მაგდა ცინცაძე, ტატა წილოსანი.

შესავალი ვებ დიზაინში

ლექციათა კურსი.

თბილისი
2009

ნაშრომი წარმოადგენს დამხმარე სახელმძღვანელოს საგანში „ვებ დიზაინი“. მასში მოცემულია ვებ-ის აღწერა, განვითარების ისტორია, აღწერილია HTML (Hyper Text Markup Language) ჰიპერტექსტების მარკირების ენა, CSS (Cascading Style Sheets) ენის სინტაქსი და გამოყენების საშუალებები აგრეთვე CSS3 და Java script-ის საფუძვლები.

მაგდა ცინცაძე, ტატა წილოსანი

1 თავი. ვების და ინტერნეტის ისტორია, სტანდარტების ჩამოყალიბება

შესავალი:

მოკლე ანოტაცია:

ინტერნეტის შექმნა

რა გასაკვირიც არ უნდა იყოს ეს საკითხი უკავშირდება 1957 წლის 4 ოქტომბერს, რომელმაც შეცვალა მსოფლიო. ამ დღეს მოხდა პირველი თანამგზავრის წარმატებით გაშვება, რამაც გამოიწვია ამერიკის მიერ პერსპექტიულ მეცნიერულ კვლევათა სამმართველოს ARPA–ს შექმნა. ომპიუტერული ქსელის შექმნის გეგმა სახელწოდებით "ARPANET" წარმოდგენილი იქნა 1967 წლის ოქტომბერს, ხოლო 1969 წლის დეკემბერში კი 4 კომპიუტერისაგან შემდგარი პირველი ქსელი ამუშავდა. არგა ხანს პრობლემად რჩებოდა სხვადასხვა ლოკალური ქსელის ერთმანეთთან დაკავშირება, მხოლოდ 1977 წელს მოხერხდა სამი ქსელის ერთმანეთთან კომუნიკაციის პირველი წარმატებული დემონსტრირება. ხოლო 1982 წელს ARPANET ქსელი, ამერიკის საზღვრებს იქეთ გადაიყვანეს ახალი პროტოკოლის (TCP/IP-ის)

გამოყენებაზე. შეიქმნა ინტერნეტი იმ სახით რომელსაც ჩვენ მას დღეს ვიცნობთ.ბულ იქნა.

World Wide Web შექმნა

Gopher (http://en.wikipedia.org/wiki/Gopher_%28protocol%29) წარმოადგენდა ინფორმაციის ამოღების ძირითად საშუალებას 90-იანი წლების დასაწყისისათვის. ის შეიქმნა მინესოტის უნივერსიტეტის ბაზაზე. 1993 წელს უნივერსიტეტმა განაცხადა რომ მოითხოვდა ლიცენზიის გადასახადს თავისი ეტალონური Gopher სერვერის რეალიზაციისათვის. ამის გამო ბევრმა ორგანიზაციამ დაიწყო Gopher –ის ალტერნატივეს ძიება. ატომური კვლევების ევროპულ საბჭოს შვეიცარიაში (CERN), გააჩნდა ასეთი ალტერნატივა. ტიმ ბერნერს ლი მუშაობდა ინფორმაციის მართვის სისტემაზე, რომელშიც ტექსტს შეეძლო მოეცვა კავშირები და მიმართვები სხვა დოკუმენტზე, შესაბამისად მომხმარებელს საშუალებას აძლევდა სწრაფად გადაადგილებულიყო ერთი დოკუმენტიდან მეორეზე. მან შექმნა სერვერი ამგვარი დოკუმენტების პუბლიკაციისათვის და აგრეთვე პროგრამა მათი წაკითხვისათვის, რომელსაც უწოდა "WorldWideWeb". ეს პროგრამული უზრუნველყოფა პირველად გამოშვებული იქნა 1991 წელს. 1993 წელს CERN-მა გამოუშვა WorldWideWeb-ის საწყისი კოდი საერთო სარგებლობისათვის, ანუ ყველას შეეძლო სრულიად უფასოდ გამოეყენებინათ ის. მოგვიანებით NCSA-მ გამოუშვა პროგრამა სახელწოდებით Mosaic , რომელიც წარმოადგენდა ვებ ბრაუზერისა და Gopher კლიენტის კომბინაციას. საწყის ეტაპზე ის ხელმისაწვდომი იყო მხოლოდ Unix მანქანებზე, თუმცა 1993 წლის დეკმბერში გამოვიდა მოზაიკის ახალი

მაგდა ცინცაძე, ტატა წილოსანი

ვერსია Apple Macintosh და Microsoft Windows-სათვის, რომელიც ძალზედ სწრაფად გახდა პოპულარული.

