღებს ფუნქციას როგორც არგუმენტს.
<i>filter()</i>	მასივში ფილტრავს ელემენტებს.
<i>forEach()</i>	მასივიდან იძახებს ყველა ელემენტს.
<i>indexOf()</i>	გასცემს პირველ (უკანასკნელ) ელემენტს, რომელიც მასივში მოიძებნა.
<i>join()</i>	აერთიანებს მასივების ელემენტს.
<i>lastIndexOf()</i>	გასცემს ყველაზე უკანასკნელ (დიდ) ელემენტს, რომელიც მასივში მოიძებნა.
<i>map()</i>	ახალ მასივში იძახებს იმ ფუნქციით რაც დაწერილი აქვს ელემენტებს.
<i>pop()</i>	უკანასკნელ ელემენტს წაშლის მასივში და გამოტანს ამ ელემენტს.
<i>push()</i>	ამატებს ელემენტებს მასივში.
<i>reduce()</i>	მიმართავს ერთდოულად მასივის ელემენტებს და ამცირებს ერთობოულად (მარცნიდან მარჯვნივ) ერთი ერთეულით.
<i>reduceRight()</i>	მიმართავს ერთდოულად მასივის ელემენტებს და ამცირებს ერთობოულად (მარჯვნიდან მარცხნივ) ერთი ერთეულით.
<i>reverse()</i>	რევერსული მეთოდი, პირველი ელემენტი ხდება უკანასკნელი და უკანასკნელი პირველი.
<i>shift()</i>	შლის პირველ ელემენტს მასივიდან .
<i>slice()</i>	რადაც მონაკვეთს იღებს მასივიდან და წერს ახალ მასივში.
<i>some()</i>	აბრუნებს TRUE მნიშვნელობას, თუ ერთი ელემენტი ამ მასივი აკმაყოფილებს, მითითებულ ფუნქციას.
<i>toSource()</i>	ობიექტში წარმოადგენს <i>source</i> კოდს.

<code>sort()</code>	დაალაგებს ელემენტებს.
<code>splice()</code>	მასივიდან ამატებს ან წაშლის ელემენტს.
<code>toString()</code>	გამოაქვს სტრიქონულად ელემენტები.
<code>unshift()</code>	ამატებს ერთ ელემენტს მასივის წინ და აბურუნებს მასივში ახალ რაოდენობას მასივში.

მაგალითი 3.14

```
<!DOCTYPE html>
<html>
<title>
მასივები
</title>
<head>
<meta charset="utf-8">
</head>
<body>
<p>ლილაკზე დაკლიკებით მასივში ელემენტების ჯამი გამოვა.</p>
<button onClick="myFunction()">
დაჭერა
</button>
<p>მასივში რიცხვების ნამრავლი: <span id="ტესტ">
</span>
</p>
<script>
var numbers = [15, 24, 10, 5, 7, 9];
function jami(zogadi, num)
{
    return zogadi* num;
}
function myFunction(item)
{
    document.getElementById("ტესტ").innerHTML = numbers.reduce(jami);
}
</script>
</body>
</html>
```

შედეგი

ლილაკზე დაკლიკებით მასივში ელემენტების ჯამი გამოვა.

დაჭერა

მასივში რიცხვების ნამრავლი: 1134000

JavaScript გლობალური თვისებები და ფუნქციები

JavaScript აქვს გლობალური ფუნქციები და თვისებები, რომელიც ჩაშენებულია ობიექტებში და რომელთა გამოძახება ნებისმიერ დროს არის შესაძლებელი, რა თქმა უნდა, მათ დეტალურად არ განვიხილავთ, ამიტომაც ცხრილის სახით არის მოცემული.

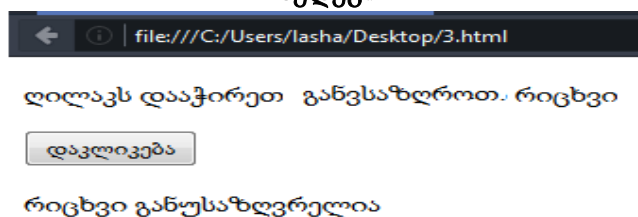
მაგალითი 3.6 თვისებები

თვისება	აღწერა
<i>Infinity</i>	უსასრულობა როგორც ურაცოფითი/დადებითი
<i>NaN</i>	გამოქვს როცა რიცხვითი სიდიდე არაა
<i>undefined</i>	განუსაზღვრელი სიდიდე

მაგალითი 3.15

```
<!DOCTYPE html>
<html>
<head>
<title> განსაზღვრეთ რიცხვი </title>
<meta charset="utf-8">
</head>
<body>
<p>ღილაკს დააჭირეთ რიცხვი განსაზღვრეთ </p>
<button onClick="myFunction()">
დაკლიკება
</button>
<p id="ტექსტი"></p>
<script>
function myFunction()
{
    var ricxvi;
    if (ricxvi === undefined)
    {
        txt = " რიცხვი განუსაზღვრელია ";
    } else {
        txt = "რიცხვი განსაზღვრულია";
    }
    document.getElementById("ტექსტი").innerHTML = txt;
}
</script>
</body>
<html>
```

შედეგი



ფუნქციები

ცხრილი 3.7 გლობალური ფუნქციები

ფუნქცია	აღწერა
<code>decodeURI()</code>	დაშიფრავს
<code>decodeURIComponent()</code>	დაშიფრავს URI კომპონენტებს
<code>encodeURI()</code>	გაშიფრავს URI
<code>encodeURIComponent()</code>	გაშიფრავს URI კომპონენტებს
<code>escape()</code>	შედის და იყენებს <code>encodeURI()</code> ან <code>encodeURIComponent()</code> შემადგენლობაში
<code>eval()</code>	ამოწმებს და ასრულებს კოდს
<code>isFinite()</code>	განსაზღვრავს, თუ სიდიდე ისაზღვრება
<code>isNaN()</code>	განსაზღვრავს, არის თუ არა ციფრი
<code>Number()</code>	გარდაქმნის ობიექტს რიცხვებად
<code>parseFloat()</code>	ანიჭებს <code>float</code> დიაპაზონში რიცხვებს
<code>parseInt()</code>	ანიჭებს <code>int</code> დიაპაზონში რიცხვებს
<code>String()</code>	გამოაქვს ობიექტი
<code>unescape()</code>	შედის და იყენებს <code>encodeURI()</code> ან <code>encodeURIComponent()</code> შემადგენლობაში

მაგალითი 3.16

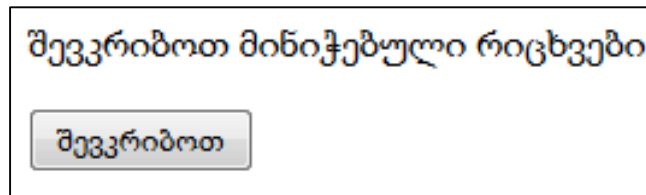
```

<!DOCTYPE html>
<html>
<head>
<title>
  parsint ჩვენება
</title>
<meta charset="utf-8">
</head>
<body>
<p> შევკრიბოთ მინიჭებული რიცხვები ..</p>
<button onClick="myFunction()">შევკრიბოთ</button>
<script>
function myFunction()
{
  var a = parseInt("10") ;
  var b = parseInt("60") ;
  var c = a+b;
  document.write(c);
}
</script>
</body>
</html>

```

შედეგი

ლილაკის დაჭერის შემდეგ გამოტანს მათ ჯამს 70-ს.



DOM მიმოხილვა

HTML ელემენტების განხილვისას მას ზოგადად შეეხებოდა, ახალაღ ვიტყვით იმას, რომ, მისი საშუალებით შესაძლებელია არსებული ელემენტების მიხედვით დაქვეყნება, HTML დოკუმენტის ელემენტებს წარმოადგენს ობიექტის ელემენტებს და აძლევს შესაძლებლობას, რომ მის მეშვეობით მოხდეს გარკვეული ცვლილებები HTML-ის დოკუმენტში, მაგალითად მისი შემადგენელი ნაწილია `document.write()`, `document.applets()` და სხვა, რა თქმა უნდა, ამით ცალკეულად არ განვიხილავთ, ინფორმაციისათვის მათ ცხრილების სახით წარმოვადგენთ.

ცხრილი 3.8 წარმოდგენილია ობიექტების თვისებები და ფუნქციები

თვისებები/მეთოდები	
<code>document.activeElement</code>	<code>document.querySelectorAll()</code>
<code>document.addEventListener()</code>	<code>document.readyState</code>
<code>document.adoptNode()</code>	<code>document.referrer</code>
<code>document.anchors</code>	<code>document.removeEventListener()</code>
<code>document.applets</code>	<code>document.renameNode()</code>
<code>document.baseURI</code>	<code>document.scripts</code>
<code>document.body</code>	<code>document.strictErrorChecking</code>
<code>document.close()</code>	<code>document.title</code>
<code>document.cookie</code>	<code>document.URL</code>
<code>document.createAttribute()</code>	<code>document.write()</code>
<code>document.createComment()</code>	<code>document.writeln()</code>
<code>document.createDocumentFragment()</code>	<code>document.querySelectorAll()</code>
<code>document.createElement()</code>	<code>document.readyState</code>
<code>document.createTextNode()</code>	<code>document.referrer</code>
<code>document.doctype</code>	<code>document.removeEventListener()</code>
<code>document.documentElement</code>	<code>document.getElementsByTagName()</code>
<code>document.documentMode</code>	<code>document.hasFocus()</code>
<code>document.documentURI</code>	<code>document.head</code>

<code>document.domain</code>	<code>document.images</code>
<code>document.domConfig</code>	<code>document.implementation</code>
<code>document.embeds</code>	<code>document.importNode()</code>
<code>document.forms</code>	<code>document.inputEncoding</code>
<code>document.getElementById()</code>	<code>document.lastModified</code>
<code>document.getElementsByTagName()</code>	<code>document.links</code>
<code>document.getElementsByName()</code>	<code>document.normalize()</code>

ცხრილი 3.9 წარმოდგენილი HTML ელემენტების თვისებები და მეთოდები

<code>element.accessKey</code>	<code>element.Lang</code>
<code>element.addEventListener()</code>	<code>element.lastChild</code>
<code>element.appendChild()</code>	<code>element.lastElementChild</code>
<code>element.attributes</code>	<code>element.namespaceURI</code>
<code>element.blur()</code>	<code>element.nextSibling</code>
<code>element.childElementCount</code>	<code>element.nextElementSibling</code>
<code>element.childNodes</code>	<code>element.nodeName</code>
<code>element.children</code>	<code>element.nodeType</code>
<code>element.classList</code>	<code>element.nodeValue</code>
<code>element.className</code>	<code>element.normalize()</code>
<code>element.click()</code>	<code>element.offsetHeight</code>
<code>element.clientHeight</code>	<code>element.offsetWidth</code>
<code>element.clientLeft</code>	<code>element.offsetLeft</code>
<code>element.clientTop</code>	<code>element.offsetParent</code>
<code>element.clientWidth</code>	<code>element.offsetTop</code>
<code>element.cloneNode()</code>	<code>element.ownerDocument</code>
<code>element.compareDocumentPosition()</code>	<code>element.parentNode</code>
<code>element.contains()</code>	<code>element.parentElement</code>
<code>element.contentEditable</code>	<code>element.previousSibling</code>
<code>element.dir</code>	<code>element.previousElementSibling</code>
<code>element.firstChild</code>	<code>element.querySelector()</code>
<code>element.firstElementChild</code>	<code>element.querySelectorAll()</code>
<code>element.focus()</code>	<code>element.removeAttribute()</code>
<code>element.getAttribute()</code>	<code>element.removeAttributeNode()</code>
<code>element.getAttributeNode()</code>	<code>element.removeChild()</code>
<code>element.getElementsByTagName()</code>	<code>element.replaceChild()</code>
<code>element.getElementsByTagName()</code>	<code>element.removeEventListener()</code>
<code>element.getFeature()</code>	<code>element.scrollHeight</code>
<code>element.hasAttribute()</code>	<code>element.scrollLeft</code>
<code>element.hasAttributes()</code>	<code>element.scrollTop</code>
<code>element.hasChildNodes()</code>	<code>element.scrollWidth</code>
<code>element.id</code>	<code>element.setAttribute()</code>
<code>element.innerHTML</code>	<code>element.setAttributeNode()</code>
<code>element.insertBefore()</code>	<code>element.style</code>
<code>element.isContentEditable</code>	<code>element.tabIndex</code>
<code>element.isDefaultNamespace()</code>	<code>element.tagName</code>
<code>element.isEqualNode()</code>	<code>element.textContent</code>
<code>element.isSameNode()</code>	<code>element.title</code>

<code>element.isSupported()</code>	<code>element.toString()</code>
<code>element.accessKey</code>	<code>nodeList.item()</code>
<code>element.addEventListener()</code>	<code>nodeList.length</code>
<code>element.appendChild()</code>	<code>element.lang</code>
<code>element.attributes</code>	<code>element.lastChild</code>
<code>element.blur()</code>	<code>element.lastElementChild</code>
<code>element.childElementCount</code>	<code>element.namespaceURI</code>
<code>element.childNodes</code>	<code>element.nextSibling</code>
<code>element.children</code>	<code>element.nextElementSibling</code>
<code>element.classList</code>	<code>element.nodeName</code>
<code>element.className</code>	<code>element.nodeType</code>
<code>element.click()</code>	<code>element.nodeValue</code>
<code>element.clientHeight</code>	<code>element.normalize()</code>
<code>element.clientLeft</code>	<code>element.offsetHeight</code>
<code>element.clientTop</code>	<code>element.offsetWidth</code>
<code>element.clientWidth</code>	<code>element.offsetLeft</code>
<code>element.cloneNode()</code>	<code>element.offsetParent</code>
<code>element.compareDocumentPosition()</code>	<code>element.offsetTop</code>
<code>element.contains()</code>	<code>element.ownerDocument</code>
<code>element.contentEditable</code>	<code>element.parentNode</code>
<code>element.dir</code>	<code>element.parentElement</code>
<code>element.firstChild</code>	<code>element.previousSibling</code>
<code>element.firstElementChild</code>	<code>element.previousElementSibling</code>
<code>element.focus()</code>	<code>element.querySelector()</code>
<code>element.getAttribute()</code>	<code>element.querySelectorAll()</code>
<code>element.getAttributeNode()</code>	<code>element.removeAttribute()</code>
<code>element.getElementsByClassName()</code>	<code>element.removeAttributeNode()</code>
<code>element.getElementsByTagName()</code>	<code>element.removeChild()</code>
<code>element.getFeature()</code>	<code>element.replaceChild()</code>
<code>element.hasAttribute()</code>	<code>element.removeEventListener()</code>
<code>element.hasAttributes()</code>	<code>element.scrollHeight</code>
<code>element.hasChildNodes()</code>	<code>element.scrollLeft</code>
<code>element.id</code>	<code>element.scrollTop</code>
<code>element.innerHTML</code>	<code>element.scrollWidth</code>
<code>element.insertBefore()</code>	<code>element.setAttribute()</code>
<code>element.isContentEditable</code>	<code>element.setAttributeNode()</code>
<code>element.isDefaultNamespace()</code>	<code>element.style</code>
<code>element.isEqualNode()</code>	<code>element.tabIndex</code>
<code>element.isSameNode()</code>	<code>element.tagName</code>
<code>element.isSupported()</code>	<code>element.textContent</code>
<code>element.accessKey</code>	<code>element.title</code>
<code>element.addEventListener()</code>	<code>element.toString()</code>
<code>element.appendChild()</code>	<code>nodeList.item()</code>

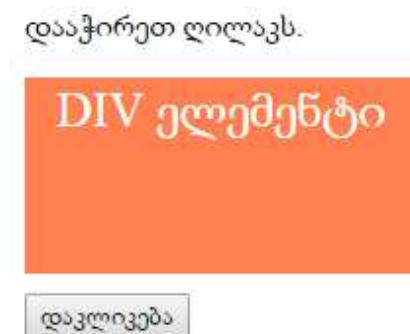
მაგალითი 3.17

```
<!DOCTYPE html>
<html>
```

```

<head>
<meta charset="utf8">
<style>
.სტილი
{
    width: 200px;
    height: 100px;
    background-color: red;
    text-align: center;
    font-size: 25px;
    color: white;
    margin-bottom: 10px;
}
</style>
</head>
<body>
<p>დააჭირეთ ღილაკს.</p>
<div id="divelementi">
DIV ელემენტი
</div>
<button onClick="myFunction()">დაკლიკება</button>
<script>
function myFunction()
{
    document.getElementById("divelementi").className = "სტილი";
}
</script>
</body>
</html>
შედეგი

```



ანალოგიურად ცხრილი 3.10 და 3.11 სახით მოცემულია ატრიბუტები და სტილი, რა თქმა უნდა, მათ ჩვენ არ გავიხილავთ, გაცნობის მიზნით, ზოგადად, ვეხებით DOM ფუნქციებსა და მეთოდებს.

Attr ობიექტების იმავე ატრიბუტების ობიექტები წარმოადგენს HTML ატრიბუტებს, ეკუთვნის ამავე ელემენტებს, მათი თვისებები და მეთოდები მოცემულია ცხრილი 3.10-ში ამავე ცხრილში მოცემულია *NodeMap* ობიექტები, რომელიც ასე ვთქვათ, არის უწყესრიგო ელემენტების შემადგენელი ატრიბუტების კოლექცია, რომელნიც, გარკვეულწილად განისაზღვრება ციფრებით.

ცხრილი 3.10 Attr და NodeMap ობიექტები

<i>attr.isId</i>	<i>nodemap.item()</i>
<i>attr.name</i>	<i>nodemap.Length</i>
<i>attr.value</i>	<i>nodemap.removeNamedItem()</i>
<i>attr.specified</i>	<i>nodemap.setNamedItem()</i>
<i>nodemap.getNamedItem()</i>	

სტილისტური ელემენტები, რომელნიც მოცემული გვაქვს ცხრილ 3.11-ში, ისინი განხილულია CSS- და ასევეა DOM ნაწილიც.

