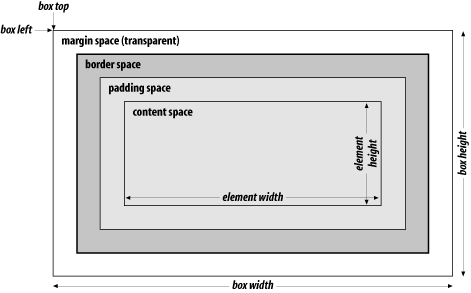
ბლოკისებური მოდელი

ბრაუზერისთვის გვერდის ელემენტები წარმოადგენენ კონტეინერებს ანუ ბლოკებს. ასეთ ბლოკებს შეიძლება ჰქონდეთ სხვადასხვა შიგთავსი - ტექსტი, სურათი, სია, ცხრილი და სხვა. ბლოკის შიდა ელემენტები თვითონაც გამოდიან ბლოკების როლში.

სქემატურად ბლოკისებური მოდელი შეიძლება შემდეგნაირად გამოვსახოთ:



ყველა ელემენტი მოთავსებულია გარკვეულ კონტეინერში. ეს შეიძლება იყოს body, div და ა. შ. სხვა ელემენტებისაგან ის გამოყოფილია ერთგვარი სივრცით - გარე დაშორებით, რომელიც CSS-ში აღიწერება თვისებით margin. ანუ margin თვისება განსაზღვრავს ელემენტის დაშორებას სხვა ელემენტებისგან ან კონტეინერის კიდედან.

ამის შემდეგ მოდის თვითონ ელემენტი, რომელიც იწყება ჩარჩოთი. ჩარჩოს CSS-ში განსაზღვრავს თვისება border.

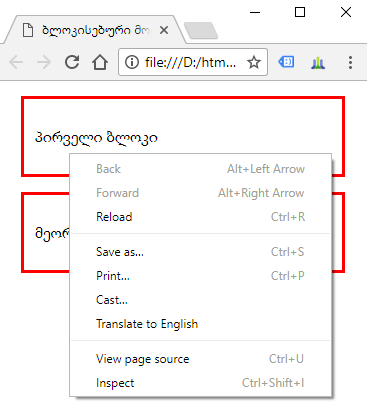
ჩარჩოს შემდეგ მოდის ელემენტის შიდა დაშორება, რომელიც CSS-ში განისაზღვრება თვისებით padding. შიდა დაშორება განსაზღვრავს მანძილს ელემენტის საზღვრიდან მის შიგთავსამდე.

ამის მერე მოდის შიდა ელემენტები, რომელთაც ასევე აქვთ ბლოკისებური მოდელი.

მაგალითად:

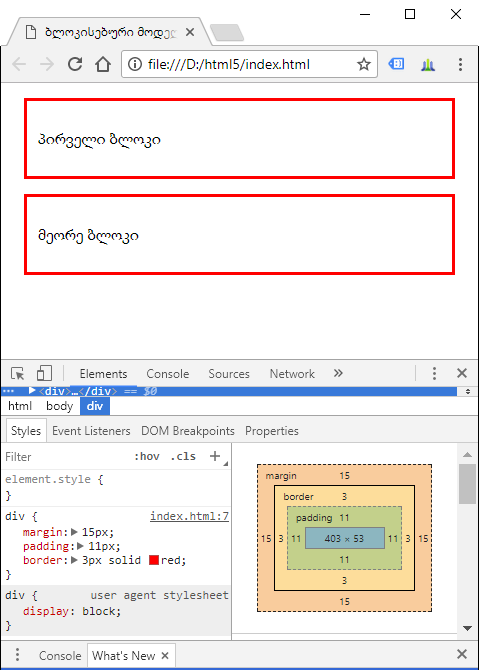
| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ბლოკური მოდელი</title> * <style> * div{ * margin: 15px; /\* გარე დაშორება \*/ * padding: 11px; /\* შიდა დაშორება \*/ * border: 3px solid red; /\* ჩარჩო - 3 პიქსელი სისქის უწყვეტი წითელი ხაზი \*/ * } * </style> * </head> * <body> * <div> * <p>პირველი ბლოკი</p> * </div> * <div> * <p>მეორე ბლოკი</p> * </div> * </body> * </html> |
| --- |

ბრაუზერში გაშვების შემდეგ ჩვენ დავინახავთ კონკრეტული ელემენტების ბლოკისებურ მოდელს. ამისათვის მაუსის მარჯვენა კლავიშით გამოსულ კონტექსტურ მენიუში უნდა ავირჩიოთ შესაბამისი ბრძანება. სხვადასხვა ბრაუზერში ამ ბრძანებას სხვადასხვა სახელი აქვს. Google Chrome-ში უნდა ავირჩიოთ ბრძანება Inspect:



Mozilla Firefox-ში და Microsoft Edge-ში შესაბამის ბრძანებას ჰქვია Inspect element.

ამ ბრძანების შესრულების მერე გაჩნდება პანელი, რომელშიც მოცემულია ელემენტის კოდი, მისი სტილი და ბლოკისებური მოდელი:



თუ არ მივუთითებთ margin, padding და border თვისებების მნიშვნელობებს, ბრაუზერი მათ მიანიჭებს სტანდარტულ მნიშვნელობებს.