"ბრაუზერების ომი"

Web-ის პოპულარულობამ გამოიწვია შესაბამისად კომერციული ინტერესის გამძაფრება. მარკ ადრისენმა დატოვა NCSA და ჯიმ კლარკთან ერთად ჩამოაყალიბა კომპანია Mosaic Communications, შემდგომში Netscape Communications Corporation-ად წოდებული და დაიწყო მუშაობა პროგრამაზე რომელსაც ეწოდა Netscape Navigator და გამოვიდა 1994 წელს, აქედან იწყება ბრაუზერების ისტორია. კომპანიამ Spyglass Inc. (NCSA-ს კომეციული განყოფილებამ) მოახდინა Mosaic ტექნოლოგიის ლიცენზირება Microsoft–სთვის ასე შეიქმნა Internet Explorer version1 1995 წელს.

მას მოყვა სწრაფი განვითარება, როდესაც თითოეული კოპანია Netscape და Microsoft ცდილობდნენ მიეღწიათ კონკურენტული უპირატესობისათვის, ამისათვის ვებ-დეველოპერების მოზიდვის მიზნით , ახალ ვერსიებში ახდენდნენ ახალ-ახალი თვისებების რეალიზებას (აღსანიშნავია რომ ეს თვისებები იყო შეუთავსებელი სხვადასხვა ბრაუზერისათვის, რაც დიდ პრობლემებს ქმნიდა საბოლოო ჯამში). ამ პროცესს მოგვიანებით ეწოდა "ბრაუზერების ომი". ამ აურზაურში მყარად ინარჩუნებდა არსებობის უფლებას ე.წ. "პატარა ოპერა" (Opera), რომელიც თავის მხრივ ცდილობდა შესაძლებლობისა და მიხედვით დაენერგა ვებ სტანდარტები და თავის შემდგომ ვერსიებში მათი მხარდაჭერის უზრუნველყოფა მოეხდინა.

ვებ სტანდარტების ჩამოყალბება

ბრაუზერების ომის დროს კომპანიები Microsoft და Netscape ორიენტირებული იყვნენ ახალი თვისებების რეალიზებებზე და არა იმ პრობლემების გამოსწორებაზე რაც უკვე არსებულ თვისებებს გააჩნდა, შესაბამისად ამატებდნენ და ამატებდნენსა კუთარ თვისებებს, რომლებოც პირდაპირ კონკურირებდნენ სხვა ბრაუზერის უკვე არსებულ თვისებებთან და რეალიზებული იყვნენ შეუთავსებლობის პრინციპით. შესაბასად ვებ დეველოპერები იძულებული იყვნენ შეექმნათ ორი სხვადასხვა გვერდი (შინაარსობრივად ერთიდაიგივე) ორი ძორითადი ბრაუზერისათვის, ანუ რეალობაში უწევდათ ორმაგი შრომა, რაღა თქმა უნდა რომ ეს მა თარ სიამოვნებდათ.

W3C ფორმირება

1994 წელს, ტიმ ბერნერს ლი-მ შექმნა World Wide Web Consortium (*W3C*) მასაჩუსეცის უნივერსიტეტში CERN–ისა და DARPA-ს ხელშეწყობით. ისინი თავის ფუნქციად მიიჩნევდნენ ვებ პროტოკოლებისა და ტექნოლოგიების სტანდარტიზაციას, რათა ინფომაცია ხელმისაწვდომი ყოფილიყო რაც შეიძლება მეტი მომხმარებლისათვის. რამოდენიმე წლის განმავლობაში კონსორციუმი აქვეყნებდა სხვადასხვა სპეციფიკაციებს (რეკომენდაციებს), მათ შორის HTML 4.0, გამოსახულების ფორმატი PNG, კასკადური ტიპის ცხრილები (CSS), ვერსიები 1 და 2.

აღსანიშნავია რომ *W3C* არ აიძულებს თავისი რეკომენდაციების დაცვას, დაცვა მოითხოვება მხოლოდ იმ შემთხვევაში თუ დეველოპერს სურს თავისი პროდუქტი მონიშნოს როგორც *W3C*-ის შესაბამისი, ამას იმ დროისათვის გაყდვების თვალსაზრისით დიდი მნიშვნელობა არ ქონდა რადგანაც უმრავლესობას საერთოდ არ ქონდა ამ სტანდარტეზე წარმოდგენა და დიდად სულაც არ ანაღვლებდათ რას ექნებოდა ნიშანი *W3C*, შესაბამისად ბრაუზერების ომი იგივე სიმძლავრით გრძელდებოდა.