ცხრილი 3.11 DOM სტილის ელემენტები

<i>alignContent</i>	<i>alignContent</i>
<i>alignItems</i>	<i>alignItems</i>
<i>alignSelf</i>	<i>alignSelf</i>
<i>animation</i>	<i>animation</i>
<i>animationDelay</i>	<i>animationDelay</i>
<i>animationDirection</i>	<i>animationDirection</i>
<i>animationDuration</i>	<i>animationDuration</i>
<i>animationFillMode</i>	<i>animationFillMode</i>
<i>animationIterationCount</i>	<i>animationIterationCount</i>
<i>animationName</i>	<i>animationName</i>
<i>animationTimingFunction</i>	<i>animationTimingFunction</i>
<i>animationPlayState</i>	<i>animationPlayState</i>
<i>background</i>	<i>background</i>
<i>backgroundAttachment</i>	<i>backgroundAttachment</i>
<i>backgroundColor</i>	<i>backgroundColor</i>
<i>backgroundImage</i>	<i>backgroundImage</i>
<i>backgroundPosition</i>	<i>backgroundPosition</i>
<i>backgroundRepeat</i>	<i>backgroundRepeat</i>
<i>backgroundClip</i>	<i>backgroundClip</i>
<i>backgroundOrigin</i>	<i>backgroundOrigin</i>
<i>backgroundSize</i>	<i>backgroundSize</i>
<i>backfaceVisibility</i>	<i>backfaceVisibility</i>
<i>border</i>	<i>border</i>
<i>borderBottom</i>	<i>borderBottom</i>
<i>borderBottomColor</i>	<i>borderBottomColor</i>
<i>borderBottomLeftRadius</i>	<i>borderBottomLeftRadius</i>
<i>borderBottomRightRadius</i>	<i>borderBottomRightRadius</i>
<i>borderBottomStyle</i>	<i>borderBottomStyle</i>
<i>borderBottomWidth</i>	<i>borderBottomWidth</i>
<i>borderCollapse</i>	<i>borderCollapse</i>
<i>borderColor</i>	<i>borderColor</i>
<i>borderImage</i>	<i>borderImage</i>
<i>borderImageOutset</i>	<i>borderImageOutset</i>
<i>borderImageRepeat</i>	<i>borderImageRepeat</i>
<i>borderImageSlice</i>	<i>borderImageSlice</i>
<i>borderImageSource</i>	<i>borderImageSource</i>
<i>borderImageWidth</i>	<i>borderImageWidth</i>

<i>borderLeft</i>	<i>borderLeft</i>
<i>borderLeftColor</i>	<i>borderLeftColor</i>
<i>borderLeftStyle</i>	<i>borderLeftStyle</i>
<i>borderLeftWidth</i>	<i>borderLeftWidth</i>
<i>borderRadius</i>	<i>borderRadius</i>
<i>borderRight</i>	<i>borderRight</i>
<i>borderRightColor</i>	<i>borderRightColor</i>
<i>borderRightStyle</i>	<i>borderRightStyle</i>
<i>borderRightWidth</i>	<i>borderRightWidth</i>
<i>borderSpacing</i>	<i>borderSpacing</i>
<i>borderStyle</i>	<i>borderStyle</i>
<i>borderTop</i>	<i>borderTop</i>
<i>borderTopColor</i>	<i>borderTopColor</i>
<i>borderTopLeftRadius</i>	<i>borderTopLeftRadius</i>
<i>borderTopRightRadius</i>	<i>borderTopRightRadius</i>
<i>borderTopStyle</i>	<i>borderTopStyle</i>
<i>borderTopWidth</i>	<i>borderTopWidth</i>
<i>borderWidth</i>	<i>borderWidth</i>
<i>bottom</i>	<i>bottom</i>
<i>boxDecorationBreak</i>	<i>boxDecorationBreak</i>
<i>boxShadow</i>	<i>boxShadow</i>
<i>boxSizing</i>	<i>boxSizing</i>
<i>captionSide</i>	<i>captionSide</i>
<i>clear</i>	<i>clear</i>
<i>clip</i>	<i>clip</i>
<i>color</i>	<i>color</i>
<i>columnCount</i>	<i>columnCount</i>
<i>columnFill</i>	<i>columnFill</i>
<i>columnGap</i>	<i>columnGap</i>
<i>columnRule</i>	<i>columnRule</i>
<i>columnRuleColor</i>	<i>columnRuleColor</i>
<i>columnRuleStyle</i>	<i>columnRuleStyle</i>
<i>columnRuleWidth</i>	<i>columnRuleWidth</i>
<i>columns</i>	<i>columns</i>
<i>columnSpan</i>	<i>columnSpan</i>
<i>columnWidth</i>	<i>columnWidth</i>
<i>content</i>	<i>content</i>
<i>counterIncrement</i>	<i>counterIncrement</i>
<i>counterReset</i>	<i>counterReset</i>
<i>cursor</i>	<i>cursor</i>
<i>direction</i>	<i>direction</i>
<i>display</i>	<i>display</i>
<i>emptyCells</i>	<i>emptyCells</i>

ჯავასკრიპტი ამოუწურავი თემაა, მისი თვისებები, მეთოდები და ფუნქციების განხილვას ასევე ყოველ ობიექტს ამომწურავად ვერ განვხილავთ, ზოგადად წარმოავადგინეთ, ჯავასკრიპტის თვისებები და ობიექტები, რომ ნათელი ყოფილიყოს შემდეგი საკითხები, როგორც არის API და გრაფიკა, რომელსაც IV თავში შევხებით .

HTML5 API თავი IV

უწინ ინტერნეტში უწინ სხვადასხვა ობიექტების შესაქმნელად სურათები გამოყენებოდა, მაგრამ პრობლემას წარმოადგენდა ის, რომ იკარგებოდა ხარისხი, HTML5-ში მსგავსი პრობლემები მოგვარებულია SVG, თუ Canava-ს საშუალებით, რა თქმა უნდა შევხებით ისეთ საკითხებს როგორცაა HTML5 API ანუ **Application Programming Interface-აპლიკაციის პროგრამირების ინტერფეისი** კლასების, ფუნქციების, პროცედურების მზა კომპლექტი, რომელსაც აპლიკაცია საოპერაციო სისტემასთან იყენებს ან სხვა რომელიმე პროგრამასთან სამუშაოდ, პროგრამისტები მას ხშირად იყენებენ აპლიკაციების შესაქმნელად, ის თამაშობს ინტერფეისის როლს პროგრამებს შორის და ამარტივებს მათ ურთიერთქმედებას. მისი ანალოგია მომხმარებელსა და კომპიუტერს შორის ურთიერთობისას არის მომხმარებლის ინტერფეისი.

გრაფიკული ელემენტები SVG

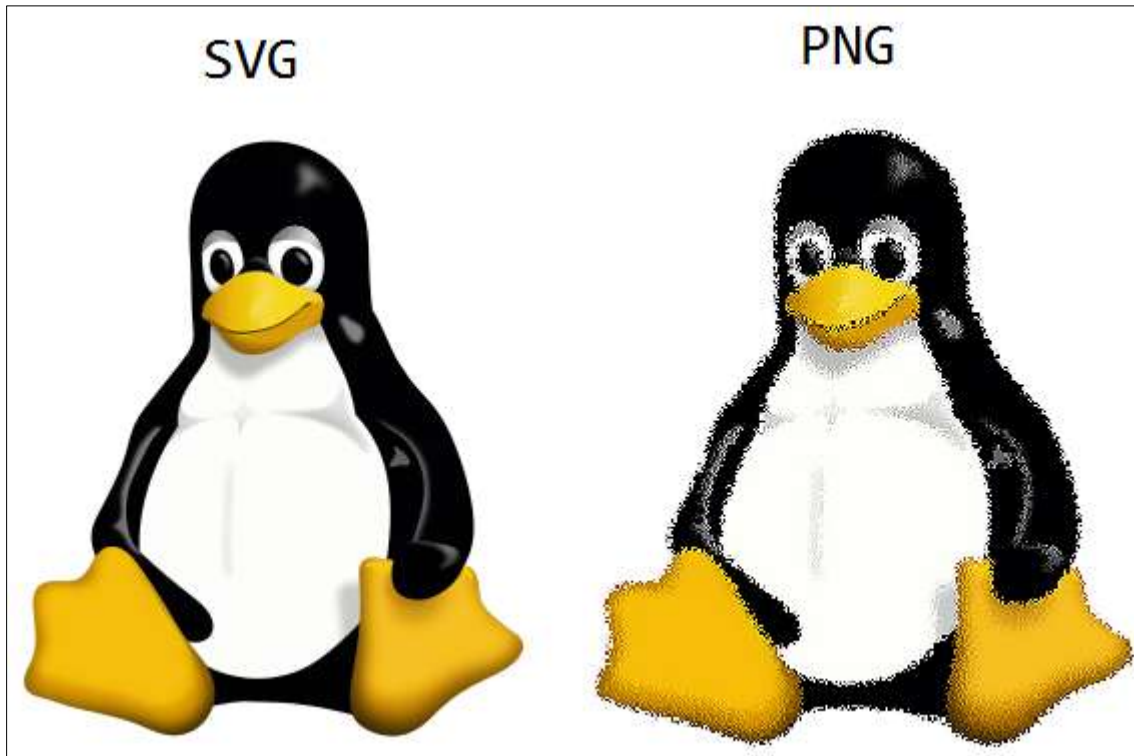
გრაფიკული ელემენტებიდან პირველ რიგში, განვიხილოთ SVG, რომელიც ინგლისურად იშიფრება, როგორც *Scalable Vector Graphics* ანუ მასშტაბირებადი ვექტორული გრაფიკა, რომელიც საშუალებას გვაძლევს, გრაფიკული ელემენტის ხარისხის დაუკარგავად შეიცვალოს მასშტაბი.

SVG-გრაფიკული ელემენტი, რომელიც იქმნება XML ფორმატში, თავის მხრივ, აბრევიატურა იშიფრება **გავრცობადი მარკირებას ენა**- რომელიც გამოიყენება მონაცემთა აღწერისა და ფოკუსირებისათვის.

გრაფიკული ელემენტის დემონსტრირება ნაჩვენებია სურათ 4.1-ში ისეთი ცნობილი ფორმატები, როგორიცაა: *jpg, png, gif* მასშტაბების, გაზრდისას ვნახვით, რომ ხარისხი იკარგება, ავიღოთ **ტუქსი** ანუ **Tux** ასე ქვია *Linux*-სის თილისმას (შემდეგ როგორც ლოგოს), რომლის იდეაც ეკუთვნის ამავე ოპერაციული სისტემის **ლინუს ტორვალდს**, ხოლო სახელი, ჯეიმზ ჰიუზმა მოიგონა და გაშიფრა, როგორც *(T)orvalds (U)ni(X)*, უნდა აღინიშნოს, რომ დღეისობით ტუქსი ლინუქსის აღიარებული სიმბოლო და ლოგოა, ინტერნეტში, როგორც მას წავაწყდებით, ე.ი საუბარია ლინუქსის ოპერაციულ სისტემაზე, ასევე იმასაც დავამატებთ, რომ იგი შემქნილია ლარი იუნგის მიერ 1996 წელს.



ტუქსის სურათი, რომელიც ორი ფორმატისაა SVG და PNG, წარმოდგენილი სურათ 4.1ზე გაზრდილია და ვიზუალურად ვხედავთ, რომ SVG-ფორმატის ტუქსი ისევ ისეა, ხოლო PNG- ფორმატს კი პიქსელები დაკარგული აქვს.



სურ 4.1 ტუქსის მაშტაბირება

SVG -ძირითადად გამოიყენება 2D გრაფიკების ასაგებად, სხვადასხვა X,Y კოორდინატთა სისტემების წარმოსადგენად, ასევე შესაძლებელია ყველა ელემენტისა და ყველა ატრიბუტის ანიმირება, ასევე ხშირად გამოიყენება დიდი რუკის შესადგენად და შემდეგ ZOOM მეთოდით, გადიდება ან დაპატარავება და ეს, ყოველივე ხარისხის დაუკარგავად.

SVG

ვექტორული გრაფიკის ელემენტები განისაზღვრება გეომეტრიული ფიგურებით როგორც ოთხკუთხედი, ელიფსი და სხვა, ხოლო თვითონ SVG ტეგს აქვს ღია და დახურული ტეგები `<svg>...</svg>` იგი ერთგვარი კონტეინერია, რომელშიც ხატავს სხვადასხვა ფიგურას.

SVG ელემენტებში შედის: *rect*, *circle*, *ellipse*, *line*, *polylin* და *polygon*, რა თქმა უნდა, მათ ქვემოთ მაგალითების სახით განვიხილავთ, ცხრილ 4.1-ში მოცემულია ელემენტები თავიანთი ატრიბუტებით.

x = "კოორდინატები", რომელიც განსაზღვრავს კოორდინატთა სისტემაზე X-ღერძის სიგრძეს, როგორც ოთხკუთხედის გვერდს, ხოლო თუ მასთან ერთად არ არის აღწერილი ატრიბუტები, ე.ი ატრიბუტების θ -ს უდროს და მათი ეფექტები არ წარმოჩნდება.

y = "კოორდინატები", რომელიც ანალოგიურად განსაზღვრავს, როგორც X-ღერძი, წარმოჩნდება, როგორც ოთხკუთხედის გვერდი.

width-„სიგრძე“-განისაზღვრება, სხვადასხვა საზომი ერთეულებში, *px*, *em* და სხვ, რომელიც განსაზღვრავს მართკუთხედის სიგანეს.

height-„სიმაღლე“ ასევე განსაზღვრება ნაცნობ სიდიდეებში და წარმოადგენს მართკუთხედის სიგრძეს.

rx და *ry* „სიგრძე“-საზომი PX, EM და სხვა, რომელიც განსაზღვრავს *x* და *y* კოორდინატების ცენტრებს წრეწირზე.

მაგალითი 4.1

```
<!DOCTYPE html>
<html>
<body>
<svg width="400" height="180">
  <rect x="50" y="20" width="150" height="150" style="fill:blue;
stroke:grey; stroke-width:5; fill-opacity:0.1; stroke-opacity:0.9">
</svg>
</body>
</html>
```

შედეგი



რაც შეეხება ახალ ატრიბუტებს, რომელიც მოცემული გვაქვს *fill* ფიგურას ფერებით ავსებს, ხოლო *stroke-width* განსაზღვრავს საზღვრების სისქეს, *stroke* კი საზღვრის ფერს ხოლო *x="50"* სად იმყოფება ოთხკუთხედი იგივეა *50px* მარცხენა კიდიდან, ხოლო *y="20"* კი ზემოდან და უდრის *20px*.

ელიფსის ან წრეწირის მიმართ უნდა ვიცოდეთ შემდეგი ატრიბუტები.

cx – ელიფსის X ღერძის კოორდინატი, საწყისი მნიშვნელობა 0.

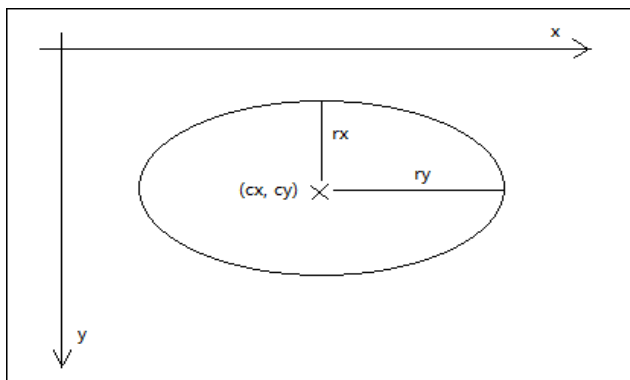
cy – ელიფსის Y ღერძის კოორდინატი, საწყისი მნიშვნელობა.

rx-*x* ღერძის მიმართ რადიუსი.

ry-*y* ღერძის მიმართ რადიუსი.

მაგალითი 4.2

```
<!DOCTYPE html>
<html>
<body>
<svg height="200" width="200">
  <circle cx="80" cy="90" r="60"
stroke="black" stroke-width="3"
fill="green" />
</svg>
<p> ქვემოთ მოცემულია ელიფსი <p>
<svg height="150" width="350">
```

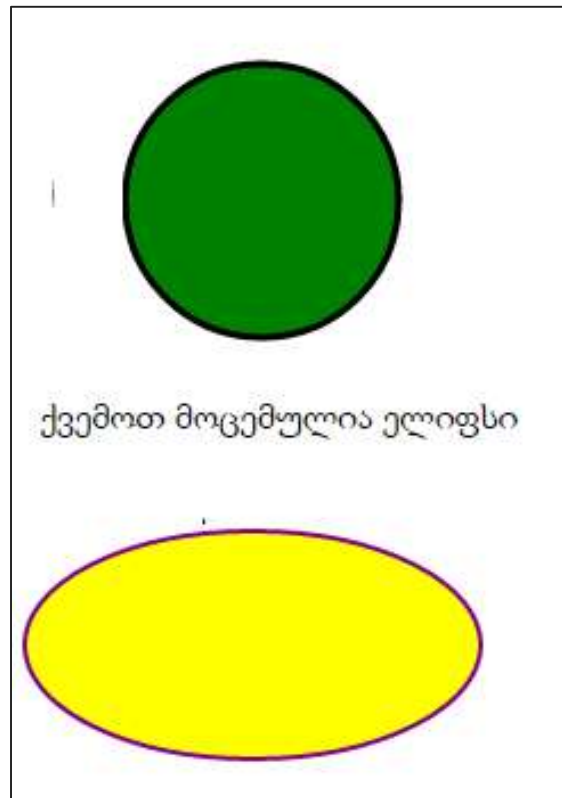


```

<ellipse cx="120" cy="80" rx="100" ry="50" style="fill:yellow;
stroke:purple; stroke-width:2" />
</svg>
</body>
</html>

```

შედეგი



<rect>

ოთხკუთხედის დახატვისას უნდა გავითვალისწინოთ, რომ *rx* და *ry* განსაზღვრავს ფიგურების კუთხეების სიმრგვალებს.