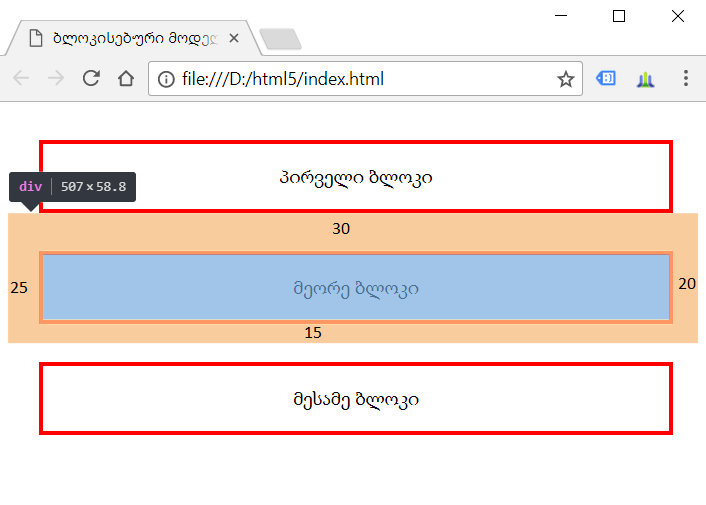
გარე დაშორებები (margin)

margin თვისება განსაზღვრავს ელემენტის დაშორებას სხვა ელემენტებისგან ან კონტეინერის საზღვრებიდან. CSS-ში არსებობს სპეციალური თვისებები თითოეული მხარისათვის:

* margin-top: დაშორება ზემოდან
* margin-bottom: დაშორება ქვემოდან
* margin-left: დაშორება მარცხნიდან
* margin-right: დაშორება მარჯვნიდან

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ბლოკისებური მოდელი</title> * <style> * div{ * margin-top: 30px; /\* დაშორება ზემოდან \*/ * margin-left: 25px; /\* დაშორება მარცხნიდან \*/ * margin-right: 20px; /\* დაშორება მარჯვნიდან \*/ * margin-bottom: 15px; /\* დაშორება ქვემოდან \*/ * border: 3px solid red; /\* საზღვარი \*/ * } * </style> * </head> * <body> * <div> * <p>პირველი ბლოკი</p> * </div> * <div> * <p>მეორე ბლოკი</p> * </div> * <div> * <p>მესამე ბლოკი</p> * </div> * </body> * </html> |
| --- |



შეიძლება 4 თვისების ნაცვლად გამოვიყენოთ ერთი:

| * margin: 30px 20px 15px 25px; |
| --- |

თვისება ენიჭება შემდეგი ფორმატით:

| * margin: დაშორება\_ზემოდან დაშორება\_მარჯვნიდან დაშორება\_ქვემოდან დაშორება\_მარცხნიდან; |
| --- |

თუ ყველა დაშორება ერთნაირია, შეგვიძლია მივუთითოთ მხოლოდ ერთხელ:

| * margin: 25px; |
| --- |

ამ შემთხვევაში დაშორება ყველა მხარეს იქნება 25 პიქსელი.

დაშორების მნიშვნელობა შეიძლება მივუთითოთ პიქსელებში (px), em ერთეულებში, პროცენტული მნიშვნელობებში, ასევე შეიძლება მივუთიტოთ მნიშვნელობა auto - ამ შემთხვევაში დაშორება იქნება სტანდარტული.

მაგალითად:

| * margin: 2em; |
| --- |

მნიშვნელობა 2em ნიშნავს, რომ დაშორება ელემენტის შრიფტის ზომაზე 2-ჯერ მეტია.

პროცენტების გამოყენებისას ბრაუზერი დაშორებას ითვლის იმ კონტეინერის სიგანიდან გამომდინარე, რომელშიც მოქცეულია ელემენტი.

როცა რამდენიმე ელემენტი არის მიმდებარედ, ბრაუზერი დაშორებას ანიჭებს ამ ორი ელემენტის შესაბამისი დაშორებებიდან უდიდესის მნიშვნელობით. ზემოთ მოყვანილ მაგალითში :

| * div{ * margin-top: 30px; /\* დაშორება ზემოდან \*/ * margin-left: 25px; /\* დაშორება მარცხნიდან \*/ * margin-right: 20px; /\* დაშორება მარჯვნიდან \*/ * margin-bottom: 15px; /\* დაშორება ქვემოდან \*/ * } |
| --- |

დაშორება პირველ და მეორე ელემენტს შორის იქნება 30 პიქსელი, რადგან მეორე ელემენტის დაშორება ზემოდან არის 30 პიქსელი და ის მეტია პირველი ელემენტის დაშორებაზე ქვემოდან (15 პიქსელი).

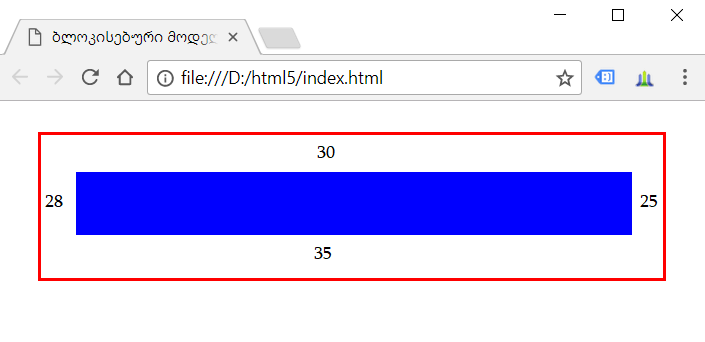
შიდა დაშორებები (padding)

თვისება padding განსაზღვრავს დაშორებებს ელემენტის საქღვრებიდან სიგთავსამდე. ისევე როგორც margin-ისთვის, CSS-ში შიდა დაშორებებისთვისაც არის თვისებები, რომლებიც განსაზღვრავს შიდა დაშორებას თითოეული მხარისთვის:

* padding-top: დაშორება ზემოდან
* padding-bottom: დაშორება ქვემოდან
* padding-left: დაშორება მარცხნიდან
* padding-right: დაშორება მარჯვნიდან

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ბლოკისებური მოდელი</title> * <style> * div.outer{ * margin: 25px; * padding-top:30px; * padding-right: 25px; * padding-bottom: 35px; * padding-left: 28px; * border: 2px solid red; * } * div.inner{ * height: 50px; * background-color:blue; * } * </style> * </head> * <body> * <div class="outer"> * <div class="inner"></div> * </div> * </body> * </html> |
| --- |



ისევე როგორც mangin-ის შემთხვევაში, padding-ის მნიშვნელობა შეიძლება გამოისახოს პროცენტებით (ელემენტის ზომებიდან გამომდინარე).