ვებ სატნდარტების პროექტი

1998 წელს ბრაუზერების ბაზარზე დომინირებდნენ Internet Explorer 4 და Netscape Navigator 4, ამ დროს მოხდა Internet Explorer 5 ბეტა ვერსიის გამოშვება რომელშიც რეალიზებული და დაპატენტებული იყო ახალი დინამიური HTML, ეს კი იმას ნიშნავდა რომ ვებ დეველოპერს უნდა ცოდნოდა JavaScript-ის 5 სხვადასხვა ჩაწერის შესაძლებლობა.

ეს კი უკვე მართლაც მეტისმეტი იყო, ამიტომ პროფესიონალი დეველოპერები გაერთიანდნენ და ჩამოაყალიბეს "Web Standards Project" (*WaSP*). მათი იდეა მდგომარეობდა იმაში რომ *W3C*-სთვის ეწოდებინათ სტანდარტები და არა რეკომენდაციები, შესაბამისად დაერწმუნებინათ კომპანიები Microsoft და Netscape ამ სტანდარტების მხარდაჭერის აუცილებლობასა და მომგებიანობაში. საწყის მეთოდათ ამ მოწოდების გავრცელებისა გამოიყენებოდა რეკლამები (ე.წ. ” roadblock” - ეს სახეობა რეკლამის არის ისეთი რომელიც გადის ყველა არხზე ერთდროულად, შესაბამისად სხვა არხზე გადართვის დროსაც უყურებ

ბაგდა ცინცაძე, ტატა წილოსანი

იგივეს) . *WaSP* ჯგუფმა გამოაქვეყნა სტატია რამოდენიმე გვერდზე ერთდროულად, მათ შორის პოპულარულ builder.com, Wired online და სხვა. მას გარდა დაიწყეს დაცინვა იმ კომპანიებისა რომლებიც თავიდან შეუერთდნენ *W3C* სტანდარტებს , შემდგომ კი დაიცყეს ისევ ახალი საშუალებების შექმნისკენ სწრაფვა.

დაბოლოს, აღსანიშნავია რომ WaSP ჯგუფი არა მხოლოდ აკრიტიკებდა, არამედ ეხმარებოდა კიდევაც. ჟგუფის 7-მა წევრმა ჩამოაყალიბა ჯგუფი " CSS Samurai", რომელიც ახდენდა CSS -ის მხარდაჭერის, ათი ძირითადი პრობლემის იდენტიფიცირებას Opera და Internet Explorer –ში (აქვე შევნიშნოთ რომ ოპერამ გამოასწორა თავისი შეცდომები, მაიკროსოფტმა კი არა).

ვებ სატნდარტების განვითარება

2000 წელს მაიკროსოფტმა გამოუშვა Internet Explorer 5 Macintosh Edition , ეს იყო ძალზედ მნიშვნელოვანი ეტაპი რადგანაც მას დაახლოებით ისეთივე მისაღები დონე ქონდა ვებ სტანდარტების მხარდაჭერისა როგორიც ოპერას CSS და HTML-ისა. ამან ხელი შეუწყო საერთო დადებითი რეაქციის გამოწვევას პროგრამისტებსა და ვებ დეველოპერებში, მათ პირველად იგრძნეს მოხერხებულობა სტანდარტების მიხედვით დაპროექტებული ვებ გვერდებისგან.

WaSP ჯგუფმა დაარწმუნა *Netscape* გადაედო ახალი მე-5 ვერსიის *Netscape navigator* გამოშვება სანამ ის არ იკნებოდა უფრო შეთანხმებული ვებ სტანდარტებთან,

ეს საფუძველი გახდა იმისა რომ ახლად გამოშვებული *Firefox* ძალზედ პოპულარულ ბრაუზერად იქცა. WaSP-მა შექმნა აგრეთვე *"Dreamweaver Task Force" , რათა სტიმული მიეცა მაკრომედიასთვის გადაეკეთებინა მისი პოპულარული ვებ გვერდების შესაქმნელი ინსტრუმენტი სტანდარტების შესაბამისი ვებ რესურსების შესაქმნელად.*