მაგალითი 4.2

```

<!DOCTYPE html>
<html>
<head>
<title>ელიფსის დახატვა</title>
</head>
<body>
  <svg width="400" height="280">
    <rect x="70" y="70" rx="30" ry="30" width="150" height="150"
    style="fill:green;stroke:black;stroke-width:5; opacity:0.7">
  </svg>
</body>
</html>

```


შედეგი



<Line>

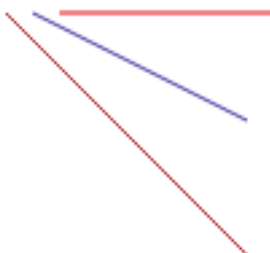
წრფივი ხაზის დახატვა, SVG კომპონენტის ერთ ერთი შემადგენელი ნაწილია, მაგრამ როგორც ყველა ელემენტი მასაც შეუძლია ანიმირება, ამიტომაც მოვიყვანოთ მაგალითი 4.3 ანიმირებული ელემენტი, რომელიც დაგვიხატავს ელემენტებს, მაგრამ მანამდე გავეცნოთ line ელემენტი:

```
<svg height="210" width="500">
<line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);
stroke-width:2" />
</svg>
```

სადაც: $x1$ და $y1$ -არის, საიდანაც უნდა დაიწყოს ხაზის გავლება.
 $x2$ და $y2$ -სადაც, უნდა დასრულდეს ხაზის გავლება.

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
<line x1="0" y1="10" x2="0" y2="100" style="stroke:#006600;"/>
<line x1="10" y1="10" x2="100" y2="100" style="stroke:#B41F22;"/>
<line x1="20" y1="10" x2="100" y2="50" style="stroke:#442CA4;"/>
<line x1="30" y1="10" x2="110" y2="10" style="stroke:#EB0509;"/>
</svg>
```

შედეგი



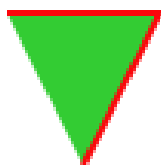
PolyLines

ეს არის წრფივი ხაზი, რომელიც კოორდინატებზე ამდენიმე წერტილს აკავშირებს, რომელის *point*-ები არის ის წერტილები, რითაც უნდა დაკავშირდეს მისი საშუალებით შესაძლებელია, როგორც სხვადასხვა ფიგურების აგება, ისე ანიმაციაც, თითოეული ხაზი არის მრავალ წერიტიალიანი, რომელიც *x* და *y* კოორდინატებით მიიღება.

მაგალითად, ავიღოთ სამკუთხედი, რომლის სამი წვეტი ანუ სამი *point*-ი გვჭირდება რათა ფიგურა დაიხატოს, აქედან გამოდინარე, შიგ არსებული სივარდილე, უნდა შეივსოს:

```
<svg xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <polyline points="10,2 60,2 35,52" style="stroke: #FB070B; stroke-width:
2; fill: #33cc33;"/>
</svg>
```

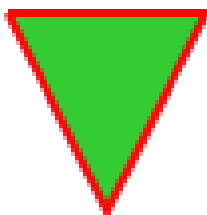
შედეგი



როგორც ვხდევით, ორი გვერდი არის გაწითლებული, აქ მხოლოდ ორი ხაზის კოორდინატები არის მოთავსებული, ხოლო მესამე გვერდზე რომელზედაც არ არის წითელი ხაზი, მანიშნებელია იმისა, რომ მას რეალურად გვერდი არ აქვს, *fill* ატრიბუტის საშუალებით არის შევსებული, შევიტანოთ მესამე ელემენტის მნიშვნელობა:

```
<polyline points="10,2 60,2 35,52 10,2" style="stroke: #FB070B; stroke-
width: 2; fill: #33cc33;"/>
```

დავამატოთ 10,2 კოორდინატები და მივიღებთ:



Style ატრიბუტი ეხმარება, როგორც გვერდის სისქის განსაზღვრაში, ისე ხაზის სტილშიც, მაგალით 4.3-ში მოცემული სკრიპტით იხატება *xbox*-ში, რომელიც არც თუ პატარა სკრიპტია.

მაგალითი 4.3

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>xbox-ის დახატვა</title>
<style>
  Body
{
```

```

    background-color: #fff;
}

img {
    max-width: 80%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
    opacity: 0;
    animation-name: show;
    animation-delay: 6s;
    animation-duration: 1s;
    animation-fill-mode: forwards;
    animation-iteration-count: 1;
    animation-timing-function: linear;
}

svg {
    max-width: 80%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
}

svg path
{
    stroke-width: 2;
    animation-fill-mode: forwards;
    animation-iteration-count: 1;
    animation-name: draw;
    animation-timing-function: linear;
}

svg path#case
{
    stroke: #666;
    stroke-dasharray: 2810;
    stroke-dashoffset: 2810;
    animation-duration: 3s;
}

svg path#cooler
{
    stroke: #ddd;

```

```

    stroke-dasharray: 300;
    stroke-dashoffset: 300;
    animation-duration: 2s;
    animation-delay: 4s;
}
svg path#structure {
    stroke: #999;
    stroke-dasharray: 680;
    stroke-dashoffset: 680;
    animation-duration: 3s;
    animation-delay: 2s;
}

@keyframes draw
{
    to {
        stroke-dashoffset: 0;
    }
}
@keyframes show {
    to {
        opacity: 1;
    }
}

</style>

</head>
<body>
<svg xmlns="http://www.w3.org/2000/svg"
    width="17.7639in" height="6.94444in"
    viewBox="0 0 1279 500">
    <path id="case"
        fill="none"
        d="M 43.00,233.00
          C 43.00,233.00 43.00,192.00 43.00,192.00
            43.00,189.42 42.76,185.61 44.17,183.39
            45.71,180.98 55.12,176.29 58.00,175.49
            58.00,175.49 67.00,174.28 67.00,174.28
            67.00,174.28 149.00,164.13 149.00,164.13
            149.00,164.13 173.00,161.83 173.00,161.83
            173.00,161.83 197.00,158.28 197.00,158.28
            197.00,158.28 214.00,156.72 214.00,156.72
            214.00,156.72 238.00,153.28 238.00,153.28
            238.00,153.28 262.00,150.84 262.00,150.84
            262.00,150.84 287.00,147.28 287.00,147.28
            287.00,147.28 311.00,144.84 311.00,144.84
            311.00,144.84 345.00,140.28 345.00,140.28
            345.00,140.28 403.00,133.71 403.00,133.71
            403.00,133.71 419.00,131.28 419.00,131.28
            419.00,131.28 534.00,117.28 534.00,117.28

```

534.00,117.28 542.00,116.71 542.00,116.71
 542.00,116.71 574.00,112.28 574.00,112.28
 574.00,112.28 599.00,109.72 599.00,109.72
 599.00,109.72 624.00,106.16 624.00,106.16
 624.00,106.16 648.00,103.72 648.00,103.72
 648.00,103.72 672.00,100.16 672.00,100.16
 672.00,100.16 689.00,98.72 689.00,98.72
 689.00,98.72 714.00,95.16 714.00,95.16
 714.00,95.16 731.00,93.71 731.00,93.71
 731.00,93.71 762.00,89.28 762.00,89.28
 762.00,89.28 779.00,87.72 779.00,87.72
 779.00,87.72 802.00,84.28 802.00,84.28
 802.00,84.28 826.00,81.84 826.00,81.84
 831.07,81.29 839.26,79.74 844.00,80.09
 844.00,80.09 873.00,87.12 873.00,87.12
 873.00,87.12 885.00,89.74 885.00,89.74
 885.00,89.74 922.00,99.12 922.00,99.12
 922.00,99.12 1008.00,120.53 1008.00,120.53
 1008.00,120.53 1081.00,138.52 1081.00,138.52
 1081.00,138.52 1141.00,153.04 1141.00,153.04
 1141.00,153.04 1152.00,156.26 1152.00,156.26
 1152.00,156.26 1198.00,167.53 1198.00,167.53
 1198.00,167.53 1209.00,169.89 1209.00,169.89
 1209.00,169.89 1220.00,173.11 1220.00,173.11
 1222.55,173.75 1227.44,174.38 1229.37,175.90
 1234.20,179.72 1234.99,187.38 1235.00,193.00
 1235.00,193.00 1235.00,231.00 1235.00,231.00
 1235.00,231.00 1236.00,248.00 1236.00,248.00
 1236.00,248.00 1236.00,332.00 1236.00,332.00
 1236.00,332.00 1237.00,347.00 1237.00,347.00
 1237.00,347.00 1237.00,382.00 1237.00,382.00
 1236.93,386.73 1235.55,393.16 1231.61,396.26
 1228.62,398.62 1222.72,397.96 1219.00,398.00
 1219.00,398.00 1209.00,398.91 1209.00,398.91
 1209.00,398.91 1172.00,401.96 1172.00,401.96
 1172.00,401.96 1163.00,401.96 1163.00,401.96
 1163.00,401.96 1130.00,404.09 1130.00,404.09
 1130.00,404.09 1061.00,409.00 1061.00,409.00
 1061.00,409.00 1041.00,410.83 1041.00,410.83
 1041.00,410.83 999.00,414.00 999.00,414.00
 999.00,414.00 982.00,415.00 982.00,415.00
 982.00,415.00 957.00,416.83 957.00,416.83
 957.00,416.83 923.00,419.09 923.00,419.09
 923.00,419.09 882.00,421.00 882.00,421.00
 882.00,421.00 867.00,420.00 867.00,420.00
 867.00,420.00 837.00,420.00 837.00,420.00
 837.00,420.00 825.00,419.04 825.00,419.04
 825.00,419.04 808.00,419.04 808.00,419.04
 808.00,419.04 791.00,418.00 791.00,418.00
 791.00,418.00 728.00,416.00 728.00,416.00

728.00,416.00 711.00,415.00 711.00,415.00
 711.00,415.00 686.00,415.00 686.00,415.00
 686.00,415.00 671.00,414.00 671.00,414.00
 671.00,414.00 650.00,414.00 650.00,414.00
 650.00,414.00 635.00,413.00 635.00,413.00
 635.00,413.00 622.00,413.00 622.00,413.00
 622.00,413.00 606.00,412.00 606.00,412.00
 606.00,412.00 570.00,411.00 570.00,411.00
 570.00,411.00 556.00,409.96 556.00,409.96
 556.00,409.96 541.00,409.96 541.00,409.96
 541.00,409.96 529.00,409.00 529.00,409.00
 529.00,409.00 499.00,409.00 499.00,409.00
 499.00,409.00 484.00,408.00 484.00,408.00
 484.00,408.00 468.00,408.00 468.00,408.00
 468.00,408.00 456.00,407.04 456.00,407.04
 456.00,407.04 442.00,407.04 442.00,407.04
 442.00,407.04 427.00,406.00 427.00,406.00
 427.00,406.00 413.00,406.00 413.00,406.00
 413.00,406.00 397.00,405.00 397.00,405.00
 397.00,405.00 363.00,404.00 363.00,404.00
 363.00,404.00 346.00,403.00 346.00,403.00
 346.00,403.00 317.00,403.00 317.00,403.00
 317.00,403.00 305.00,402.04 305.00,402.04
 305.00,402.04 288.00,402.04 288.00,402.04
 288.00,402.04 271.00,400.96 271.00,400.96
 271.00,400.96 259.00,400.96 259.00,400.96
 259.00,400.96 249.00,400.00 249.00,400.00
 249.00,400.00 234.00,400.00 234.00,400.00
 234.00,400.00 221.00,399.00 221.00,399.00
 221.00,399.00 208.00,399.00 208.00,399.00
 208.00,399.00 193.00,398.00 193.00,398.00
 193.00,398.00 164.00,398.00 164.00,398.00
 164.00,398.00 149.00,397.00 149.00,397.00
 149.00,397.00 135.00,397.00 135.00,397.00
 135.00,397.00 118.00,396.00 118.00,396.00
 107.58,395.98 105.48,392.37 97.00,387.17
 97.00,387.17 63.00,365.84 63.00,365.84
 57.52,362.24 43.77,355.82 41.60,350.00
 40.89,348.09 41.00,345.07 41.00,343.00
 41.00,343.00 41.00,327.00 41.00,327.00
 41.00,327.00 42.00,311.00 42.00,311.00
 42.01,305.86 41.31,287.62 42.93,284.02
 44.17,281.27 45.51,280.38 48.00,279.00
 44.84,274.58 44.84,260.42 48.00,256.00
 39.03,250.07 42.89,241.75 43.00,233.00 Z
 M 42.00,182.00
 C 42.00,182.00 41.00,183.00 41.00,183.00
 41.00,183.00 41.00,182.00 41.00,182.00
 41.00,182.00 42.00,182.00 42.00,182.00 Z
 M 42.00,354.00

```

C 42.00,354.00 41.00,355.00 41.00,355.00
  41.00,355.00 41.00,354.00 41.00,354.00
  41.00,354.00 42.00,354.00 42.00,354.00 Z" />
  <path id="cooler"
fill="none"
d="M 856.00,245.00
  C 855.51,250.81 850.11,269.42 847.00,274.00
    847.00,274.00 847.00,245.00 847.00,245.00
    847.00,245.00 856.00,245.00 856.00,245.00 Z
  M 847.00,293.00
  C 847.05,283.64 850.96,274.74 854.00,266.00
    854.00,266.00 858.93,252.00 858.93,252.00
    859.66,249.98 860.16,247.58 862.30,246.80
    864.55,245.57 869.48,246.67 872.00,246.80
    872.00,246.80 864.02,274.00 864.02,274.00
    864.02,274.00 855.72,302.00 855.72,302.00
    855.72,302.00 847.00,328.00 847.00,328.00
    847.00,328.00 847.00,293.00 847.00,293.00 Z
  M 849.57,334.00
  C 849.57,334.00 855.14,315.00 855.14,315.00
    855.14,315.00 868.14,272.00 868.14,272.00
    869.65,267.61 873.34,251.96 875.27,249.33
    877.38,246.47 883.57,247.97 887.00,248.00
    887.00,248.00 877.34,281.00 877.34,281.00
    877.34,281.00 869.98,306.00 869.98,306.00
    869.98,306.00 858.00,346.00 858.00,346.00
    853.85,343.74 848.96,339.16 849.57,334.00 Z
  M 901.00,249.00
  C 901.00,249.00 892.04,281.00 892.04,281.00
    892.04,281.00 878.28,329.00 878.28,329.00
    878.28,329.00 870.00,358.00 870.00,358.00
    859.72,354.33 860.91,348.41 863.41,340.00
    863.41,340.00 866.58,329.00 866.58,329.00
    866.58,329.00 890.00,249.00 890.00,249.00
    890.00,249.00 901.00,249.00 901.00,249.00 Z
  M 872.64,362.00
  C 871.44,358.25 876.27,344.30 877.67,340.00
    877.67,340.00 897.51,274.00 897.51,274.00
    897.51,274.00 902.63,257.00 902.63,257.00
    903.26,254.66 903.79,251.69 906.28,250.81
    908.30,249.74 913.55,250.80 916.00,250.81
    916.00,250.81 892.43,332.00 892.43,332.00
    892.43,332.00 881.00,371.00 881.00,371.00
    877.95,368.81 873.83,365.72 872.64,362.00 Z
  M 884.69,375.00
  C 883.68,371.49 888.76,356.30 890.00,352.00
    890.00,352.00 911.63,278.00 911.63,278.00
    911.63,278.00 916.86,260.00 916.86,260.00
    917.50,257.76 918.43,253.37 920.51,252.06
    922.14,251.12 926.90,251.91 929.00,252.06

```

929.00,252.06 915.42,302.00 915.42,302.00
 915.42,302.00 892.00,383.00 892.00,383.00
 889.21,380.99 885.70,378.49 884.69,375.00 Z
 M 905.97,347.00
 C 905.97,347.00 924.14,283.00 924.14,283.00
 924.14,283.00 930.02,262.00 930.02,262.00
 930.67,259.90 931.90,255.07 933.56,253.88
 935.89,251.93 940.46,253.26 943.00,253.88
 943.00,253.88 934.43,285.00 934.43,285.00
 934.43,285.00 922.57,328.00 922.57,328.00
 922.57,328.00 914.37,357.00 914.37,357.00
 914.37,357.00 908.63,378.00 908.63,378.00
 908.63,378.00 904.58,386.98 904.58,386.98
 904.58,386.98 895.00,388.00 895.00,388.00
 895.00,388.00 905.97,347.00 905.97,347.00 Z
 M 957.00,255.00
 C 957.00,255.00 942.58,307.00 942.58,307.00
 942.58,307.00 928.88,355.00 928.88,355.00
 928.88,355.00 922.85,377.00 922.85,377.00
 922.85,377.00 919.28,386.01 919.28,386.01
 919.28,386.01 909.00,387.00 909.00,387.00
 909.00,387.00 916.26,360.00 916.26,360.00
 916.26,360.00 926.09,325.00 926.09,325.00
 926.09,325.00 934.43,294.00 934.43,294.00
 934.43,294.00 945.00,254.00 945.00,254.00
 945.00,254.00 957.00,255.00 957.00,255.00 Z
 M 969.00,256.00
 C 969.00,256.00 960.88,286.00 960.88,286.00
 960.88,286.00 941.71,356.00 941.71,356.00
 941.71,356.00 935.86,377.00 935.86,377.00
 935.86,377.00 932.30,385.01 932.30,385.01
 932.30,385.01 923.00,386.00 923.00,386.00
 923.00,386.00 929.85,361.00 929.85,361.00
 929.85,361.00 947.97,297.00 947.97,297.00
 947.97,297.00 960.00,255.00 960.00,255.00
 960.00,255.00 969.00,256.00 969.00,256.00 Z
 M 947.97,346.00
 C 947.97,346.00 964.12,286.00 964.12,286.00
 964.12,286.00 969.53,266.00 969.53,266.00
 969.53,266.00 972.56,258.02 972.56,258.02
 972.56,258.02 981.00,257.00 981.00,257.00
 981.00,257.00 971.73,293.00 971.73,293.00
 971.73,293.00 954.89,356.00 954.89,356.00
 954.89,356.00 949.89,375.00 949.89,375.00
 949.89,375.00 946.49,384.40 946.49,384.40
 946.49,384.40 938.00,386.00 938.00,386.00
 938.00,386.00 947.97,346.00 947.97,346.00 Z
 M 993.00,258.00
 C 993.00,258.00 983.42,296.00 983.42,296.00
 983.42,296.00 967.01,356.00 967.01,356.00