შიდა დაშორებების ჩაწერაც შეიძლება შემოკლებულად:

| * padding: 30px 25px 35px 28px; |
| --- |

თანმიმდევრობა აქაც იგივეა: დაშორება ზემოდან, დაშორება მარჯვნიდან, დაშორება ქვემოდან, დაშორება მარცხნიდან.

თუ ოთხივე მნიშვნელობა ერთნაირია, შეიძლება მივუთიტოთ მხოლოდ ერთხელ:

| * padding: 25px; |
| --- |

ჩარჩოები

ჩარჩო გამოყოფს ელემენტის შიგთავსს გარემოცვისაგან, ამავე დროს ჩარჩო წარმოადგენს ელემენტის შემადგენელ ნაწილს.

ჩარჩოს კონფიგურაციისთვის გამოიყენება რამდენიმე თვისება:

* border-width: ჩარჩოს სისქე
* border-style: ჩარჩოს სტილი
* border-color: ჩარჩოს ფერი

ჩარჩოს სისქე border-width შეიძლება გამოისახოს px და em ერთეულებით:

| * border-width: 2px; |
| --- |

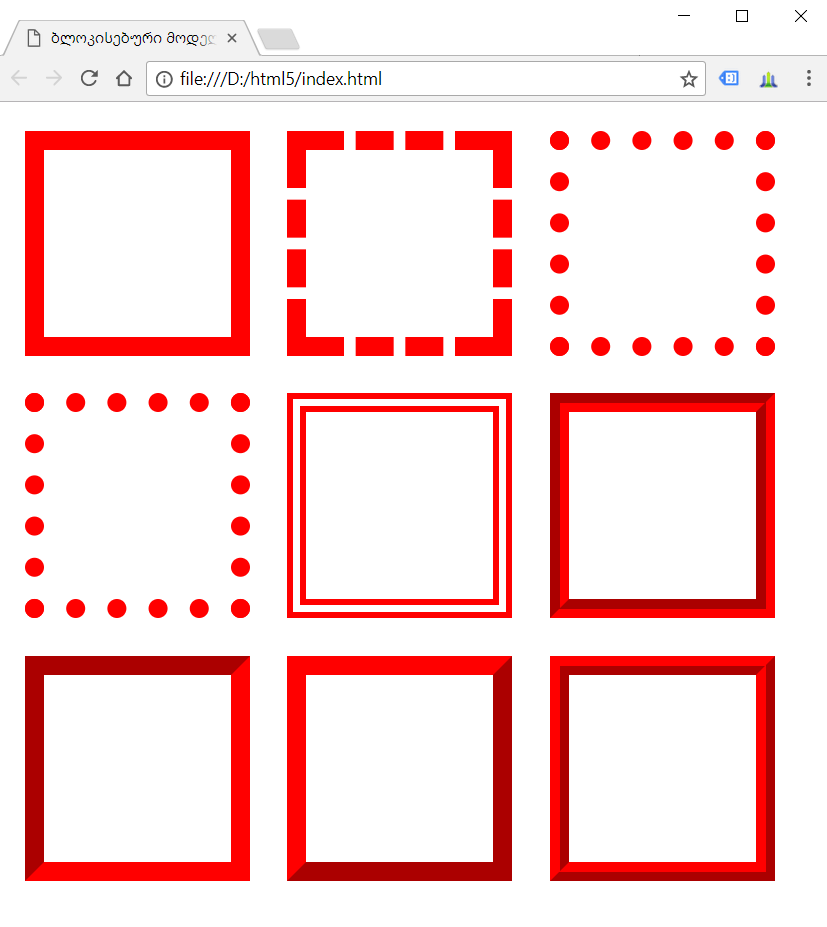
ასევე ჩარჩოს სისზე შეიძლება გამოისახოს კონსტანტებით: thin (თხელი - 1px), medium (საშუალო - 3 px), thick (სქელი - 5px).

თვისება border-color მნიშვნელობად ღებულობს CSS-ის ფერებს:

| * border-color: red; |
| --- |

თვისება border-style განსაზღვრავს ჩარჩოს სტილს. ის ღებულობს შემდეგ მნიშვნელობებს:

* none: ჩარჩოს გარეშე
* solid: ჩარჩო ჩვეულებრივი ხაზის სახით
* dashed: წყვეტილი ხაზი
* dotted: წერტილოვანი ხაზი
* double: წყვილი ხაზი
* groove: ხაზი სამგანზომილებიანი ეფექტით
* inset: ჩაზნექილი ხაზი
* outset: ამოზნექილი ხაზი
* ridge: ასევე სამგანზომილებიანი ეფექტით



საჭიროების შემთხვევაში ჩარჩოს სისქე, სტილი და ფერი შეიძლება განვსაზღვროთ თითოეული მხარისთვის ცალ-ცალკე:

| * /\* ზედა ჩარჩო \*/ * border-top-width * border-top-style * border-top-color * /\* ქვედა ჩარჩო \*/ * border-bottom-width * border-bottom-style * border-bottom-color * /\* მარცხენა ჩარჩო \*/ * border-left-width * border-left-style * border-left-color * /\* მარჯვენა ჩარჩო \*/ * border-right-width * border-right-style * border-right-color |
| --- |