იყო მოსაზრება რომ დაახლოებით წელიწადში, უფრო მეტიც, 6 თვეში ყველა საიტი იქნებოდა პროექტირებული ვებ სტანდარტების დაცვით, თუმცა ეს მოსაზრება ძალზედ გადაჭარბებულია, დღესდღეისობითაც კი არაა ეს ასე, ანუ ყველა საიტი არ აკმაკოფილებს ამ სტანდარტებს, მაგრამ მიუხედავად ამისა ბევრი დაეთანხმა ამ მოთხოვნებს, შესაბამისად ძველმა ბრაუზერებმა დაკარგეს ბაზარი, მრავალი საიტი კი გადაკეთდა, მაგალიდათ ჟურნალი Wired 2000 წელს, პოპულარული გვერდი A List Apart - 2001 წელს, ESPN - 2003-ში. ამ დროიდან მოყოლებული პროფესიონალ დეველოპერების საზოგადოებაში სტანდარტების დაცვა გახდა სავალდებულო, ამდენად საკმაოდ აქტუალური და მნიშვნელოვანია ფეხი ავუბათ და შევისწავლოთ ვებ გვერდების შექმნა ისე როგორც ეს მსოფლიოს აღიარებული პროფესიონალების მიერაა რეკომენდირებული, ანუ დავიცვათ ვებ სტანდარტები.

დამატებითი ლიტერატურა

- http://en.wikipedia.org/wiki/History_of_the_Internet, http://ru.wikipedia.org/wiki/Интернет
- http://en.wikipedia.org/wiki/History_of_the_World_Wide_Web, http://ru.wikipedia.org/wiki/WWW
- http://www.w3.org/Consortium/history, http://ru.wikipedia.org/wiki/W3C)

- http://webstandards.org/
  http://www.webstandards.org/about/history/
- http://www.alistapart.com/
- http://www.csszengarden.com/

როგორ მუშაობს ინტერნეტი
როგორ ურთიერტობენ კომპიუტერები ინტერნეტში?

კომპიუტერებისდა საბედნიეროდ ყველაფერი საკმაოდ მარტივად ხდება, როდესაც საქმე გვაქვს World Wide *Web*-თან, ყველა საუბრობს ერთ ენაზე HTML . HTML არის ასე ვთქვათ საერთო დიალექტი (სპეციფიკაცია), რომელც საშუალებას აძლევს მაგალითად Windows – იან მანქანას ჰარმონიაში იყოს Linux მანქანასთან. ვებ ბრაუზერის ანუ სპეციალური პროგრამის მეშვეობით, ხდება HTML კოდის ინტერპრეტაცია და მისი გარდაქმნა ადამიანისათვის ხელმისაწვდომ ფორმად – ვებ გვერდად. ნებისმიერი ტიპის კომპუტერზე HTML – ზე დაწერილიგვერდის წაკიტხვა შესაძლებელია ნებისმიერ ადგილას სხვადასხვა მოწყობილობების საშუალებით,მათ შორის ტელეფონებით, კომუნიკატორებითა და სხვა.

მიუხედავად იმისა რომ ისინი საუბრობენ ერთ ენაზე სხვადასხვა მოწყობილობების –თან წვდომა უნდაეყრდნობოდეს გარკვეულ წესებს,რომ შეძლონ ერთმანეთთან ურთიერთობა. *HTTP* განსაზღვრავს ამ ძირითად წესებს ინტერნეტისათვის. *HTTP* –ის თანახმად კლიენტის მანქანამ (მაგალითად კომპიუტერმა) იცის რომ მან უნდა მოახდინოს სერვერზე ვებ გვერდის მოთხოვნის ინიცირება. სერვერი წარმოადგენს კომპიუტერს რომელზეც ეშვება/სრულდება ვებ

სერვერის პროგრამა, რომელიც იღებს რა მოთხოვნას, პოულობს შესაბამის გვერდს და უგზავნის მას კლიენტს, რომელზეც ის გამოდის ვებ ბრაუზერის ფანჯარაში.

ციკლი მოთხოვნა/პასუხი

უფრო ვრცლად განვიხილოთ *HTTP* –ის ციკლი მოთხოვნა/პასუხი:

1. ყველა მოთხოვნა/პასუხი იწყება რესურსის უნიფიცირებული მიმთითებლის (URL, Universal Resource Locator) შეყვანით ვებ ბრაუზერის სამისამართე ზოლში, მაგალითად ასე: http://dev.opera.com.