967.01,356.00 961.34,377.00 961.34,377.00
 961.34,377.00 958.42,383.98 958.42,383.98
 958.42,383.98 951.00,385.00 951.00,385.00
 951.00,385.00 958.63,355.00 958.63,355.00
 958.63,355.00 977.47,283.00 977.47,283.00
 977.47,283.00 984.00,258.00 984.00,258.00
 984.00,258.00 993.00,258.00 993.00,258.00 Z
 M 970.11,357.00
 C 970.11,357.00 989.12,287.00 989.12,287.00
 989.12,287.00 994.20,268.00 994.20,268.00
 994.20,268.00 997.51,260.01 997.51,260.01
 997.51,260.01 1005.00,260.01 1005.00,260.01
 1005.00,260.01 997.58,288.00 997.58,288.00
 997.58,288.00 979.63,356.00 979.63,356.00
 979.63,356.00 974.52,376.00 974.52,376.00
 974.52,376.00 970.58,383.98 970.58,383.98
 970.58,383.98 963.00,385.00 963.00,385.00
 963.00,385.00 970.11,357.00 970.11,357.00 Z
 M 984.85,348.00
 C 984.85,348.00 1000.65,287.00 1000.65,287.00
 1000.65,287.00 1005.63,269.00 1005.63,269.00
 1005.63,269.00 1008.72,261.00 1008.72,261.00
 1008.72,261.00 1016.00,261.00 1016.00,261.00
 1016.00,261.00 991.87,355.00 991.87,355.00
 991.87,355.00 986.71,375.00 986.71,375.00
 986.71,375.00 983.44,382.98 983.44,382.98
 983.44,382.98 976.00,384.00 976.00,384.00
 976.00,384.00 984.85,348.00 984.85,348.00 Z
 M 1028.00,262.00
 C 1028.00,262.00 1003.29,356.00 1003.29,356.00
 1003.29,356.00 998.42,375.00 998.42,375.00
 998.42,375.00 995.28,383.01 995.28,383.01
 995.28,383.01 988.00,384.00 988.00,384.00
 988.00,384.00 992.15,367.00 992.15,367.00
 992.15,367.00 1006.89,309.00 1006.89,309.00
 1006.89,309.00 1019.00,261.00 1019.00,261.00
 1019.00,261.00 1028.00,262.00 1028.00,262.00 Z
 M 1036.89,263.60
 C 1036.89,263.60 1036.37,271.00 1036.37,271.00
 1036.37,271.00 1032.13,288.00 1032.13,288.00
 1032.13,288.00 1013.88,360.00 1013.88,360.00
 1013.88,360.00 1010.21,375.00 1010.21,375.00
 1010.21,375.00 1007.30,382.01 1007.30,382.01
 1007.30,382.01 1000.00,383.00 1000.00,383.00
 1000.00,383.00 1005.77,360.00 1005.77,360.00
 1005.77,360.00 1016.10,319.00 1016.10,319.00
 1016.10,319.00 1017.75,311.00 1017.75,311.00
 1017.75,311.00 1030.00,262.00 1030.00,262.00
 1030.00,262.00 1036.89,263.60 1036.89,263.60 Z

M 1021.12,344.00
C 1021.12,344.00 1034.42,289.00 1034.42,289.00
1034.42,289.00 1040.26,266.13 1040.26,266.13
1041.53,263.24 1044.89,262.12 1047.53,264.18
1049.55,265.92 1048.15,269.75 1047.53,272.00
1047.53,272.00 1042.27,294.00 1042.27,294.00
1042.27,294.00 1028.53,349.00 1028.53,349.00
1028.53,349.00 1020.00,383.00 1020.00,383.00
1020.00,383.00 1012.00,383.00 1012.00,383.00
1012.00,383.00 1021.12,344.00 1021.12,344.00 Z
M 1060.00,265.00
C 1060.00,265.00 1041.77,338.00 1041.77,338.00
1041.77,338.00 1031.00,382.00 1031.00,382.00
1031.00,382.00 1023.00,383.00 1023.00,383.00
1023.00,383.00 1032.12,344.00 1032.12,344.00
1032.12,344.00 1044.37,295.00 1044.37,295.00
1044.37,295.00 1052.00,264.00 1052.00,264.00
1052.00,264.00 1060.00,265.00 1060.00,265.00 Z
M 1041.13,351.00
C 1041.13,351.00 1056.12,291.00 1056.12,291.00
1056.12,291.00 1060.58,273.00 1060.58,273.00
1060.58,273.00 1063.56,266.01 1063.56,266.01
1063.56,266.01 1070.00,266.01 1070.00,266.01
1070.00,266.01 1064.47,289.00 1064.47,289.00
1064.47,289.00 1050.53,346.00 1050.53,346.00
1050.53,346.00 1042.00,382.00 1042.00,382.00
1042.00,382.00 1034.00,382.00 1034.00,382.00
1034.00,382.00 1041.13,351.00 1041.13,351.00 Z
M 1080.00,267.00
C 1080.00,267.00 1070.47,305.00 1070.47,305.00
1070.47,305.00 1058.37,355.00 1058.37,355.00
1058.37,355.00 1054.00,373.00 1054.00,373.00
1054.00,373.00 1051.27,380.40 1051.27,380.40
1051.27,380.40 1045.00,382.00 1045.00,382.00
1045.00,382.00 1052.63,348.00 1052.63,348.00
1052.63,348.00 1064.65,297.00 1064.65,297.00
1064.65,297.00 1072.00,266.00 1072.00,266.00
1072.00,266.00 1080.00,267.00 1080.00,267.00 Z
M 1089.00,267.00
C 1089.00,267.00 1080.89,303.00 1080.89,303.00
1080.89,303.00 1068.50,357.00 1068.50,357.00
1068.50,357.00 1063.00,381.00 1063.00,381.00
1063.00,381.00 1055.00,382.00 1055.00,382.00
1055.00,382.00 1063.12,346.00 1063.12,346.00
1063.12,346.00 1082.00,267.00 1082.00,267.00
1082.00,267.00 1089.00,267.00 1089.00,267.00 Z
M 1099.00,268.00
C 1099.00,268.00 1091.89,300.00 1091.89,300.00
1091.89,300.00 1078.99,355.00 1078.99,355.00
1078.99,355.00 1074.58,373.00 1074.58,373.00

1074.58,373.00 1071.49,379.98 1071.49,379.98
 1071.49,379.98 1065.00,381.00 1065.00,381.00
 1065.00,381.00 1073.50,345.00 1073.50,345.00
 1073.50,345.00 1081.37,310.00 1081.37,310.00
 1081.37,310.00 1091.00,268.00 1091.00,268.00
 1091.00,268.00 1099.00,268.00 1099.00,268.00 Z
 M 1095.79,294.00
 C 1095.79,294.00 1100.00,276.00 1100.00,276.00
 1100.00,276.00 1102.58,270.02 1102.58,270.02
 1102.58,270.02 1108.00,270.02 1108.00,270.02
 1108.00,270.02 1088.23,354.00 1088.23,354.00
 1088.23,354.00 1084.00,372.00 1084.00,372.00
 1084.00,372.00 1081.27,379.01 1081.27,379.01
 1081.27,379.01 1076.00,380.00 1076.00,380.00
 1076.00,380.00 1095.79,294.00 1095.79,294.00 Z
 M 1116.00,270.00
 C 1116.00,270.00 1109.66,300.00 1109.66,300.00
 1109.66,300.00 1097.42,355.00 1097.42,355.00
 1097.42,355.00 1093.65,372.00 1093.65,372.00
 1093.65,372.00 1090.57,378.98 1090.57,378.98
 1090.57,378.98 1085.00,378.98 1085.00,378.98
 1085.00,378.98 1096.88,330.00 1096.88,330.00
 1096.88,330.00 1111.00,270.00 1111.00,270.00
 1111.00,270.00 1116.00,270.00 1116.00,270.00 Z
 M 1126.00,272.00
 C 1126.00,272.00 1122.01,288.00 1122.01,288.00
 1122.01,288.00 1113.11,329.00 1113.11,329.00
 1113.11,329.00 1102.00,379.00 1102.00,379.00
 1102.00,379.00 1096.00,379.00 1096.00,379.00
 1096.00,379.00 1100.80,355.00 1100.80,355.00
 1100.80,355.00 1112.42,302.00 1112.42,302.00
 1112.42,302.00 1119.00,271.00 1119.00,271.00
 1119.00,271.00 1126.00,272.00 1126.00,272.00 Z
 M 1134.00,272.00
 C 1134.00,272.00 1121.89,327.00 1121.89,327.00
 1121.89,327.00 1116.12,354.00 1116.12,354.00
 1116.12,354.00 1112.27,371.00 1112.27,371.00
 1112.27,371.00 1109.42,377.98 1109.42,377.98
 1109.42,377.98 1104.00,379.00 1104.00,379.00
 1104.00,379.00 1111.89,344.00 1111.89,344.00
 1111.89,344.00 1128.00,272.00 1128.00,272.00
 1128.00,272.00 1134.00,272.00 1134.00,272.00 Z
 M 1123.42,331.00
 C 1123.42,331.00 1131.08,297.00 1131.08,297.00
 1131.08,297.00 1134.88,280.00 1134.88,280.00
 1134.88,280.00 1137.72,272.99 1137.72,272.99
 1137.72,272.99 1143.00,272.00 1143.00,272.00
 1143.00,272.00 1135.42,308.00 1135.42,308.00
 1135.42,308.00 1120.00,379.00 1120.00,379.00
 1120.00,379.00 1113.00,378.00 1113.00,378.00

1113.00,378.00 1123.42,331.00 1123.42,331.00 Z
 M 1130.35,343.00
 C 1130.35,343.00 1140.37,297.00 1140.37,297.00
 1140.37,297.00 1145.98,274.60 1145.98,274.60
 1145.98,274.60 1151.00,273.00 1151.00,273.00
 1151.00,273.00 1145.42,300.00 1145.42,300.00
 1145.42,300.00 1133.65,354.00 1133.65,354.00
 1133.65,354.00 1130.42,369.00 1130.42,369.00
 1130.42,369.00 1128.07,376.40 1128.07,376.40
 1128.07,376.40 1123.00,378.00 1123.00,378.00
 1123.00,378.00 1130.35,343.00 1130.35,343.00 Z
 M 1159.00,274.00
 C 1159.00,274.00 1154.40,299.00 1154.40,299.00
 1154.40,299.00 1142.74,354.00 1142.74,354.00
 1142.74,354.00 1138.87,371.00 1138.87,371.00
 1138.87,371.00 1136.28,377.01 1136.28,377.01
 1136.28,377.01 1131.00,378.00 1131.00,378.00
 1131.00,378.00 1138.77,342.00 1138.77,342.00
 1138.77,342.00 1148.45,296.00 1148.45,296.00
 1148.45,296.00 1153.00,274.00 1153.00,274.00
 1153.00,274.00 1159.00,274.00 1159.00,274.00 Z
 M 1146.58,346.00
 C 1146.58,346.00 1156.37,299.00 1156.37,299.00
 1156.37,299.00 1159.60,284.00 1159.60,284.00
 1159.60,284.00 1161.93,276.60 1161.93,276.60
 1161.93,276.60 1167.00,275.00 1167.00,275.00
 1167.00,275.00 1150.63,354.00 1150.63,354.00
 1150.63,354.00 1145.15,376.40 1145.15,376.40
 1145.15,376.40 1140.00,378.00 1140.00,378.00
 1140.00,378.00 1146.58,346.00 1146.58,346.00 Z
 M 1175.00,276.00
 C 1175.00,276.00 1167.39,314.00 1167.39,314.00
 1167.39,314.00 1160.45,347.00 1160.45,347.00
 1158.70,355.72 1156.67,370.44 1153.00,378.00
 1153.00,378.00 1149.00,378.00 1149.00,378.00
 1149.00,378.00 1148.00,377.00 1148.00,377.00
 1148.00,377.00 1161.42,312.00 1161.42,312.00
 1161.42,312.00 1169.00,275.00 1169.00,275.00
 1169.00,275.00 1175.00,276.00 1175.00,276.00 Z
 M 1182.22,278.31
 C 1182.76,279.93 1181.63,283.29 1181.23,285.00
 1181.23,285.00 1178.12,300.00 1178.12,300.00
 1178.12,300.00 1162.00,377.00 1162.00,377.00
 1162.00,377.00 1156.00,377.00 1156.00,377.00
 1156.00,377.00 1164.21,337.00 1164.21,337.00
 1164.21,337.00 1177.00,276.00 1177.00,276.00
 1178.93,276.16 1181.46,276.04 1182.22,278.31 Z
 M 1167.74,358.00
 C 1167.74,358.00 1179.80,300.00 1179.80,300.00
 1179.80,300.00 1183.12,285.00 1183.12,285.00

1183.12,285.00 1185.01,278.60 1185.01,278.60
 1185.01,278.60 1190.00,277.00 1190.00,277.00
 1190.00,277.00 1184.45,305.00 1184.45,305.00
 1184.45,305.00 1175.79,348.00 1175.79,348.00
 1175.79,348.00 1170.00,377.00 1170.00,377.00
 1170.00,377.00 1164.00,377.00 1164.00,377.00
 1164.00,377.00 1167.74,358.00 1167.74,358.00 Z
 M 1197.00,277.00
 C 1197.00,277.00 1190.67,312.00 1190.67,312.00
 1190.67,312.00 1189.54,320.00 1189.54,320.00
 1189.54,320.00 1182.69,354.00 1182.69,354.00
 1182.69,354.00 1177.28,375.40 1177.28,375.40
 1177.28,375.40 1172.00,377.00 1172.00,377.00
 1172.00,377.00 1176.55,352.00 1176.55,352.00
 1176.55,352.00 1186.21,305.00 1186.21,305.00
 1186.21,305.00 1192.00,277.00 1192.00,277.00
 1192.00,277.00 1197.00,277.00 1197.00,277.00 Z
 M 1194.80,301.00
 C 1194.80,301.00 1200.14,279.57 1200.14,279.57
 1200.14,279.57 1204.00,278.00 1204.00,278.00
 1204.00,278.00 1197.80,312.00 1197.80,312.00
 1197.80,312.00 1185.00,376.00 1185.00,376.00
 1185.00,376.00 1180.00,376.00 1180.00,376.00
 1180.00,376.00 1194.80,301.00 1194.80,301.00 Z
 M 1212.00,279.00
 C 1212.00,279.00 1207.00,304.00 1207.00,304.00
 1207.00,304.00 1198.60,347.00 1198.60,347.00
 1198.60,347.00 1193.00,376.00 1193.00,376.00
 1193.00,376.00 1187.00,376.00 1187.00,376.00
 1187.00,376.00 1199.34,315.00 1199.34,315.00
 1199.34,315.00 1207.00,278.00 1207.00,278.00
 1207.00,278.00 1212.00,279.00 1212.00,279.00 Z
 M 1219.00,280.00
 C 1219.00,280.00 1215.00,299.00 1215.00,299.00
 1215.00,299.00 1199.00,376.00 1199.00,376.00
 1199.00,376.00 1195.00,376.00 1195.00,376.00
 1195.00,376.00 1214.00,280.00 1214.00,280.00
 1214.00,280.00 1219.00,280.00 1219.00,280.00 Z
 M 1225.00,280.00
 C 1225.00,280.00 1221.28,303.00 1221.28,303.00
 1221.28,303.00 1211.26,352.00 1211.26,352.00
 1211.26,352.00 1206.69,373.49 1206.69,373.49
 1206.69,373.49 1203.00,376.00 1203.00,376.00
 1202.11,370.01 1207.04,350.01 1208.58,343.00
 1208.58,343.00 1214.28,313.00 1214.28,313.00
 1214.28,313.00 1221.00,280.00 1221.00,280.00
 1221.00,280.00 1225.00,280.00 1225.00,280.00 Z
 M 1226.00,296.00
 C 1226.00,296.00 1226.00,312.00 1226.00,312.00
 1225.98,324.36 1216.81,360.09 1214.00,375.00

```

1214.00,375.00 1209.00,376.00 1209.00,376.00
1209.00,376.00 1214.55,347.00 1214.55,347.00
1214.55,347.00 1218.21,327.00 1218.21,327.00
1218.21,327.00 1224.00,296.00 1224.00,296.00
1224.00,296.00 1226.00,296.00 1226.00,296.00 Z
M 1222.79,364.00
C 1221.52,369.72 1222.15,373.59 1216.00,375.00
1216.00,375.00 1218.72,362.00 1218.72,362.00
1218.72,362.00 1225.00,332.00 1225.00,332.00
1228.28,340.26 1224.67,355.56 1222.79,364.00 Z
M 1223.00,374.00
C 1223.00,374.00 1225.00,365.00 1225.00,365.00
1226.41,368.54 1227.04,372.32 1223.00,374.00 Z" />
    <path id="structure"
fill="none"
d="M 685.70,223.84
C 685.70,223.84 689.72,213.40 689.72,213.40
689.72,213.40 692.67,208.58 692.67,208.58
692.67,208.58 696.68,209.38 696.68,209.38
696.68,209.38 700.70,211.79 700.70,211.79
700.70,211.79 703.11,215.81 703.11,215.81
703.11,215.81 701.77,219.82 701.77,219.82
701.77,219.82 698.02,225.18 698.02,225.18
698.02,225.18 694.81,234.28 694.81,234.28
694.81,234.28 692.13,243.12 692.13,243.12
692.13,243.12 691.59,250.08 691.59,250.08
691.59,250.08 687.85,245.79 687.85,245.79
687.85,245.79 685.44,238.56 685.44,238.56
685.44,238.56 685.44,230.53 685.44,230.53
685.44,230.53 685.70,223.30 685.70,223.30
685.70,223.30 685.70,223.84 685.70,223.84 Z
M 697.75,203.49
C 697.75,203.49 705.25,199.74 705.25,199.74
705.25,199.74 711.68,198.40 711.68,198.40
711.68,198.40 718.64,198.94 718.64,198.94
718.64,198.94 726.40,202.15 726.40,202.15
726.40,202.15 718.90,203.76 718.90,203.76
718.90,203.76 714.89,205.63 714.89,205.63
714.89,205.63 711.94,208.31 711.94,208.31
711.94,208.31 707.66,205.90 707.66,205.90
707.66,205.90 703.64,204.29 703.64,204.29
703.64,204.29 697.75,203.49 697.75,203.49 Z
M 729.35,204.83
C 729.35,204.83 733.36,208.84 733.36,208.84
733.36,208.84 735.77,214.20 735.77,214.20
735.77,214.20 737.65,219.55 737.65,219.55
737.65,219.55 738.72,224.91 738.72,224.91
738.72,224.91 738.72,230.80 738.72,230.80
738.72,230.80 738.45,235.89 738.45,235.89
738.45,235.89 737.11,240.71 737.11,240.71