თვისება border

ცალ-ცალკე თვისებების ნაცვლად შეიძლება გამოვიყენოთ ერთი თვისება border:

| * border: სისქე სტილი ფერი; |
| --- |

მაგალითად:

| * border: 2px solid red; |
| --- |

ცალ-ცალკე მხარეებისთვის გამოიყენება თვისებები:

* border-top
* border-bottom
* border-left
* border-right

მაგალითად:

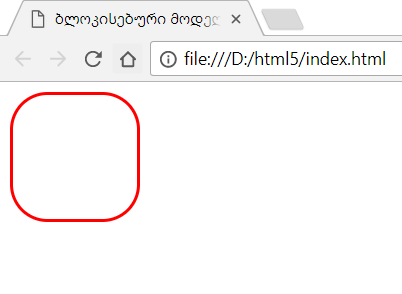
| * border-top: 2px solid red; |
| --- |

ჩარჩოს რადიუსი

თვისება border-radius საშუალებას იძლევა მომრგვალდეს ჩარჩოს კუთხეები. იგი ღებულობს მნიშვნელობას px ან em ერთეულებში. მაგალითად:

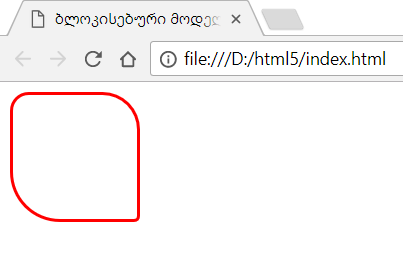
| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ბლოკისებური მოდელი</title> * <style> * div{ * width: 100px; * height:100px; * border: 2px solid red; * border-radius: 30px; * } * </style> * </head> * <body> * <div></div> * </body> * </html> |
| --- |

ახლა ჩარჩოს ყველა კუთხე იქნება მომრგვალებული 30 პიქსელიანი რადიუსით:



რადიუსი შეიძლება განისაზღვროს თითოეული კუთხისთვის ცალ-ცალკე. ამისთვის border-radius მიიღებს 4 მნიშვნელობას (ზედა მარცხენა, ზედა მარჯვენა, ქვედა მარჯვენა, ქვედა მარცხენა):

| * border-radius: 15px 30px 5px 40px; |
| --- |



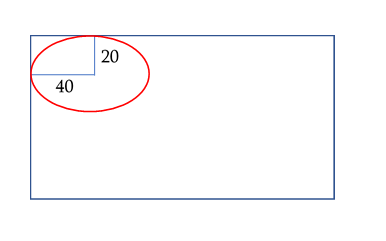
საერთო მნიშვნელობის ნაცვლად შეიძლება მივანიჭოთ მნიშვნელობები ცალ-ცალკე:

| * border-top-left-radius: 15px; * border-top-right-radius: 30px; * border-bottom-right-radius: 5px; * border-bottom-left-radius: 40px; |
| --- |

შესაძლებელია კუთხე მომრგვალდეს ოვალურად, რისთვისაც გამოიყენება ორი რადიუსი:

| * border-radius: 40px/20px; |
| --- |

ამ დროს იგულისხმება, რომ X ღერძის მიმართ რადიუსი იქნება 40 პიქსელი, ხოლო Y ღერძის მიმართ - 20 პიქსელი:



ელემენტის ზომები

ელემენტებს ზომები ენიჭება თვისებებით width (სიგანე) და height (სიმაღლე).

მიუთითებლობისას ამ თვისებების მნიშვნელობაა auto, ანუ ბრაუზერი ელემენტს ანიჭებს ამ ელემენტისთვის განკუთვნილ სტანდარტულ მნიშვნელობას. სიგანე და სიმაღლე გამოისახება ზომების სტანდარტული ერთეულებით: px და em, ასევე პროცენტებით:

| * width: 150px; * width: 75%; * height: 15em; |
| --- |

px - პიქსელი განსაზღვრავს ზუსტ სიგანეს და სიმაღლეს. em დამოკიდებულია ელემენტის შრიფტის ზომაზე - თუ შრიფტის ზომა 16 პიქსელია და ელემენტის ზომას მივუთითებთ 15em, ეს ნიშნავს რომ ზომა იქნება 15\*16=240 px. თუ ელემენტს შრიფტის ზომა არა აქვს განსაზღვრული, აიღება მემკვიდრეობით მიღებული ზომა ან ზომა მიუთითებლობისას.

პროცენტული მნიშვნელობა ნიშნავს, რომ ზომა გამოითვლება კონტეინერის ზომიდან პროცენტულად. მაგალითად, თუ body ელემენტის სიგანე 1000 პიქსელია, ხოლო მასში div ელემენტს სიგანე აქვს 75 %, ეს ნიშნავს, რომ ელემენტის სიგანე იქნება 1000\*75/100=750 პიქსელი. თუ ბრაუზერის ფანჯრის სიგანეს შევცვლით, პროპორციულად შიცვლება div ელემენტის ზომა. ზუსტად ანალოგიურად ითვლება სიმაღლე პროცენტული მნიშვნელობისას - ოღონდ ამ შემთხვევაში კონტეინერის სიმაღლე მრავლდება მითითებულ პროცენტულ მნიშვნელობაზე.