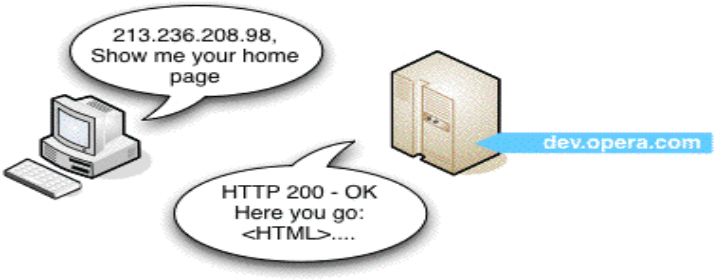
   აქვე შევნიშნოთ რომ სინამდვილეში ვებ ბრაუზერი სულაც არ იყენებს URL-ს ვებ სერვერზე გვერდის მოთხოვნისათვის, ამისათვის გამოიყენება ინტერნეტ პროტოკოლი ანუ იგივე IP მისამართი. მაგალითად http://dev.opera.com–ის IP მისამართია: 213.236.208.98.

2. ცადეთ ბრაუზერის სამისამართე ზოლში აკრიფოთ http://www.apple.com და დააჭირეთ ღილაკს “Enter”, შემდეგ კი აკრიფეთ http://17.149.160.10/ და ისევ დააჭირეთ ღილაკს “Enter”, ნახავთ რომ ორივე შემთხვევაში მოხვდებით ერთიდაიგივე ადგილას. გამოდის რომ http://www.apple.com წარმოადგენს ფაქტიურ ალიასს (სინონიმს, ფსევდონიმს) http://17.149.160.10/ -სა, შესაბამისად ისმის კითხვა რატომ, რაში გვჭირდება ეს? , საქმე იმაში მდგომარეობს რომ ადამიანისათვის გაცილებით

მოსახერხებელია სიტყვების დამახსოვრება ვიდრე ოთხ ნაწილიანი რიცხვების მიმდევრობისა. სისტემა რომელიც ამ სამუშაოს ასრულებს არის დომენების სახელების სისტემა ანუ DNS, რომელიც წარმოადგენს ფაქტიურად ამომწურავ ავტომატურ კატალოგს ყველა მანქანისა რომლებიც ჩართული არიან ინტერნეტში. ღოცა თქვენ ათავსებთ http://dev.opera.com –ს სამისამართე ზოლში და აჭერთ ღილაკს “Enter”, ეს მისამართი ეგზავნება დომენების სისტემას, რომელიც ცდილობს მის დაკავშირებას შესაბამის IP მისამართთან . არსებობს უამრავი მანქანა ჩართული ინტერნეტში და ყველა DNS სერვერს სულაც არ გააჩნია სრული სია ქსელში არსებული ყველა მანქანისა, ამიტომაც არსებობს სისტემა, რომელსაც შეუძლია გადაუგზავნოს მოთხოვნა შესაბამის სერვერს დასამუშავებლად.

DNS სისტემა პოულობს ვებ გვერდს http://www.apple.com, განსაზგვრავს რომ ის არის მისამსრთზე `17.149.160.10` და უგზავნის ამ IP მისამართს ვებ ბრაუზეს..

თქვენი მანქანა უგზავნის მოთხოვნას მანქანას(სერვერს) მითითებულ IP მისამართზე და ელოდება პასუხს. თუ ყველაფერი წარმატებითაა მანქანა სერვერი მოკლე შეტყობინებას უგზავნის კლიენტ მანქანას, რითაც ამბობს რომ ყველაფერი ნორმალურადაა, რასაც მოყვება თავად ვებ გვერდიც:



სურათზე ნაჩვენებია შემტხვევა როცა ყველაფერი ნორმალურად წარიმართებადა სერვერი აბრუნებს შესაბამის ვებ გვერდს.

თუ რაიმე არასწორად მოხდა, მაგალითად მისამართი იქნა არასწორად შეყვანილი, მაშინ მიიღებთ შეტყობინებას შეცდომის თაობაზე: Error 404 "page not found" ("გვერდი ვერ მოიძებნა")

3. ცადეთ შეიყვანოთ მისამართი : http://dev.opera.com/joniscool.html - ეს გვერდი არ არსებობს, ამიტომაც მიიღებთ შეცდომა 404-ს.