```

737.11,240.71 735.24,244.99 735.24,244.99
 735.24,244.99 732.56,247.94 732.83,247.94
 733.10,247.94 732.02,242.04 732.02,242.04
 732.02,242.04 729.88,235.08 729.88,235.08
 729.88,235.08 727.20,229.46 727.20,229.46
 727.20,229.46 724.80,224.64 724.80,224.64
 724.80,224.64 722.39,219.82 722.39,219.82
 722.39,219.82 718.64,214.73 718.64,214.73
 718.64,214.73 721.58,211.25 721.85,211.25
 722.12,211.25 725.87,208.31 725.87,208.31
 725.87,208.31 729.35,204.83 729.35,204.83 Z
 M 710.60,223.30
 C 710.60,223.30 706.05,226.52 706.05,226.52
 706.05,226.52 701.77,231.60 701.77,231.60
 701.77,231.60 698.29,237.23 698.29,237.23
 698.29,237.23 695.61,242.31 695.61,242.31
 695.61,242.31 693.47,248.20 693.47,248.20
 693.47,248.20 692.40,251.95 692.40,251.95
 692.40,251.95 696.95,256.77 696.95,256.77
 696.95,256.77 703.38,259.72 703.38,259.72
 703.38,259.72 708.19,260.52 708.19,260.52
 708.19,260.52 713.01,261.05 713.01,261.05
 713.01,261.05 717.57,259.98 717.57,259.98
 717.57,259.98 722.92,257.84 722.92,257.84
 722.92,257.84 727.74,254.09 727.74,254.09
 727.74,254.09 729.88,251.68 729.88,251.68
 729.88,251.68 729.08,246.60 729.08,246.60
 729.08,246.60 727.20,243.12 727.20,242.85
 727.20,242.58 723.46,236.15 723.46,236.15
 723.46,236.15 717.57,228.66 717.57,228.66
 717.57,228.66 710.60,223.30 710.60,223.30 Z
 M 827.61,93.53
 C 827.61,92.42 827.61,91.84 827.61,91.84
 827.61,91.84 842.07,81.13 842.07,81.13
 842.07,81.13 843.94,87.29 843.94,87.29
 843.94,87.29 843.94,341.91 844.21,341.91
 844.48,341.91 834.84,356.37 834.84,356.37
 834.84,356.37 828.15,352.62 828.15,352.62
 828.15,352.62 827.66,115.89 827.61,93.53 Z
 M 825.74,93.98
 C 825.47,93.98 821.45,94.52 821.45,94.52
 821.45,94.52 816.90,94.52 816.90,94.52
 816.90,94.52 410.99,141.10 410.99,141.10
 410.99,141.10 376.45,144.85 376.45,144.85
 376.45,144.85 358.78,146.46 358.78,146.46
 358.78,146.46 351.29,147.26 351.29,147.26
 351.29,147.26 361.19,139.76 361.19,139.76M 351.02,148.87
 C 351.02,148.87 352.09,352.62 352.09,352.62
 352.09,352.62 826.27,353.70 826.27,353.70M 846.09,343.79
 C 846.09,343.79 894.01,395.46 894.01,395.46

```

894.01,395.46 1234.86,379.40 1234.86,379.40M 846.35,234.28
C 846.35,234.28 1234.32,273.64 1234.32,273.64M 44.45,183.41
C 44.45,183.41 348.88,148.33 348.88,148.33M 43.38,249.54
C 43.38,249.54 349.41,229.19 349.41,229.19M 46.59,257.04
C 46.59,257.04 349.68,238.83 349.68,238.83M 334.69,249.81
C 334.69,249.81 341.91,249.81 341.91,249.81M 334.15,254.09
C 334.15,254.09 341.91,253.83 341.91,253.83M 47.12,277.12
C 47.12,277.12 349.14,262.13 349.14,262.13M 43.64,283.81
C 43.64,283.81 349.95,272.03 349.95,272.03M 43.38,351.29
C 43.38,351.29 349.14,352.62 349.14,352.62M 72.29,259.45
C 72.29,259.45 324.51,243.65 324.51,243.65
324.51,243.65 325.31,258.11 325.31,258.11
325.31,258.11 70.95,272.03 70.95,272.03
70.95,272.03 72.29,259.45 72.29,259.45 Z
M 78.18,265.87
C 78.18,265.87 314.60,251.68 314.60,251.68" />

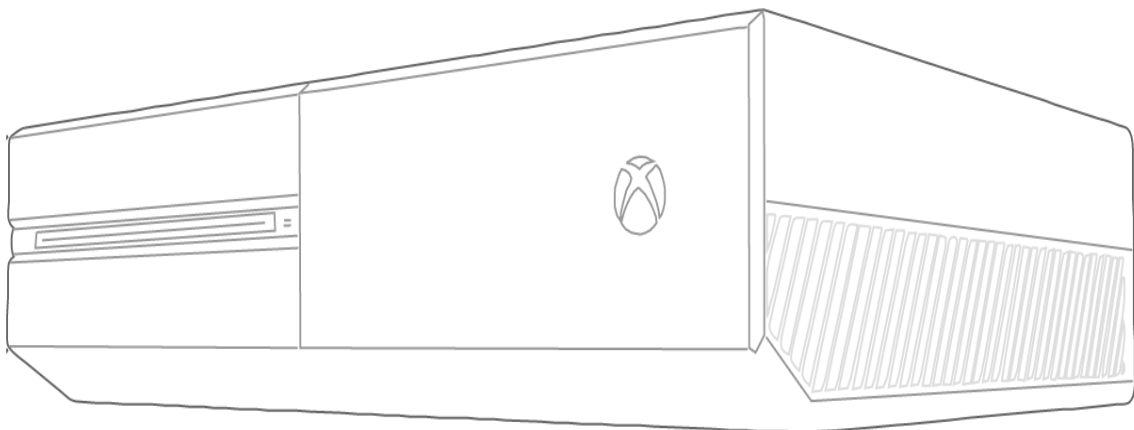
```

```

</svg>
</body>
</html>

```

შედეგი



არსებული სურათი დაიხატება და გრაფიკულად წარმოჩნდება, სკრიპტის ბოლოს შეგიძლიათ ჩაამატოთ `<img>` ტეგი, სადაც ნაჩვენები იქნება ორიგინალი სურათი, ჩვენ წერტილების საშუალებით (ანუ კოორდინატებით) დავხატეთ ის გრაფიკი, რაც გვაქვს მოცემული.

`<path>`

მაგალით 4.3 –ში კოდში ყურადღებას მიაქცევდით ყურადღებას ისეთ ელემენტებს, როგორი ასოებია *M*, *L*, *C* თუ სხვა, ტეგი `<path>` სიტყვიდან გამომდინარედანაც აღნიშნავს გზას/ბილიკს, გასაზღვრავს წერტილების გზას და კუთხეს, თუ როგორც უნდა წარმოჩნდეს, *polyLine*-ს შემთხვევაში დავინახეთ, რომ *path* ელემენტის გამოყენებით შესაძლებელია, როგორც სწორი, ისე მრუდი, დახრილი, ჰორიზონტალური თუ სხვადასხვა ხაზების გავლება, ამიტომაც ყოველივე მათგანი აღვნიშნეთ ასოებით:

M = ნაბიჯი(ერთით დაძვრა)
 L = ხაზი
 H = ჰორიზონტალური ხაზი
 V = ვერტიკალური ხაზი
 C = მრუდი
 S = გლუვი მრუდი
 Q = კვადრატული ბეზიეს მრუდი(Bézier curve მას HTML-ში შევსებთ)
 T = გლუვი კვადრატული ბეზიეს მრუდი
 A = ელიფსური Arc
 Z = ახლო(დამთავრებული, ბოლო წერტილი) გზას

მოცემულია დიდი ასოებით, მაგრამ არსებობს პატარა ასოებითაც აღწერა, დიდი ასოების შემთხვევაში საქმე გვაქვს აბსოლიტურ პოზიციებთან, ხოლო პატარა ასოების შემთხვევაში კი რელევანტურ პოზიციებთან, ხოლო რაც შეეხება *<path>* ატრიბუტებია:

d = "მონაცემები"

განსაზღვრავს კონტურის ფორმას, მისი საშუალებით ანიმაცია შესაძლებელია, თუ მასშია აღწერილია ყველა ელემენტი.

pathLength="რიცხვი"

სადაც აღწერილია ყველა მომხარებლის მიერ გაწერილი სიდიდეები, ხოლო მაგალით 4.4-ში აღწერილია ARC ტიპი, სადაც მოცემულია ზემოთ განხილული ყველა ელემენტი.

მაგალითი 4.4

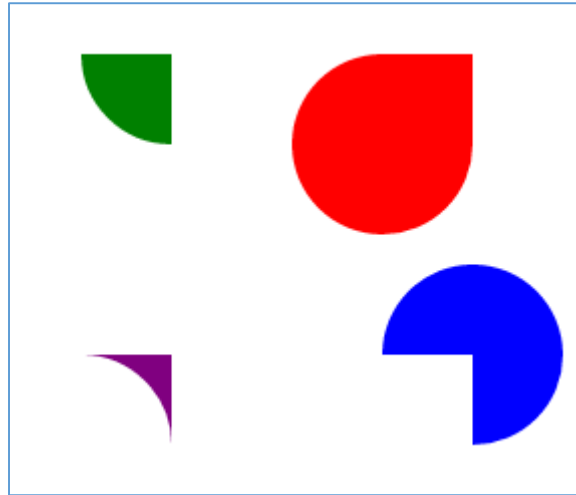
```

<html>
<body>
<svg width="325" height="325" xmlns="http://www.w3.org/2000/svg">
  <path d="M80 80
    A 45 45, 0, 0, 0, 125 125
    L 125 80 Z" fill="green"/>
  <path d="M230 80
    A 45 45, 0, 1, 0, 275 125
    L 275 80 Z" fill="red"/>
  <path d="M80 230
    A 45 45, 0, 0, 1, 125 275
    L 125 230 Z" fill="purple"/>
  <path d="M230 230
    A 45 45, 0, 1, 1, 275 275
    L 275 230 Z" fill="blue"/>
</svg>
</body>
</html>

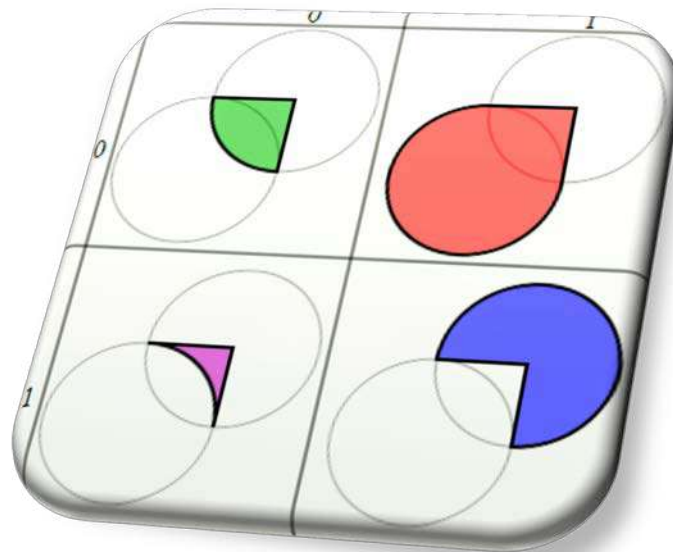
```

შედეგი

მოცემულია ჯერ მაგალითი 4.4-ს, ხოლო შემდეგ წარმოდგენილია ჩვენება იმისა, თუ როგორ იხატება.



გრაფიკულად წარმოდგება:



<text>

როდესაც SVG გარემოში ვლაპარაკობთ ტექსტებზე ჩვენ მოცემული გვაქვს ორი საკითხი, პირველი უშუალოდ განვიხილავთ text-ს, SVG-ში, ხოლო მეორე საკითხად კი გვაქვს SVG ფონტები.

მისი სინტაქსია :

```
<text x="20" y="15" fill="red"> SVG კარგი რამაა </text>
```

როგორც ზემოთ SVG სხვა ელემენტების განხილვიდან ვიცით x და y წარმოდგენს იმ ადგილმდებარეობას, თუ სად უნდა გამოჩნდეს ტექსტი, ხოლო რაც შეეხება SVG ფონტებს, ფონტებთან მიმართებით CSS ელემენტებს ვიყენებთ SVG -ს შემთხვევაში დატოვილია ყველა ის ელემენტი რაც ფონტებთან მიმართებით გვჭირდება და წარმოდგენილია, როგორც SVG ატრიბუტები: *font-family*, *font-style*, *font-*

weight, font-variant, font-stretch, font-size, font-size-adjust, kerning, letter-spacing, word-spacing და *text-decoration*.

ასევე უნდა ითქვას ისიც, რომ *text*-ელემენტს აქვს მონათესავე ელემენტები და ატრიბუტები, რომელებიც ასევე ამუშავებს ტექსტებს და მოიცავს *text* ტეგს.

tspan მოიცავს ტექსტში ქვეტექსტების დაწერას, მაგალითად:

```
<svg height="90" width="200">
  <text x="10" y="20" style="fill:red;">
    მსუბუქი
    <tspan x="20" y="45"> სატვირთო.</tspan>
    <tspan x="30" y="70"> მიკრო ავტობუსი.</tspan>
  </text>
</svg>
```

რაც შეეხება ატრიბუტებს, შედის ისეთი ატრიბუტები, როგორცაა: *x*- რომელიც განსაზღვრავს ახალ კოორდინანტს, სადაც ტექსტი უნდა განთავსდეს, *dx*- იწყებს ჰორიზონტალურად და ანიჭებს მნიშვნელობას *x*-ს, *rotate*-ტექსტს ატრიალებს სხვადასხვა კუთხით, *textLength*- განსაზღვრავს ტექსტის სიგრძეს.

tref

tref - ელემენტი განსაზღვრავს ელემენტს, მისი საშუალებით შესაძლებელია მოხდეს ტექსტის კოპირება-გადატანა ან გამოცხადება სხვა ადგილას, მისი დამხმარე ატრიბუტია *xlink:href*.

```
<text id="magaliti"> მოცემული გვაქვს ტექსტი. </text>
<text>
  <tref xlink:href="#magaliti" />
</text>
```

ფილტრები

SVG ფილტრების საშუალებით შესაძლებელია გამოიყოს გრაფიკულ ელემენტებზე ჩრდილები ან მოხდეს მასზე ზემოქმედება, იმისათვის, რომ ფილტრები ამოვიყენოთ ამისთვის საჭიროა განისაზღვროს *<defs>* ტეგი, ხოლო ID მოხდეს მისი გამოცხადება, რომელიც უნიკალურია.

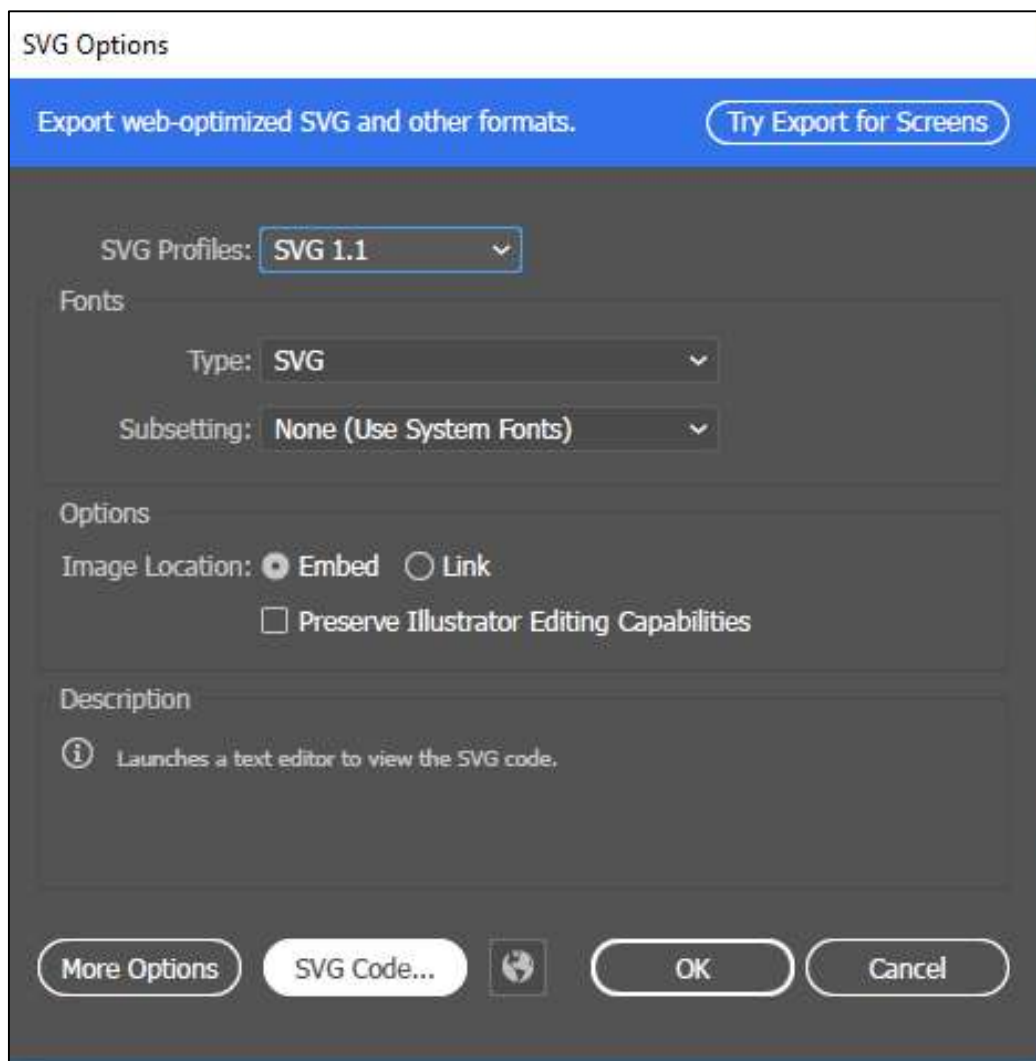
<i>feBlend</i>	<i>feGaussianBlur</i>	<i>feTurbulence</i>
<i>feColorMatrix</i>	<i>feImage</i>	<i>feDistantLight</i>
<i>feComponentTransfer</i>	<i>feMerge</i>	<i>fePointLight</i>
<i>feComposite</i>	<i>feMorphology</i>	<i>feSpotLight</i>
<i>feConvolveMatrix</i>	<i>feOffset</i> - filter for	
<i>feDiffuseLighting</i>	<i>drop shadows</i>	
<i>feDisplacementMap</i>	<i>feSpecularLighting</i>	
<i>feFlood</i>	<i>feTile</i>	

მისი დეკლარირება ხდება შემდეგნაირად:

```
<defs>
  <filter id="სახელი" >
რაიმე ელემენტის გამოცხადება
  </filter>
</defs>
```

SVG -ში სამუშოდ გამოიყენება ისეთი ვექტორული პროგრამები, როგორიცაა: *Skencil*, *Karbon14*, *CorelDraw*, *Inkscape*, *Illustrator* და სხვ.