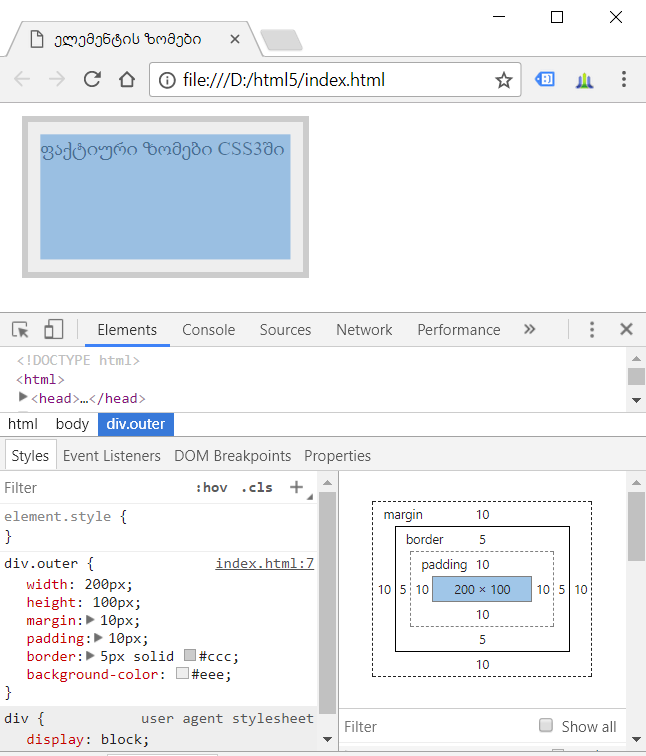
მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ზომები</title> * <style> * div.outer{ * width: 75%; * height: 200px; * margin: 10px; * border: 1px solid #ccc; * background-color: #eee; * } * div.inner{ * width: 80%; * height: 80%; * margin: auto; * border: 1px solid red; * background-color: blue; * } * </style> * </head> * <body> * <div class="outer"> * <div class="inner"></div> * </div> * </body> * </html> |
| --- |



ამავე დროს, ელემენტის ფაქტიური ზომები შეიძლება განსხვავდებოდეს width და height თვისებების მნიშვნელობებისაგან. მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ზომები</title> * <style> * div.outer{ * width: 200px; * height: 100px; * margin: 10px; * padding: 10px; * border: 5px solid #ccc; * background-color: #eee; * } * </style> * </head> * <body> * <div class="outer"> * ფაქტიური ზომები CSS3ში * </div> * </body> * </html> |
| --- |



როგორც სკრინშოტიდან ჩანს, თვისება width განსაზღვრავს ელემენტის შიგთავსს, ხოლო მთლიანად ბლოკისთვის გამოყოფილი ადგილი შედგება შიგთავსს (width) + შიდა დაშორება (padding) + ჩარჩოს სისქე (border-width) + გარე დაშორება (margin). შესაბამისად, თვითონ ელემენტის სიგანე ტოლი იქნება არა 200, არამედ 230 პიქსელისა ჩარჩოების ჩათვლით, ხოლო ელემენტისთვის გამოყოფილი სიგანე გარე დაშორების ჩათვლით იქნება 250 პიქსელი. ანალოგიურად გამოითვლება სიმაღლეც.

დამატებითი თვისებების მეშვეობით შეიძლება შეიზღუდოს ელემენტის მინიმალური და მაქსიმალური ზომები:

* min-width: მინიმალური სიგანე
* max-width: მაქსიმალური სიგანე
* min-height: მინიმალური სიმაღლე
* max-height: მაქსიმალური სიმაღლე

| * min-width: 200px; * width:50%; * max-width: 300px; |
| --- |

ამ შემთხვევაში ელემენტის სიგანე მისი შემცველი კონტეინერის 50 %-ია, მაგრამ არანაკლებ 200 პიქსელისა და არაუმეტეს300 პიქსელისა.

თვისება box-sizing

box-sizing თვისების მეშვეობით შეიძლება შევცვალოთ ელემენტის ზომების დათვლის წესი. მან შეიძლება მიიღოს შემდეგი მნიშვნელობები:

* content-box: მნიშვნელობა მიუთითებლობისას, რომლის დროსაც ელემენტის რეალური ზომა შედგება width და height, მნიშვნელობებს დამატებული შიდა დაშორება და ჩარჩოს სისქე.
* padding-box: ამ დროს width და height მნიშვნელობები უკვე შეიცავენ შიდა დაშორებას და რეალური ზომის დასათვლელად ემატება მხოლოდ ჩარჩოს სისქე.
* border-box: ამ დროს width და height მნიშვნელობები უკვე შეიცავენ როგორც შიდა დაშორებებს, ასევე ჩარჩოს სისქესაც. შესაბამისად, width და height მნიშვნელობები შეადგენენ ელემენტის რეალურ ზომებს.

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ზომები</title> * <style> * div{ * width: 200px; * height: 100px; * margin: 10px; * padding: 10px; * border: 5px solid #ccc; * background-color: #eee; * } * div.outer1{ * box-sizing: content-box; * } * div.outer2{ * box-sizing: border-box; * } * </style> * </head> * <body> * <div class="outer1"> * ელემენტის რეალური ზომები CSS3-ში * </div> * <div class="outer2"> * ელემენტის რეალური ზომები CSS3-ში * </div> * </body> * </html> |
| --- |

პირველ შემთხვევაში ზომებს ემატება შიდა დაშორება და ჩარჩოს სისქე ამიტომ მისი ზომა უფრო დიდია:



ელემენტის ფონი

ელემენტის ფონი CSS-ში აღიწერება თვისებით background. ის ფაქტობრივად, ის წარმოადგენს შემდეგი თვისებების შემოკლებულ ვარიანტს:

* background-color: ფონის ფერი

| * background-color: #ff0507; |
| --- |

* background-image: ელემენტის ფონად გამოიყენებს სურათს

| * background-image: url(dubi.png); |
| --- |

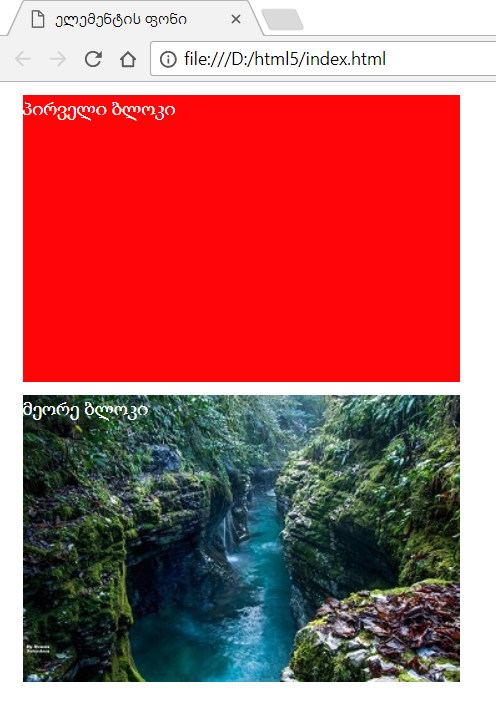
იგი ღებულობს ერთ მნიშვნელობას - url, რომელსაც მოსდევს ფრჩხილებში ჩასმული ფაილის მისამართი. მოცემულ შემთხვევაში იგულისხმება, რომ ვებგვერდთან ერთად იმავე საქაღალდეში არის ფაილი dubi.png. შეიძლება მიეთითოს ფაილის როგორც სრული მისამართი , ასევე ფარდობითი მისამართი:

| * background-image: url(http://mysite.ge/files/dubi.png); * background-image: url(../files/dubi.png); |
| --- |

* background-repeat: ფონური გამოსახულების განმეორების რეჟიმი
* background-size: ფონური გამოსახულების ზომა
* background-position: ფონური გამოსახულების მდებარეობა
* background-attachment: აწესრიგებს ფონური გამოსახულების ელემენტთან მიბმის რეჟიმს
* background-clip: განსაზღვრავს არეს, რომელიც ამოიჭრება სურათიდან და გამოიყენება ფონის სახით
* background-origin: განსაზღვრავს ფონური გამოსახულების საწყის პოზიციას

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ფონი</title> * <style> * div{ * width: 250px; * height: 200px; * margin: 10px; * color: white; * } * .colored{ * background-color: #ff0507; * } * .imaged{ * background-image: url(kanioni.png); * } * </style> * </head> * <body> * <div class="colored">">პირველი ბლოკი</div> * <div class="imaged">">მეორე ბლოკი</div> * </body> * </html> |
| --- |



გამოსახულების გამეორება

იმ შემთხვევაში, თუ ელემენტის ზომები მეტია ფონად გამოყენებული სურათის ზომებზე, ხდება სურათის გამეორება ისე, რომ ელემენტი მთლიანად შეივსოს ფონით. background-repeat თვისების მეშვეობით შესაძლებელია სურათის გამეორების მართვა. ის ღებულობს შემდეგ მნიშვნელობებს:

* repeat-x: გამეორება მხოლოდ ჰორიზონტალურად
* repeat-y: გამეორება მხოლოდ ვერტიკალურად
* repeat: გამეორება ორივე მიმართულებით (მნიშვნელობა მიუთითებლობისას)
* space: სურათი მეორდება ელემენტის სრულად შესავსებად, მაგრამ არ წარმოქმნის არასრულ ფრაგმენტებს
* round: ხდება სურათის მასშტაბირება ელემენტის სრულად შესავსებად
* no-repeat: სურათი არ მეორდება

მაგალითად:

| * background-image: url(nature.jpg); * background-repeat: round; |
| --- |

გამოსახულების ზომა

background-size თვისების გამოყენებით შესაძლებელია შევცვალოთ ფონად გამოყენებული გამოსახულების ზომა. ზომები (თანმიმდევრობით სიგანე, სიმაღლე) შეიძლება მივუთითოთ სტანდარტული ზომის ერთეულებით (px, em), პროცენტებით ან ერთ-ერთი წინასწარ განსაზღვრული მუდმივათი:

* contain: ხდება სურათის მასშტაბირება უდიდესი მხარის მიხედვით პროპორციების შენარჩუნებით
* cover:ხდება სურათის მასშტაბირება უმცირესი მხარის მიხედვით პროპორციების შენარჩუნებით
* auto: მნიშვნელობა მიუთითებლობისას. სურათის ზომები არ იცვლება

თუ გვინდა სურათის მასშტაბირება ისე, რომ მან სრულად შეავსოს ელემენტი გამეორებების გარეშე, სიგანეში და სიმაღლეში უნდა მივუთითოთ 100 %:

| * background-size:100% 100%; |
| --- |

ზომების მითითებისას ჯერ ეთითება სიგანე, მერე სიმაღლე:

| * background-size:150px 100px; |
| --- |

შეიძლება ერთი ზომა მივუთითოთ ზუსტად, მეორეში კი ჩავწეროთ auto და მეორე ზომას ბრაუზერი პროპორციულად გადათვლის:

| * background-size:150px auto; |
| --- |

სურათის მდებარეობა

background-position თვისება განსაზღვრავს ფონად გამოყენებული სურათის პოზიციას ელემენტის მიმართ. ის ღებულობს მნიშვნელობას ელემენტის მარცხენა ზედა კიდიდან ჰორიზონტალური და ვერტიკალური დაშორების სახით. მაგალითად:

| * background-position: 30px 15px; |
| --- |

ნიშნავს, რომ ფონის გამოსახულება მარცხენა კიდიდან დაშორებულია 30 პიქსელით, ხოლო ზემოთა კიდიდან - 15 პიქსელით.