Adobe Illustrator 2021-ში დავამუშავეთ JS პიქტოგრამა (icon-ი) *Curvature tool*-ს და ფაილის შენახვის დროს მიუთითეთ, SVG ფორმატი, შენახვისას ამოვარდება *SVG Option* ფანჯარა



შემდეგ ვაწვებით SVG Code-ს, სადაც *notepad*-ში (ან სხვა რედაქტორში) ამოვარდება, სადაც იქნება SVG კოდი, კოდში მოვძებნით (ჩვენს შემთხვევაში)

```
<path class="cls-1" d="M79,132a15,15,0,0,0,1.54,47,47..... რომელიც  
მთავრდება .....75578.39,132.92,79,132Z" transform="translate(-64 -103)"/>  
სადაც transform-ელემენტი აღარ გვჭირდება.
```

საქართველოს რუკის დასახატად მაგალითი 4.3-ში ჩავსვით *path* კოორდინატები, რომელნიც *Adobe Illustrator*-ის საშუალებით გავიგეთ.

მაგალითი 4.5

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title> დასაჯა</title>
</style>
  Body
  {
    background-color: #fff;
  }

img {
  max-width: 80%;
  width: 100%;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
  opacity: 0;
  animation-name: show;
  animation-delay: 6s;
  animation-duration: 1s;
  animation-fill-mode: forwards;
  animation-iteration-count: 1;
  animation-timing-function: linear;
}

svg {
  max-width: 80%;
  width: 100%;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}
svg path
{
  stroke-width: 2;
  animation-fill-mode: forwards;
  animation-iteration-count: 1;
  animation-name: draw;
```

```

    animation-timing-function: linear;
}
svg path#gtu
{
    stroke: #666;
    stroke-dasharray: 2810;
    stroke-dashoffset: 2810;
    animation-duration: 3s;
}

svg path#gtu
{
    stroke: #ddd;
    stroke-dasharray: 300;
    stroke-dashoffset: 300;
    animation-duration: 2s;
    animation-delay: 4s;
}
svg path#gtu {
    stroke: #999;
    stroke-dasharray: 680;
    stroke-dashoffset: 680;
    animation-duration: 3s;
    animation-delay: 2s;
}

@keyframes draw
{
    to {
        stroke-dashoffset: 0;
    }
}
@keyframes show {
    to {
        opacity: 1;
    }
}
</style>
</head>
<body>

<svg xmlns=http://www.w3.org/2000/svg
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
viewBox="0 0 56 56" style="enable-background:new 0 0 56 56;"
xml:space="preserve">
    <g>
<path id="gtu" style="fill:none;"
d="M36.985,0H7.963C7.155,0,6.5,0.655,6.5,1.926V55c0,0.345,0.655,1,1.463,1h
40.074c0.808,0,1.463-0.655,1.463-1V12.978c0-0.696-0.093-0.92-0.257-
1.085L37.607,0.257C37.442,0.093,37.218,0,36.985,0z"/>
<polygon style="fill:none;" points="37.5,0.151 37.5,12 49.349,12 "/>

```

```

<path id="gtu" style="fill:none;"
d="M48.037,56H7.963C7.155,56,6.5,55.345,6.5,54.537V39h43v15.537C49.5,55.34
5,48.845,56,48.037,56z"/>
<g>
<path id="gtu" style="fill:none;"
d="M26.021,42.719v7.848c0,0.474-0.087,0.873-0.26,1.196c-0.174,0.323-
0.406,0.583-0.697,0.779c-0.292,0.196-0.627,0.333-1.005,0.41s-0.769,0.116-
1.169,0.116c-0.201,0-0.436-0.021-0.704-0.062s-0.547-0.104-0.834-0.191
s-0.563-0.185-0.827-0.294c-0.265-0.109-0.488-0.232-0.67-0.369l0.697-
1.107c0.091,0.063,0.221,0.13,0.39,0.198
s0.353,0.132,0.554,0.191c0.2,0.06,0.41,0.111,0.629,0.157s0.424,0.068
,0.615,0.068c0.482,0,0.868-0.094,1.155-0.28
s0.439-0.504,0.458-0.95v-7.711H26.021z"/>
<path id="gtu" style="fill:none;" d="M34.184,50.238c0,0.364-0.075,0.718-
0.226,1.06s-0.362,0.643-0.636,0.902s-0.611,0.467-1.012,0.622
c-0.401,0.155-0.857,0.232-1.367,0.232c-0.219,0-0.444-0.012-0.677-
0.034s-0.468-0.062-0.704-0.116
c-0.237-0.055-0.463-0.13-0.677-0.226s-0.399-0.212-0.554-0.349l0.287-
1.176c0.127,0.073,0.289,0.144,0.485,0.212

s0.398,0.132,0.608,0.191c0.209,0.06,0.419,0.107,0.629,0.144c0.209,0.036,0.
405,0.055,0.588,0.055c0.556,0,0.982-0.13,1.278-0.39
s0.444-0.645,0.444-1.155c0-0.31-0.105-0.574-0.314-0.793c-0.21-0.219-
0.472-0.417-0.786-0.595s-0.654-0.355-1.019-0.533
c-0.365-0.178-0.707-0.388-1.025-0.629c-0.319-0.241-0.584-0.526-0.793-
0.854c-0.21-0.328-0.314-0.738-0.314-1.23
c0-0.446,0.082-0.843,0.246-1.189s0.385-0.641,0.663-0.882s0.602-
0.426,0.971-0.554s0.759-0.191,1.169-0.191
c0.419,0,0.843,0.039,1.271,0.116c0.428,0.077,0.774,0.203,1.039,0.376c-
0.055,0.118-0.119,0.248-0.191,0.39
c-0.073,0.142-0.142,0.273-0.205,0.396c-0.064,0.123-0.119,0.226-
0.164,0.308c-0.046,0.082-0.073,0.128-0.082,0.137
c-0.055-0.027-0.116-0.063-0.185-0.109s-0.167-0.091-0.294-0.137c-0.128-
0.046-0.297-0.077-0.506-0.096
c-0.21-0.019-0.479-0.014-0.807,0.014c-0.183,0.019-0.355,0.07-
0.52,0.157s-0.311,0.193-0.438,0.321
c-0.128,0.128-0.229,0.271-0.301,0.431c-0.073,0.159-0.109,0.313-
0.109,0.458c0,0.364,0.104,0.658,0.314,0.882
c0.209,0.224,0.469,0.419,0.779,0.588c0.31,0.169,0.646,0.333,1.012,0.492c0.
364,0.159,0.704,0.354,1.019,0.581
s0.576,0.513,0.786,0.854C34.078,49.261,34.184,49.7,34.184,50.238z"/>
</g>
<g>
<path id="gtu" style="fill:none;" d="M19.5,19v-4c0-0.551,0.448-1,1-
1c0.553,0,1-0.448,1-1s-0.447-1-1-1c-1.654,0-3,1.346-3,3v4c0,1.103-0.897,2-
2,2c-0.553,0-1,0.448
1,1s0.447,1,1,1c1.103,0,2,0.897,2,2v4c0,1.654,1.346,3,3,3c0.553,0,1-
0.448,1-1s-0.447-1-1-1c-0.552,0-1-0.449-1-1v-4c0-1.2-0.542-2.266-1.382-
3C18.958,21.266,19.5,20.2,19.5,19z"/>
<circle style="fill:none;" cx="27.5" cy="18.5" r="1.5"/>

```

შედეგი



მაგალითად, ასე:

მაგალითად :

<canvas id="saxeli" width="200" height="205"> მგოქმეგბა ჩაწეო </canvas>

მაშინ DOM-ის შემთხვევაში იქნება:

```
var saxeli_canvas = document.getElementById("saxeli");
```

DOM იყენებს `document.getElementById()` მეთოდს, ან სურვილისამებრ თქვენ, რომელი მეთოდიც გსურთ, შეგიძლია გამოიყენოთ, მაგრამ გრაფიკის დასახატად მიეთითება 2d, რაც აღნიშნავს ორ განზომილებიან გრაფიკს, ორგანზომილაბიანი გამოსახულებას, რომლის გამოსახისთვის საჭიროა `canvas` ელემენტის `getContext("2d")` მოცემული მიმითითებელია, მოიცავს თვისებებს და მოვლენებს, რომლის საშუალებითაც შესაძლებელია სხვადასხვა ფიგურების დახატვა.

ბრაუზერების მწარმოებლებს, ჰქონდათ მცდელობები, რომ ყოფილიყო შესაძლებელი მათივე ბრაუზერებში 3D განზომილებიანი გრაფიკების აგება, მაგრამ მათი სტანდარტიზაცია არ მოუხდენიათ, ხოლო რაც შეეხება 2D განზომილებიანი ფიგურების კონტექსტის შესაქმნელად საჭიროა გამოყენებული იქნას, ისეთი თვისებები, როგორიცაა `fillStyle`-რომელიც პასუხისმგებელია ფერებზე, სტანდარტული ფერი მისი არის შავი; `fillRect (x,y, სიგანე, სიგრძე)` ავსებს ფიგურას; `moveTo(x, y)` საწყისი კოორდინანტები; `lineTo(x, y)` ხატავს საბოლოო კოორდინანტებს;

ქვემოთ მოცემულია მეთოდი, თუ როგორ შესაძლებელია კოორდინატებზე აიგოს გრაფიკი, და ერთი კოორდინანტებიდან მეორეზე გადასვლა რომელსაც ბილიკი ჰქვია.

ცხრილი 4.1 მოცემული მეთოდებით შესაძლებელია გრაფიკების აგება

მეთოდები	
<code>beginPath()</code>	იწყებს ბილიკს.
<code>moveTo(x, y)</code>	მოცემული კოორდინანტებით ქვე ბილიკი იქმნება.
<code>closePath()</code>	როცა მთავრდება, ქვებილიკი იწყებს ახალ ბილიკზე გადასვლას ახალი კოორდინანტებზე და როცა მთავრდება, ისევ ახალზე გადადის.
<code>fill()</code>	მოცემული მეთოდი ავსებს ქვებილიკებს.
<code>stroke()</code>	განსაზღვრავს ხაზის ქვე ბილიკის სტილს .
<code>arc(x, y, radius, startAngle, endAngle, anticlockwise)</code>	მოცემული მეთოდებით შესაძლებელია აიგოს შესაბამისი ფიგურა, რკალი და სხვ.

ცხრილ 4.1-ში მოცემული მეთოდები უცვლელად რჩება, მხოლოდ იცვლება `arc()` მეთოდი, როცა ვხატავთ:

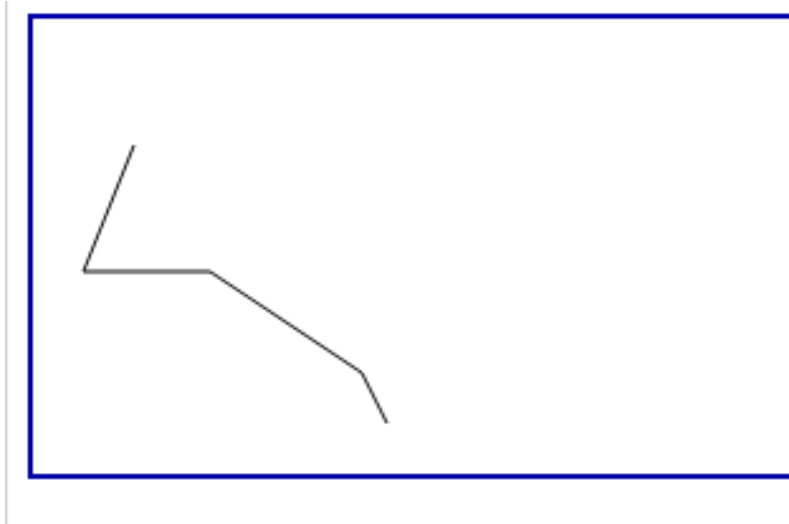
`lineTo(x, y)`-მოცემული მეთოდის დროს შესაძლებელია დამატება კოორდინანტები, ეს ნიშნავს, რომ ხაზი ივლება და მოცემული სკრიპტით იხატება ტეხილი:

```
<canvas id="text" width="300" height="350" style="border:2px solid #0003a7;">
</canvas>
<script>
var es = document.getElementById("text");
var axali = es.getContext("2d");
axali.beginPath();
```

```

axali.moveTo(40, 50);
axali.lineTo(20, 100);
axali.lineTo(70, 100);
axali.lineTo(130, 140);
axali.lineTo(140,160);
axali.stroke();
</script>

```



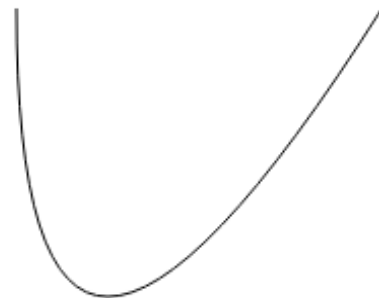
Bezier მრუდს, რომლის კოორდინატებია *bezierCurveTo (cp1x, cp1y, cp2x, cp2y, x, y)* (მასზე ზემოთ გვექონდა ლაპარაკი)

კვადრატული მრუდი-*quadraticCurveTo(cpx, cpy, x, y)*-სადაც *cpx* და *cpy* წარმოადგენს სამართავ წერტილებს *x* და *y* კი საბოლოო კოორდინატებს:

```

var c=document.getElementById('text');
var axali =c.getContext('2d');
axali.beginPath();
axali.moveTo(20,20);
axali.quadraticCurveTo(20,300,200,20);
axali.stroke();

```



მოცემული კოდი დაიხატება მრუდი

სადაც საწყისი წერტილებია *moveTo(20,20);*
სამართავი *quadraticCurveTo (20,300,200,20);*
საბოლოო *quadraticCurveTo (20,300,200,20);*

drawImage(image, dx, dy)- მოცემული მეთოდით იხატება სურათი *canvas* ელემენტში, და სურათზე შეგვიძლია შევცვალოთ კოორდინატები, ზომები, იგი განისაზღვრება კიდევ სამი მეთოდით ანუ:

```

drawImage(image, dx, dy)
drawImage(image, dx, dy, dw, dh)
drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)

```

ცხრილი 4.2 მოცემულია მეთოდები, რომლის საშუალებითაც შესაძლებელია *canvas* ელემენტთან მუშაობა, ცხრილის შემდეგ მოცემულია ფრაქტალური ხის მაგალითი.

ცხრილი 4.2 მოცემულია *canvas* ელემენტის თვისებები და ფუნქციები

canvas ელემენტები

- *canvas*
- *getContext*
- *toDataURL*
- *save*
- *Restore*

ოთხკუთხედის ხატვა და ფუნქციები

- *clearRect*
- *strokeRect*
- *fillRect*
- *drawImage* (სურათის ფუნქცია)

Path(ბილიკის) ფუნქციები

- *beginPath*
- *closePath*
- *moveTo*
- *lineTo*
- *rect*
- *arc*
- *arcTo*
- *ellipse*
- *quadraticCurveTo*
- *bezierCurveTo*
- *stroke*
- *fill*
- *clip*
- *isPointInPath*
- *sPointInStroke*
- *Path2D* ობიექტები

Linedash პარამეტრები

- *getLineDash*
- *setLineDash*
- *LineDashOffset*

ფერების თვისებები და გრადიენტები

- *fillStyle*
- *The strokeStyle*

- *Linear gradients*
- *Radial gradients*
- *Patterns* (*createPattern* ფუნქცია)

გარდაქმნის ფუნქციები

- *translate*
- *rotate*
- *scale*
- *transform*
- *setTransform*

Text თვისებები და ფუნქციები

- *font*
- *textAlign*
- *textBaseLine*
- *fillText*
- *strokeText*
- *measureText*

ჩრდილების თვისებები

- *shadowColor*
- *shadowBlur*
- *shadowOffsetX*
- *shadowOffsetY*

პიქსელების მანიპულაცია

- *ImageData* ობიექტი
- *getImageData* ფუნქცია
- *putImageData* ფუნქცია
- *createImageData* ფუნქცია

სხვა თვისებები

- *linewidth*
- *lineJoin*
- *lineCap*
- *globalAlpha*
- *globalCompositeOperation*

მაგალით 4.6-ში აგებულია ფრაქტალების ხე, რომელზეც მაუსის მიტანით სხვადასხვა ტიპის ფრაქტალად გარდაიქმნება.

მაგალითი 4.6

```
<!DOCTYPE html>
<html>
<head>
    <title>ფრაქტალები ხე</title>
</head>
<body style="margin: 0px;">
<canvas id="canvas" width="600" height="600"/>
<script>
    var canvas = document.getElementById("canvas");
    var c = canvas.getContext("2d");
    var centerX = canvas.width / 2;
    var xissimagle = 100;
    var totissigrdze = 0.75;
    var totiskutxe = 0.27;
    var totissigrme = 10;

    function XISAGEBA(x1, y1, x2, y2, TOTISSIGRDZE, TOTIAngle, depth)
    {
        if(depth == 0)
            return;
        else{
            c.beginPath();
            c.moveTo(x1, y1);
            c.lineTo(x2, y2);
            c.closePath();
            c.stroke();

            TOTISSIGRDZE *= totissigrdze;

            function TOTI(angle)
            {
                var TOTIX2 = x2 + TOTISSIGRDZE * Math.cos(angle);
                var TOTIY2 = y2 + TOTISSIGRDZE * Math.sin(angle);
                XISAGEBA(x2, y2, TOTIX2, TOTIY2, TOTISSIGRDZE,
                    angle, depth - 1);
            }

            // მარჯვენა ტოტი
            TOTI(TOTIAngle + totiskutxe);

            // მარცხენა ტოტი
            TOTI(TOTIAngle - totiskutxe);
        }
    }
}
```

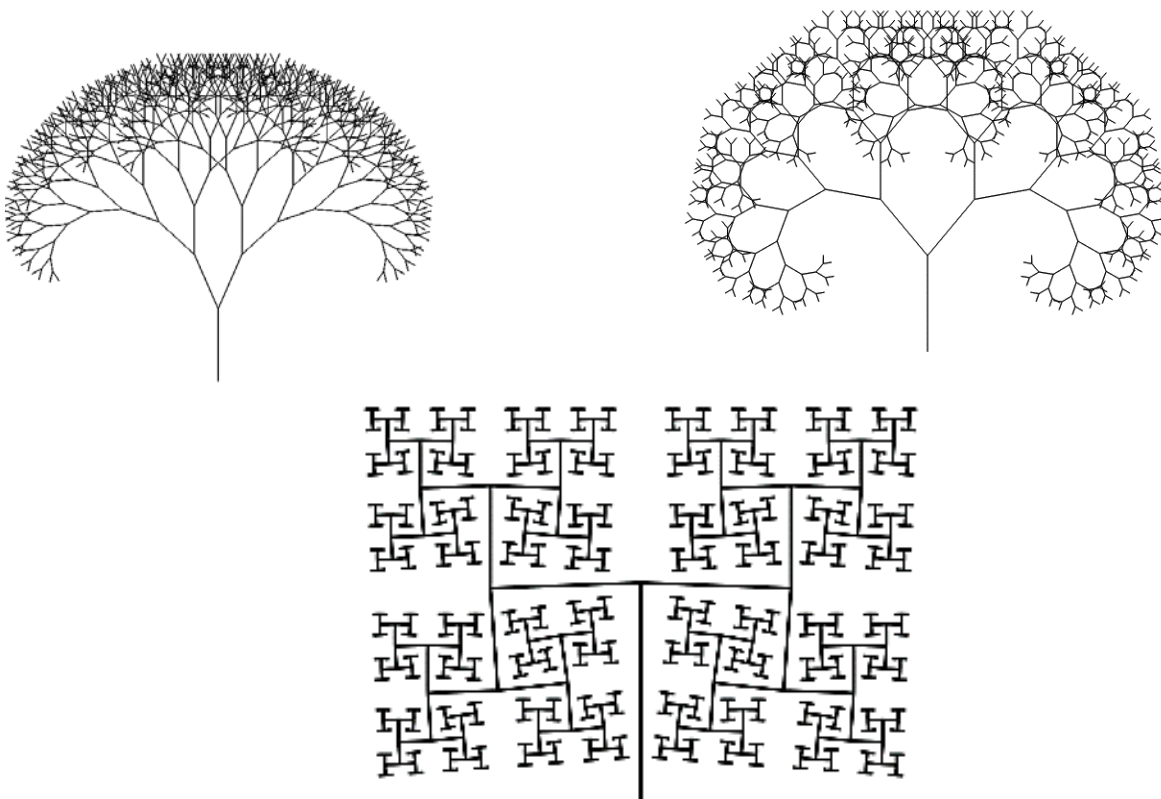
```

function reXISAGEBA()
{
  c.clearRect(0,0, canvas.width, canvas.height);

  var x1 = centerX;
  var y1 = canvas.height;
  var x2 = centerX;
  var y2 = canvas.height - xissimagle;
  XISAGEBA(x1, y1, x2, y2, xissimagle,
    - Math.PI / 2, totissigrme);
}
canvas.addEventListener("mousemove",function(e)
{
  totissigrdze = e.x / 300;
  totiskutxe = e.y / canvas.height * Math.PI;
  reXISAGEBA();
  console.log("totissigrdze = "+totissigrdze);
  console.log("totiskutxe = "+totiskutxe);
});
reXISAGEBA();
</script>
</body>
</html>

```

შედეგები



მაგალითი 4.7 მოცემულია ნიმუში, რომელიც მათემატიკის მატრიცის პრინციპით მოქმედებს, გარდაქმნები, მაგალითად, გვაქვს მოცემული მეთოდი `setTransform(ა, ბ, გ, დ, ე, ვ)` მატრიცულად ის ჩაიწერება:

$$\begin{bmatrix} ა & ბ & გ \\ ბ & დ & ე \\ 0 & 0 & 1 \end{bmatrix}$$

ბოლო ხაზი ყოველთვის `0 0 1` არის

გარდაქმნებისთვის `es.setTransform(1, 0, 0, -1, 0, 0)` გარდაქმნა `y` ღერძის გარშემო, ხოლო მისი მოპირდაპირე კოორდინატები მიხუსით იქნება `es.setTransform(-1, 0, 0, 1, 0, 0)`, ანუ სარკისებური ანარეკლი.

მაგალითი 4.7

```
<html>
<head>
  <title>
    ანარეკლი
  </title>
  <script type="text/javascript">
    var text, ctx;
    function init()
    {
      can = document.getElementById("text");
      ctx = can.getContext("2d");
      ctx.fillStyle = "blue";
      ctx.font = "48pt Helvetica";
      ctx.fillText("canvas მაგალითები", 2, 100);
      ctx.setTransform(1, 0, 0, -1.4, 0, 0);
      ctx.fillStyle = "red";
      ctx.fillText("canvas მაგალითები", -2, -100);
    }
  </script>
</head>
<body onload="init()">
  <canvas id="text" height="800" width="600">
</canvas>
```

</body>

</html>

შედეგი

canvas მაგალითები
canvas ეგზერციზები

Canvas-ს გრაფიკებზე მუშაობა შესაძლებელია სხვადასხვა ბიბლიოთეკით, თამაშების დაწერა, სხვადასხვა ანიმაციის შექმნა, სხვადასხვა გრაფიკის შექმნისას გამოყენება *javascript*-ის *framework*-ები ანუ იმავე ბიბლიოთეკით, რომელიც მზა სახით გვაქვს მოცემული სხვადასხვა ტიპის მეთოდები, ლინკის საშუალებით ხდება მასთან დაკავშირება და გამოყენება.

Drag და Drop (ჩაგდება და გადაადგილება)

Drag და *Drop* (ჩაგდება და გადაადგილება)- ზე ზემოთ ვილაპააკეთ და მოვიყვანეთ მაგალითი 1.29 , სადაც ილიუსტრირება ხდება მოცემული მეთოდის, თუ როგორ არის შესაძლებელია ფაილების გადაადგილება, მაგალითი 4.8

მაგალითი 4.8

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<title> HTML5 Drag და Drop</title>
<script type="text/javascript">
    function dragStart(e)
    {
        e.dataTransfer.effectAllowed = "მოძრაობა";
        e.dataTransfer.setData("Text", e.target.getAttribute("id"));
    }

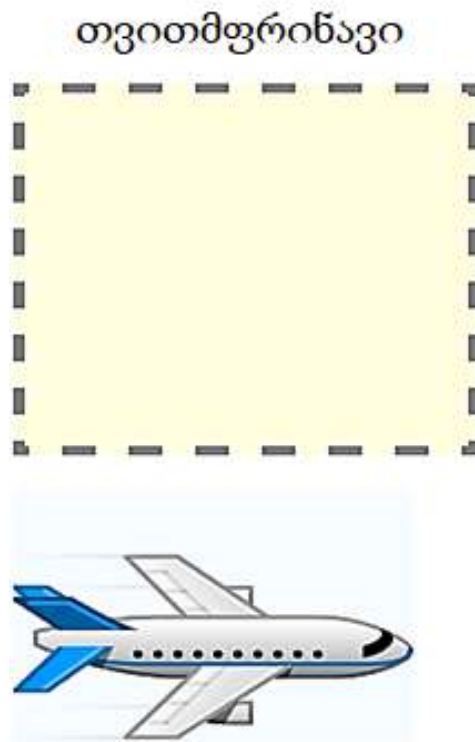
    function dragOver(e)
    {
        e.preventDefault();
        e.stopPropagation();
    }
    function drop(e)
    {
        e.stopPropagation();
        e.preventDefault();
        var data = e.dataTransfer.getData("Text");
        e.target.appendChild(document.getElementById(data));
    }
</script>
<style type="text/css">
    #dropBox
    {
        width: 200px;
        height: 200px;
        border: 5px dashed gray;
```

```

        background: lightyellow;
        text-align: center;
        margin: 20px 0;
        color: orange;
    }
    #dropBox img
    {
        margin: 10px;
    }
</style>
</head>
<body>
    <h2>თვითმფრინავი</h2>
    <div id="dropBox" ondragover="dragOver(event);" ondrop="drop(event);">
    </div>
    
</body>
</html>

```

შედეგი



თვითმფრინავის გადაადგილება შეგვიძლია ოთხკუთხედ გრაფაში, ამით მსგავსი მაგალითის მოყვანა შეგვიძლია უამრავი, აქედან გამომდინარე, მოკლეთ აღვწერთ ძირითად მოვლენებს.

მოვლენები ,რომელიც *drag & drop*-ს აქვს

- ✓ *ondragstart* -როდესაც მომხმარებელი ელემენტში ჩაგდებას იწყებს
- ✓ *ondrag* - როდესაც ელემენტი ვარდება;
- ✓ *ondragend* - როდესაც ელემენტი ჩაგდებას ამთავრებს.;
- ✓ *ondragenter* - მოცემულ მოვლენა აღწერს, პროცეს როდესაც ელემენტი ჩასაგდებ ველში შედის;
- ✓ *ondragover* - მოვლენა ამთავრებს ჩაგდებას;
- ✓ *ondragleave* მოვლენა, როდესაც ტოვებს ჩაგდებულ ელემენტს;
- ✓ *ondrop* -არის მოვლენა, რომელიც ჩაგდებული ელემენტი ჩასაგდებ ელემენტში ვარდება.

DataTransfer ობიექტი, რომელიც თავის მხრივ არის დაკავშირებით მოვლენებთან, აღწერს მათ, მაგალითად, მოვლენა *event.dataTransfer* აბრუნებს *dataTransfer*.

DataTransfer.dropEffect

მის თვისებებში შედის: *none, copy, link ან move*.

DataTransfer.effectAllowed

შედის ყველა ტიპის ოპერატორი, რომელიც: *none, copy, copyLink, copyMove, link, linkMove, move, all*.

DataTransfer.files

მოიცავს ყველა ადგილობრივ ხელმისაწვდომ ფაილს.

***DataTransfer.items* (წაკითხვის რეჟიმი)**

მოცემულია,მეთოდი რომლის საშუალებითაც *DataTransferItemList* გადასადგომ-ლებელი ობიექტების სია.

***DataTransfer.types* (წაკითხვის რეჟიმი)**

ყველა მასივის ელემენტს აქვს მინიჭებული *dragstart*-ის მოვლენა.

dataTransfer.setDragImage(element, x, y)

მოცემული მეთოდით შესაძლებელია განვაახლოთ გაადადგილება

***dataTransfer.getData* (ფორმატი)**

გამოაქვს სპეციალური ფორმატის, თუ არ მოიძებნა რაიმე ინფორმაცია, მაშინ ცარიელ ველს გამოიტანს.

dataTransfer.setData(format, data)

მიანიჭებს სპეციალურ ფორმატს.

dataTransfer.clearData([format])

ასუფთავებს კონკრეტულ ფორმატს

Geolocation (ადგილმდებარეობა)

Geolocation API საშუალებით შესაძლებელია, განისაზღვროს ადგილმდებარეობა, მის ადგილას, როგორც კოორდინატების ისე რუკის მითითებაც შეიძლება, თანამედროვე სამყაროში, ამ აპლიკაციის გამოყენება ძალიან მოსახერხებელია, ხშირად ტელეფონებში ადგილმდებარეობის ფუნქციის გვჭირდება რათა გავიგოთ რაიმეს კოორდინატები, მაგალითად, თუ ჩვენ სად ვიმყოფებით, როგორც სხვა პროგრამებს, მასაც აქვს მეთოდები და მოვლენები, ყველა თანამედროვე ბრაუზერს აქვს თავისი, მხარდაჭერა იმისათვის რომ ვიმუშაოთ მის მეთოდებთან და მოვლენებთან, საჭიროა განისაზღვროს ჯერ ობიექტი, თუ ობიექტის სახელი არის „saxeli“ მაშინ ობიექტი შეიქმნება:

```
var saxeli = navigator.saxeli;
```

Geolocation მეთოდები:

- ✓ *getCurrentPosition*-რომელსაც აქვს თავის მოვლენა (ან თვისება).
- ✓ *showLocation*-ადგილმდებარეობის ჩვენება.
- ✓ *ErrorHandler*-მოიცავს შეცდომის დაფიქსირებას რომელიც იქმნება *PositionError* ობიექტის გამოყენებით
- ✓ *options*-ეს პარამეტრი განსაზღვრავს.
- ✓ *watchPosition*-მეთოდი განსაზღვრავს და განახლებს ადგილმდებარეობას და მომხმარებელს პერიოდულად ინფორმაციას აწვდის. ასევე შესაძლებელია დაიწეროს *Latitude*-განედი და *Longitude*-გრძედის კოორდინატები.
- ✓ *clearWatch*-მეთოდის თვისება არის *watchId*, რომელსაც აქვს უნიკალური ID, მოცემული *watchPosition* მეთოდის მიერ გამოგზავნებულ განახლებას აუქმებს/წყვეტს.

მაგალითი 4.9

მოცემული მაგალითი გვიჩვენებს ჩვენს კოორდინატებს (გრძედისა და განედის) კოორდინატებს.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Geolocation </title>
<body>
<p>
დაწერს გრძედისა და განედის მონაცემებს
</p>
<button onclick="getLocation()">
დააჭირეთ
</button>
<p id="text"></p>
<script>
var es = document.getElementById("text");
```

```

function getLocation()
{
    if (navigator.geolocation)
    {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else
    {
        es.innerHTML ="თქვენს ბრაუზერს არ აქვს Geolocation დადგენის  
მხარდაჭერა";
    }
}
function showPosition(position)
{
    es.innerHTML = "გრძედი: " + position.coords.Latitude +  
    "<br>განედი:" + position.coords.Longitude;
}
</script>
</body>
</html>

```

შედეგი

დაწერს გრძედისა და განედის მონაცემებს

დააჭირეთ

გრძედი: 41.7275184

განედი: 44.7780281

ლოკაციის თვისებები

ცხრილი 4.3 ლოკაციის თვისებები

თვისებები	აღწერა
<i>coords</i>	განსაზღვრავს კოორდინატებს.
<i>coords.Latitude</i>	განედის მონაცემები, რომელიც ათობითში განისაზღვრება მისი სიდიდეც განისაზღვრება $[-90.00, +90.00]$.
<i>coords.Longitude</i>	განსაზღვრავს გრძედის მონაცემებს, რომელიც განისაზღვრება ათობითში მისი არეალია არის $[-180.00, +180.00]$.
<i>coords.altitude</i>	[არასავალდებულო] სპეციფიკური მონაცემების სიმაღლე მეტრებში WGS 84 ელიფსური.
<i>coords.accuracy</i>	[არასავალდებულო] კოორდინატების გრძედისა და განედის განსაზღვრება სწორად მეტრებში

<code>coords.altitudeAccuracy</code>	[არასავალდებულო] კოორდინატების განსაზღვრება მეტრებში
<code>coords.heading</code>	[არასავალდებულო] სხვადასხვა სპეციფიკურ მოწყობილობაში განსაზღვრავს მოძრაობის მიმართულებას კუთხეებში და ითვლის ჩრდილოეთიდან საათის საწინაღმდეგოდ
<code>coords.speed</code>	[არასავალდებულო] განსაზღვრავს მოწყობილობის „მოცემულ“ ადგილზე სიჩქარეს მეტრ/წამში.
<code>timestamp</code>	განსაზღვრავს იმ დროს, როდესაც ადგილმდებარეობა ინფორმაცია მოძიებულია და ობიექტის პოზიცია განთავსებულია.

მაგალითი 4.10

```

<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<title>Geolocation </title>
<body>
<p id="test"> დააჭირეთ ღილაკს.</p>
<button onclick="getLocation()">დააჭირე</button>
<div id="ruka"></div>
<script>
var es = document.getElementById("test");
function getLocation()
{
    if (navigator.geolocation)
    {
        navigator.geolocation.getCurrentPosition(showPosition, showError);
    }
    else
    {
        es.innerHTML = "Geolocation is not supported by this browser.";
    }
}

function showPosition(position)
{
    var latlon = position.coords.latitude + "," + position.coords.longitude;

    var img_url = "https://maps.googleapis.com/maps/api/staticmap?center="+
    +latlon+"&zoom=14&size=500x600&sensor=false&key=AIzaSyBu-
916DdpKAjTmJNIngS6HL_kDIKU0aU";
    document.getElementById("ruka").innerHTML = "<img src='"+img_url+"'>";

```