გარდა ამისა, აღნიშნულმა თვისებამ შესაძლოა მიიროს შემდეგი მნიშვნელობები:

* top: სურათი სწორდება ელემენტის ზედა კიდის მიმართ
* left: სურათი სწორდება ელემენტის მარცხენა კიდის მიმართ
* right:სურათი სწორდება ელემენტის მარჯვენა კიდის მიმართ
* bottom: სურათი სწორდება ელემენტის ქვედა კიდის მიმართ
* center: სურათი სწორდება ელემენტის ცენტრში

მაგალითად:

| * background-position: top right; |
| --- |

აქ ფონის სურათი სწორდება ელემენტის ზედა და მარჯვენა კიდეებთან.

background-attachment

background-attachment თვისება განსაზღვრავს, თუ როგორ იქნება ფონის გამოსახულება მიმაგრებული ელემენტზე. ის ღებულობა შემდეგ მნიშვნელობებს:

* fixed: ფონი დაფიქსირებულია ელემენტის მიმართ და არ იცვლება ელემენტის შიგთავსის გადახვევისას (scroll)
* local: გადახვევისას (scroll) ელემენტის ფონი იცვლება (მიყვება გადახვევას)
* scroll: ფონი დაფიქსირებულია ელემენტის მიმართ და არ იცვლება ელემენტის შიგთავსის გადახვევისას (scroll). fixed-ისგან განსხვავებით, ამ შემთხვევაში ელემენტებს შესაძლებლობა აქვთ გამოიყენონ სხვადასხვა ფონი, fixed-ის შემთხვევაში ყველა ელემენტს საერთო ფონი აქვს.

მაგალითად:

| * div{ * width: 300px; * height: 250px; * overflow:scroll; /\* გადახვევის დამატება \*/ * border: 1px solid #ccc; * background-image: url(nature.png); * background-size: 512px 384px; * background-attachment: scroll; * background-repeat: no-repeat; * } |
| --- |

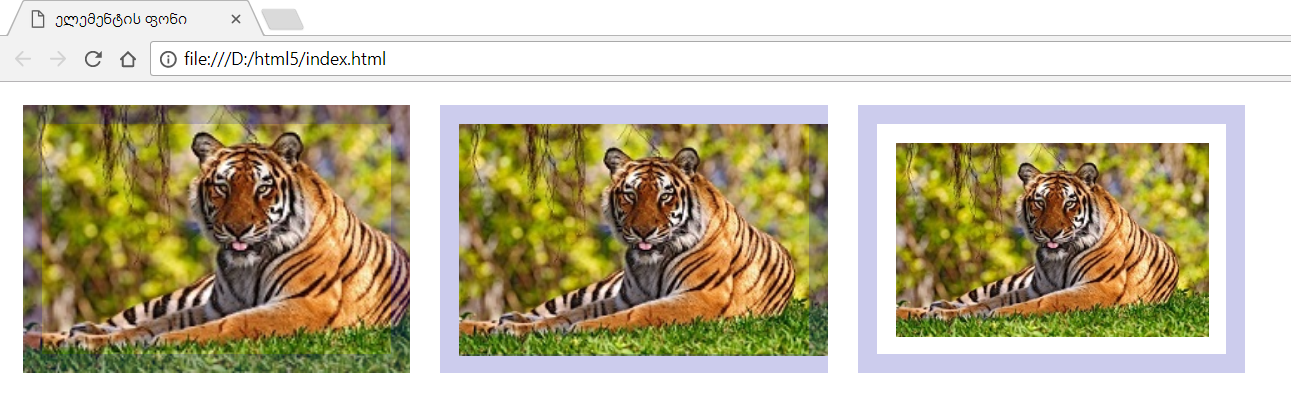
background-origin

თვისება background-origin განსაზღვრავს პოზიციას, საიდანაც დაიწყება ელემენტის ფონის გამოსახულება. ის ღებულობს შემდეგ მნიშვნელობებს:

* border-box: ფონი იწყება ელემენტის ჩარჩოდან
* padding-box: ფონი იწყება შიდა დაშორებიდან
* content-box: ფონი იწყება ელემენტის შიგთავსიდან

გამოვიყენოთ სამივე მნიშვნელობა:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ფონი</title> * <style> * div{ * width: 250px; * height: 155px; * margin: 10px; * display: inline-block; /\*მოვათავსოთ ბლოკები გვერდიგვერდ \*/ * padding:15px; * border: 15px solid rgba(0,0,165,0.2); * background-image: url(tiger.jpg); * background-size: cover; * background-repeat: no-repeat; * } * .borderBox {background-origin: border-box;} * .paddingBox {background-origin: padding-box;} * .contentBox {background-origin: content-box;} * </style> * </head> * <body> * <div class="borderBox"></div> * <div class="paddingBox"></div> * <div class="contentBox"></div> * </body> * </html> |
| --- |



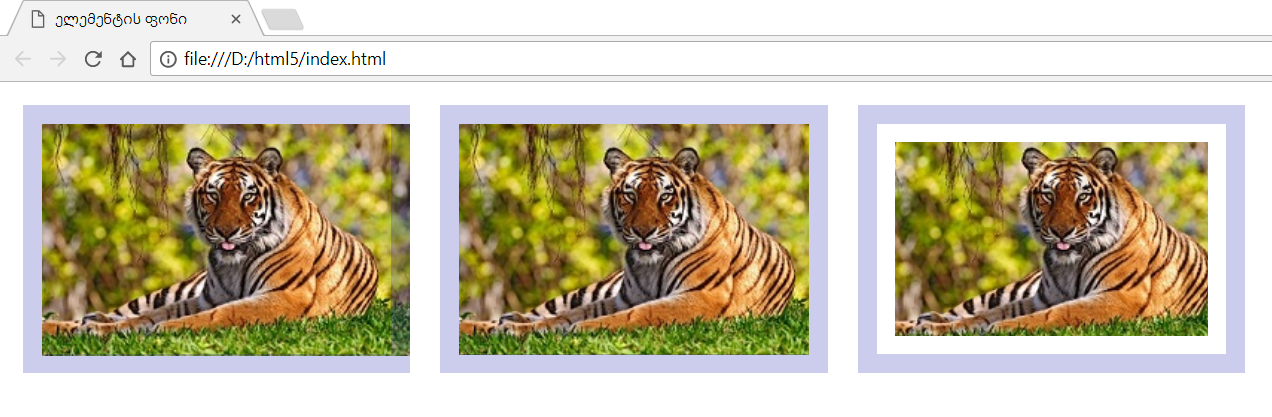
background-clip

background-clip თვისება განსაზღვრავს, სურატის რა ნაწილი გამოიყენება ფონად. ის ღებულობს იმავე მნიშვნელობებს, რასაც background-origin:

* border-box: სურათი ჩამოიჭრება ელემენტის ჩარჩოებთან
* padding-box: სურათი ჩამოიჭრება ელემენტის შიდა დაშორებასთან, ჩარჩოს ქვემოთ მოქცეული ნაწილი ჩამოიჭრება
* content-box: სურათი ჩამოიჭრება შიგთავსთან, შიდა დაშორებების ქვეშ მოქცეული ნაწილი ჩამოეჭრება

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ელემენტის ფონი</title> * <style> * div{ * width: 250px; * height: 155px; * margin: 10px; * display: inline-block; /\*მოვათავსოთ ბლოკები გვერდიგვერდ \*/ * padding:15px; * border: 15px solid rgba(0,0,165,0.2); * background-image: url(tiger.jpg); * background-size: cover; * background-repeat: no-repeat; * } * .borderBox {background-clip: border-box;} * .paddingBox {background-clip: padding-box;} * .contentBox {background-clip: content-box;} * </style> * </head> * <body> * <div class="borderBox"></div> * <div class="paddingBox"></div> * <div class="contentBox"></div> * </body> * </html> |
| --- |



ელემენტის ჩრდილი

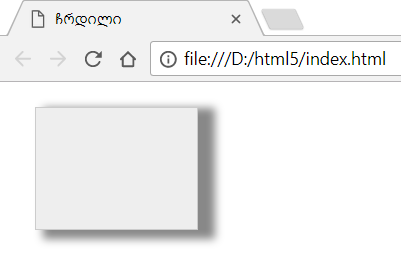
ელემენტისთვის ჩრდილის მისაცემად გამოიყენება თვისება box-shadow. ის იღებს ერთდროულად რამდენიმე მნიშვნელობას:

| * box-shadow: hoffset voffset blur spread color inset |
| --- |

* hoffset: ჩრდილის ჰორიზონტალური წანაცვლება ელემენტის მიმართ. დადებითი მნიშვნელობისას ჩრდილი წანაცვლებულია მარჯვნივ, უარყოფითისას - მარცხნივ
* voffset: ჩრდილის ვერტიკალურიწანაცვლება ელემენტის მიმართ. დადებითი მნიშვნელობისას ჩრდილი წანაცვლებულია ქვევით, უარყოფითისას - ზევით
* blur: არააუცილიებელი მნიშვნელობა. განსაზღვრავს ჩრდილის კუთხეების გაბნეულობის რადიუსს. რაც უფრო მეტია, მით უფრო ნაკლებად გამოხატულია კუთხეები. მიუთითებლობისას მისი მნიშვნელობა 0-ის ტოლია
* spread: არააუცილებელი მნიშვნელობა. განსაზღვრავს ჩრდილის მიმართულებას. დადებითი მნიშვნელობისას ჩრდილი მიმართულია ელემენტიდან გარეთ, ხოლო უარყოფითისას - ელემენტისკენ
* color: არააუცილებელი მნიშვნელობა. განსაზღვრავს ჩრდილის ფერს
* inset: არააუცილებელი მნიშვნელობა, რომელიდ ჩრდილს მიმართავს ელემენტის შიგნით

მაგალითად:

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>ჩრდილი</title> * <style> * div{ * width: 128px; * height: 96px; * margin: 20px; * border: 1px solid #ccc; * background-color: #eee; * box-shadow: 10px 4px 10px 3px #888; * } * </style> * </head> * <body> * <div></div> * </body> * </html> |
| --- |



ერთ ელემენტს შეიძლება განესაზღვროს რამდენიმე ჩრდილი, ისინი გამოიყოფა მძიმეებით:

| * box-shadow: 5px 3px 8px 3px #faa, 10px 4px 10px 3px #888 inset; |
| --- |

ელემენტის კონტურები

ელემენტის კონტურების გამოყენება ჩარჩოების მსგავსია, მაგრამ მათ სხვადასხვა დანიშნულება აქვთ. კონტურები გამოიყენება რაღაც ელემენტების სხვებისაგან გამოსაყოფად, ყურადღების მისაქცევად. კონტურები ელემენტის გარეთაა ჩარჩოს შემდეგ.

კონტურებისთვის CSS3-ში გამოიყენება თვისება outline, რომელიც შემდეგი თვისებების შემოკლებაა:

* outline-color: კონტურის ფერი
* outline-offset: კონტურის წანაცვლება
* outline-style: კონტურის სტილი, იღებს იმავე მნიშვნელობებს, რასაც border-style:
  + none: კონტურის გარეშე
  + solid: კონტური ჩვეულებრივი ხაზით
  + dashed: წყვეტილი ხაზი
  + dotted: წერტილოვანი ხაზი
  + double: ორმაგი ხაზი
* outline-width: კონტურის სისქე

| * <!DOCTYPE html> * <html> * <head> * <meta charset="utf-8"> * <title>კონტურები</title> * <style> * div{ * width: 128px; * height: 96px; * margin: 20px; * border: 1px solid #ccc; * background-color: #eee; * outline-color: red; * outline-style: dashed; * outline-width: 2px; * } * </style> * </head> * <body> * <div></div> * </body> * </html> |
| --- |



outline თვისების გამოყენებით შეიძლება ეს ყველაფერი შემოკლებულად ჩავწეროთ:

| * outline: red dashed 2px; |
| --- |