```

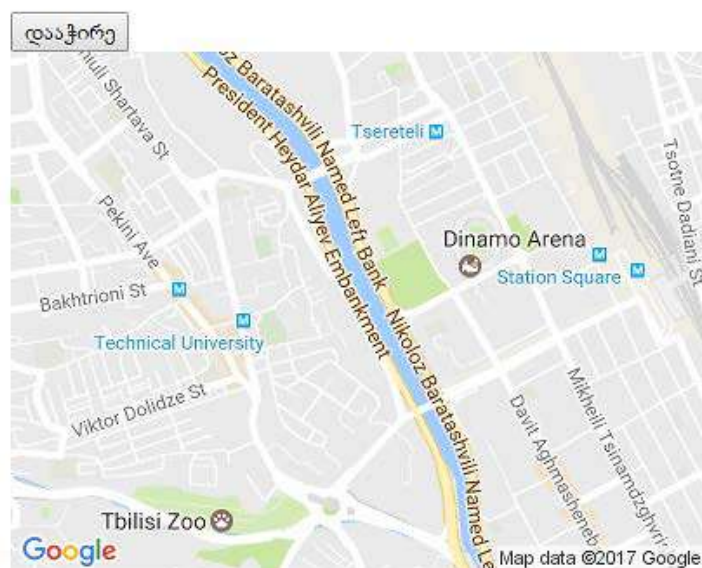
}
function showError(error)
{
    switch(error.code)
    {
        case error.PERMISSION_DENIED:
            es.innerHTML="მომხმარებლის მიერ ადგილმდებარეობა ვერ განისაზღვრა  
ადგილმდებარეობა."

            break;
        case error.POSITION_UNAVAILABLE:
            es.innerHTML = "ვერ დადგინდა ლოკაციის ადგილი "
            break;
        case error.TIMEOUT:
            es.innerHTML = "ლოკაციის ადგილმდებარეობის დადგენის დრო  
გავიდა"
            break;
        case error.UNKNOWN_ERROR:
            es.innerHTML = "დაუდგენელი შეცდომა."
            break;
    }
}
</script>
</body>
</html>

```

შედეგი

დააჭირეთ ღილაკს.



Local Storage

Local storage-ანუ ადგილობრივი სივრცე, ამ ტერმინში იგულისხმება ისეთი ინფორმაციის შენახვა, რომელიც არ იგზავნება სერვერზე და ინახება ფიზიკურ მეხსიერებაში **cookie**-ების საშუალებით, ძირითადად **cookie**-ები, რომელიც ყველა ბრაუზერს აქვს უფრო დაცული არის, ვიდრე პირადი ინფორმაციის პაროლის და სხვა ინფორმაციის გაგზავნა სერვერზე, ამიტომ, როგორც ბრაუზერს, ისე საიტის ოპტიმიზება ხდება.

რა არის cookie?- ეს არის რაღაც ინფორმაცია, რომელიც ინახება ადგილობრივ მეხსიერებაში, მაგალითად თუ შევედით *facebook*-ზე და სახელი პაროლი აქვს დამახსოვრებული, ბრაუზერში საიტის აკრეფის შემდეგ ბრაუზერს გადასცემს დამახსოვრებულ სახელს და პაროლს, რომელიც თავდაპირველად ავკრიფეთ და მივეცით „დამახსოვრებას“, იგი ანალოგიური მიდგომით გადასცემს ყველა საიტზე გარკვეულ ინფორმაციას, რომელიც არ იქნა გადაცემული სერვერზე და შეინახა კომპიუტერში, მაგრამ ინფორმაციის გადაცემის ლიმიტიც განსაზღვრულია, ინფორმაციის მაქსიმალური ტევადობა არის 5MB მეტი, არ გადასცემს სერვერზე (დამოკიდებულია სხვადასხვა ბრაუზერის არქიტექტურაზე 5MB-დან 10MB) და ინახავს ლოკალურად, რაც განსაზღვრავს საიტების ჩატვირთვის სისწრაფეს.

Local storage და *cookie* ორისაც არსებობს სხვაობა, რომ *cookie*-ების შემთხვევაში ინფორმაცია ინახება ტექსტურ რეჟიმში და აწვდის ბრაუზერს შესაბამისად, რომელის ინფორმაციის ტევადობა არა უმეტეს 4 KB არის, ახალი ტექნოლოგიებით საშუალებით HTML5-ში მოველიან *Local Storage*, რომელიც ინფორმაციის შენახვა შესაძლებელია *JavaScript* გავლით, რომელიც მოიცავს მთლიან ვებ გვერდის კონტენტსაც, იგი არ გაუშვებს/წაიკითხვას გვერდს სერვერზე თუ ლოკალურად მის კოპირება არ მოახდინა.

ინფორმაციის შენახვა ყოფენ კიდევ ორ ნაწილად *Session Storage* და *Local Storage*, *Session Storage* არის სივრცე და ადგილი სადაც ინფორმაცია, რომელსაც გახსნილი ბრაუზერი გამოიყენებს მოცემულ მომენტში, ხოლო *Local storage* კი ბრაუზერის დახურვის შემდგომ მუდამ ინახება ინფორმაცია.

- *window.LocalStorage* - ლოკალურად ხდება ინფორმაციის შენახვა
- *window.sessionStorage* - ინფორმაცია ინახება ერთჯერადად და დახურვის შემდგომ-იშლება

ძირითად ობიექტს წარმოადგენს კოდის წერის დროს *storage*-სივრცე, მოცემულ *storage*-ს აქვს პარამეტრი *Length*-სიგრძე და მეთოდები:

```
clear()
getItem()
key()
removeItem()
setItem()
```

მაგალითი 4.11

```
<!DOCTYPE html>
<html Lang="ka">
<head>
<meta charset="UTF-8">
<title>Local Storage</title>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></s
cript>
<script type="text/javascript">
if(localStorage)
{
    $(document).ready(function()
    {
        $(".save").click(function()
        {
            // სახელის დაწერა
            var firstName = $("#firstName").val();

            // მონაცემების შენახვა
            localStorage.setItem("first_name", firstName);
            alert("სახელი არის:");
        });
        $(".access").click(function()
        {
            alert("საღამო " + localStorage.getItem("first_name"));
        });
    });
} else
{
    //შეტყობინება
    alert("ბოდიში თქვენ ბრაუზერს არ აქვს LocalStorage ფუნქციის
მხარდაჭერა.");
}
</script>
</head>
<body>
<form>
<label>ჩაწერეთ სახელი: <input type="text" id="firstName"></label>
<button type="button" class="save">შენახვა</button>
<button type="button" class="access">სახელი</button>
</form>
</body>
</html>
```

შედეგი

ჩაწერეთ სახელი:



Application Cache

Application Cache საშუალებით მომხმარებლებს ეძლევათ საშუალება *offline* რეჟიმში წვდომა ჰქონდეს Web პლიკაციებთან, მისი ატრიბუტი არის *manifest*, რომელიც მოიხსიენება როგორც manifest ფაილად და იგი არის ჩვეულებრივი ტექსტის ფაილი.

ტექსტურ რედაქტორში ქეშირება ხდება ასე, თუ საიტის სახელი არის პირობითად *saiti.ge* მაშინ ქეშირება ტექსტურ რედაქტორში შემდეგნაირად ხდება.

CACHE MANIFEST

v1 - 2017-02-03

es komentaria.

<http://www.saiti.ge/index.html>

<http://www.saiti.ge/header.png>

<http://www.saiti.ge/kategoria>

manifest მოიცავს სამ ნაწილს:

- **CACHE MANIFEST** - ფაილი, სადაც ყოველთვის ახლდება ქეში
- **NETWORK** - შემთხვევაში ითხოვს კავშირს სერვერთან და არასდროს ახდენს ქეშირებას.
- **FALLBACK** - ფაილში არის სია, თუ რა უნდა გაკეთდეს, როცა გვერდი მიწვდომელია *application cache*-საბოლოოდ მომხმარებელს აძლევს სამ შესაძლებლობას:

1. ბრაუზერის საშუალებით *offline* რეჟიმში მომხმარებლებს აქვთ საიტის ნავიგაციასთან წვდომა;

2. დაქეჩილი ფაილები იმყოფება ლოკალურ მეხსიერებაში, აქედან გამომდინარე იტვირთება სწრაფად;
3. ბრუზერი იწერს მხოლოდ იმ განახლებულ რესურს/ფაილებს, რომელიც ცვლილებაც სერვერზე მოხდა

მისი სინტაქსია:

```
<html manifest="URL">
```

მაგალითი 4.12

ჩვენება სამივე შემთხვევაში გვაქვს

```
CACHE MANIFEST
# 2017-02-09
/style.css
/2.gif
/home.js
NETWORK:
Login.asp

FALLBACK:
/public_html/ /offline.html
```

რეალური *cache manifest*, რომელიც *google*-მა დააგენერირა და გამოგვიგზავნა.

*This zip file contains optimized resources for http://scripts.ge/. The optimized resources are listed below in the format of:
filename: url*

Note: We only include up to 10 MB of optimized contents. If the optimized contents of your page are larger than 10 MB, we list them in this file too, with 'NOT INCLUDED' to indicate those URLs require further optimization.

```
css/reset.css: http://scripts.ge/css/reset.css
image/logo.png: http://scripts.ge/images/logo.png
image/bg.png: http://scripts.ge/images/bg.gif
image/search.png: http://scripts.ge/images/search.png
```

WEB WORK

WEB WORK მეთოდით შესაძლებელია საიტზე მიმდინარე პროცესების აღწერა და *Javascript* -ში წარმოდგენა, ხოლო მას **DOM** ობიექტები ყოფს, გამოიყენება სამი ნაწილი: დოკუმენტის ობიექტი, ფანჯრის ობიექტი და მშობლიური ობიექტი.

API სამუშაო ინტერფეისის მეთოდებია:

<i>AbstractWorker</i>	<i>SharedWorkerGlobalScope</i>
<i>ChromeWorker</i>	<i>Worker</i>
<i>DedicatedWorkerGlobalScope</i>	<i>WorkerGlobalScope</i>
<i>ServiceWorker</i>	<i>WorkerLocation</i>
<i>SharedWorker</i>	<i>WorkerNavigator</i>

მაგალითად, ავიღოთ *Worker()* კონსტრუქტორი, რომელიც თავის მხრივ ქმნის *background*-ზე პროცესებს და გასაცემს *work* ობიექტს,

```
var myWorker = new Worker(aURL);
```

aURL არის პარამეტრი, რომელიც *DOMString* ეკუთვნის და იგივე *URL* რომელიც თავის მხრივ, არის ლინკი, სადაც *js* ფაილი არის მითითებული.

მოცემულია UTF-16 არასრული სტრინგი

<i>Attr</i>	<i>DOMUserData</i>	<i>USVString</i>
<i>ByteString</i>	<i>Document</i>	<i>UserDataHandler</i>
<i>CDATASection</i>	<i>DocumentFragment</i>	<i>XMLDocument</i>
<i>CSSPrimitiveValue</i>	<i>DocumentType</i>	
<i>CSSValue</i>	<i>Element</i>	
<i>CSSValueList</i>	<i>ElementTraversal</i>	
<i>CharacterData</i>	<i>Entity</i>	
<i>ChildNode</i>	<i>EntityReference</i>	
<i>Comment</i>	<i>Event</i>	
<i>CustomEvent</i>	<i>EventTarget</i>	
<i>DOMConfiguration</i>	<i>HTMLCollection</i>	
<i>DOMError</i>	<i>MutationObserver</i>	
<i>DOMErrorHandler</i>	<i>Node</i>	
<i>DOMException</i>	<i>NodeFilter</i>	
<i>DOMImplementation</i>	<i>NodeIterator</i>	
<i>DOMImplementationList</i>	<i>NodeList</i>	
<i>DOMImplementationRegistry</i>	<i>NonDocumentTypeChildNode</i>	
<i>y</i>	<i>ProcessingInstruction</i>	
<i>DOMImplementationSource</i>	<i>PromiseResolver</i>	
<i>DOMLocator</i>	<i>Range</i>	
<i>DOMObject</i>	<i>Text</i>	
<i>DOMParser</i>	<i>TextDecoder</i>	
<i>DOMPoint</i>	<i>TextEncoder</i>	
<i>DOMRect</i>	<i>TimeRanges</i>	
<i>DOMTimeStamp</i>	<i>TreeWalker</i>	
<i>DOMTokenList</i>	<i>TypeInfo</i>	

მაგალითი 4.13

```

<html>
<head>
<script type="text/javascript">
function CalculatePi()
{
    var ციკლი = document.getElementById("ციკლი");
    var c = parseInt(ციკლი.value);
    var f = parseFloat(ციკლი.value);
    var Pi=0, n=1;
    try {
        if (isNaN(c) || f != c ) {
            throw("არასწორი რიცხვი");
        } else if (c<=0) {
            throw("უარყოფითი რიცხვი");
        }
        for (var i=0;i<=c;i++)
        {
            Pi=Pi+(4/n)-(4/(n+2));
            n=n+4;
        }
        document.getElementById("pissidide").innerHTML = Pi;
    } catch (e) {
        var msg = "დაფიქსირდა შეცდომა";
        if (e=="არასწორი რიცხვი")
            msg += "არასწორი რიცხვი.";
        else if (e=="უარყოფითი რიცხვი")
            msg += "რიცხვი უნდა იყოს დადებითი";
        else
            msg += e.message;
        alert(msg);
    } } </script>
</head>
<body>
<label for="ციკლი">შეტანილი ციფრი ციკლში:</label>
<input id="ციკლი" type="number" value="100" />
<input type="button" onclick="CalculatePi()" value=" ითვლის π " />
<br><br>
<div id="pissidide">π-ს ანუ პ-ის სიდიდეს გამოიტანს </div>
</body>
</html>

```

შედეგი

3'133320203641314

გადავიყვანო მანძილს მანძილი:

HTML SSE

HTML5 SSE აბრევიატურა და იშიფრება, როგორც *Server sent Event*, რომელიც კომუნიკაციას ამყარებს ვებ გვერდებთან და კომუნიკაციისათვის იყენებს DOM-ის მოვლენების საშუალებით ანდა კონკრეტულად *XMLHttpRequest* ობიექტით, რომელიც ამღევს საშუალებას დაამყაროს კავშირი *WEB* სერვერთან, თუმცა არის საიტები, როდესაც არ ხდება კავშირის მუდამ და ხდება წყვეტა, მაგალითისთვის მოვიყვანოთ ახალი ამბების საიტი ან ფინანსური საიტები, სადაც პერიოდულად ინფორმაციის განახლება ხდება, აახლებს რა ვებ გვერდს და შემდეგ კვლავ წყვეტს მასთან კავშირს, ამაზე პასუხისმგებელია SSE.

Server Sent events (SSE)-Web ტექნოლოგია პირველად მოგვევლინა როგორც *EventSource* და პირველად მის შესახებ დააანონსა *Opera*-ს ბრაუზერმა 2009 წელს, შემდეგ *W3C* გაუწია რეკომენდაცია და ყველა ბრაუზერს აქვს დღეს მისი მხარდაჭერა, უნდა აღინიშნოს მისი მნიშვნელობა, *SSE* გასაგებად უნდა გვესმოდეს *AJAX* სის საზღვრები, პირველად იყო *polling*- ანუ გამოკითხვა, გარკვეული დროის ინტერვალში აკითხავს ბრაუზერი სერვერს, მაგალითად, ყოველ 3 წამში ერთხელ და ნახულობს მოხდა თუ არა ცვლილება, და თუ აღმოჩნდებოდა ახალი ინფორმაცია ბრაუზერი დაბრუნებდა სერვერზე და ანახლებს მონაცემებს, პრობლემა არის ტექნიკა, ყოველჯერ უწევს კავშირი და გახსნა.

მონაცემები, რომელიც იგზავნება *SSE*, არ არის ბინარული მონაცემები არამედ არის ჩვეულებრივი ტექსტური მონაცემი, იმისათვის, რომ მოხდეს სწორი დიალოგური რეჟიმი სერვერსა და კლიენტს შორის, საჭიროა *head*-ში იქნეს გაწერილი სწორად *Content-Type text/event-stream*.

ანუ

```
header("Content-Type: text/event-stream");
```

PHP-ში წარმოდგენილია კოდი

```
<?php
```

```
header('Content-Type: text/event-stream'); header('Cache-Control: no-cache');
```

```
$time = date('r');
```

```
echo "data: The server time is:
```

```
{ $time }\n\n"; flush(); ?>
```

ან კიდევ ერთი

PHP

```

header("Content-Type: text/event-stream");
header("Cache-Control: no-cache"); header("Connection:
keep-alive");

$lastId = $_SERVER["HTTP_LAST_EVENT_ID"]; if (isset($lastId)
&& !empty($lastId) && is_numeric($lastId)) {
    $lastId = intval($lastId);
    $lastId++;
} while (true)
{
    $data = \\ query DB ან სხვა
    რესურსი if ($data) {
        sendMessage($lastId, $data);
        $lastId++;
    }
    sleep(2);
} function sendMessage($id, $data)
{
    echo "id: $id\n";    echo
"data: $data\n\n"; ob_flush();
flush();
}

```

EventSource ობიექტები

onopen-როცა სერვერზე კავშირი დიაა.

onmessage-როცა შეტყობინებაა მოდის

onerror-როცა შეცდომა ფიქსირდება

ლიტერატურა

Beginning HTML & CSS Rob Larsen Published simultaneously in Canada

ISBN: 978-1-118-34018-9

www.rgraph.net

<http://diveinto.html5doctor.com/canvas.html>

<http://www.geekchamp.com/html5-tutorials/11-html5-svg>

<https://www.w3.org/>

<https://www.w3schools.com>

<https://developer.mozilla.org/en-US/docs/.../HTML5>

<https://tutorialrepublic.com>

<http://curran.github.io/>

რედაქტორი ბ. ცხადაძე

გადაეცა წარმოებას 26.06.2023. ხელმოწერილია დასაბეჭდად 19.07.2023. ქაღალდის
ზომა 60X84 1/8. პირობითი ნაბეჭდი თაბახი 19. № 3560

საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“, თბილისი, კოსტავას 77



Verba volant,
scripta manent