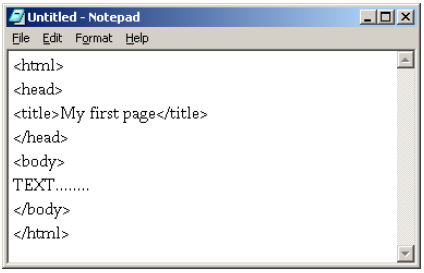
თავი 2. WEB გვერდების პროექტირება

ინტერნეტში არსებული ინფორმაციის დათვალიერება შესაძლებელია ინტერნეტ-აგენტების, მათ შორის ბრაუზერების საშუალებით.ეს ინფორმაცია როგორც წესი ინახება რომელიმე Web სერვერზე საიტის სახით. Web საიტი შედგება Web გვერდებისაგან, რომელთა შექმნა და ერთმანეთთან დაკავშირება ხდება html (Hyper Text Markup Language) საშუალებით. html არ წარმოადგენს პროგრამირების

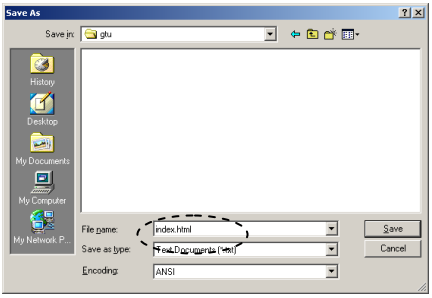
ენას, მისი საშუალებით ხდება მხოლოდ Web გვერდების შექმნა და რედაქტირება.

WEB გვერდის შესაქმნის მარტივი საშუალება

Web გვერდის შექმნა შესაძლებელია ნებისმიერ ტექსტური რედაქტორის მეშვეობით. ამისათვის ვიქცევით ასე: ვხსნით ტექსტურ რედაქტორს მაგალიდათ **NotePad** (Start > Run > ვკრეფავთ **NotePad)** და **NotePad** –ის ფანჯარაში ვკრიფავთ html კოდს:



შემდგომ შექმნილი დოკუმენტის დამახსოვრებისათვის Save as type-ში ვუთითებთ All files და ვანიჭებთ ფაილს html ან htm გაფართოებას. ე.ი. დამახსოვრების დროს File name-ში ვკრიფავთ სახელი.html მაგ: **index.html**;



Web გვერდის დათვალიერება ხდება ინტერნეტ-აგენტების (კომპიუტერში –ბრაუზერის, ტელეფონში-აგენტის) საშუალებით, რედაქტირებისათვის კი ვბრუნდებით ტექსტურ რედაქტორში, ვახდენთ შესაბამისი დოკუმენტის რედაქტირებას და ვიმახსოვრებთ მას. (ბრაუზერიდან html დოკუმენტის გახსნა ხდება კონტექსტური მენიუდან View Source ბრძანებით.)

ინტერნეტ-ბრაუზერში დოკუმენტის განახლება ხდება ინსტრუმენტების ველიდან  Refresh ბრძანებით ან კლავიატურიდან F5.

ტეგები

ნებისმიერი html/htm დოკუმენტი შედგება **ტეგებისაგან**.

ტეგი ეს არის ბრძანება რომელიც მოთავსებულია < და > სიმბოლოებს შორის (მაგ. <h1>) და განსაზღვრავს მის შემდეგ წარმოდგენილი ტექსტის ან გრაფიკული გამოსახულების მახასიათებლებს/თვისებებს. მაგალითად: მდებარეობას, ზომას, ფერს და ა. შ.

გამოიყენება **<ტეგის_სახელი>** - გამხსნელი და **</ტეგის_სახელი>** - დამხურავი ტეგი, რომელთა საშუალებითაც ხდება ამა თუ იმ ბრძანების მოქმედების არის განსაზღვრა. (არსებობს გამონაკლისი ტეგები რომლებისთვისაც არ გამოიყენება დამხურავი ტეგი,

სხვასასხვა სტანდარტში სხვადასხვანაირადაა, მაგალითად img ტეგი არ იხრება HTML-ში, XHTML-ში კი თვითდახურვადი ტეგია)

ძირითადად ტეგი შედგება ორი ნაწილისაგან:

ტეგის სახელი - განსაზღვრავა რომელ ობიექტზე ხდება მოქმედება;
ატრიბუტი – მიუთითებს რა მოქმედება უნდა შესრულდეს ამ ობიექტზე.

მაგალითად:

<ტეგის_სახელი ატრიბუტი1="მნიშვნელობა" ატრიბუტი2="მნიშვნელობა">
...
მოქედების ველი
....
</ტეგის_სახელი>

მაგ.: ****
Tbilisis Saxelwifo Universiteti
****

ძირითადი/ბაზისური ტეგები

<HTML> ... </HTML> - არის Web დოკუმენტის პირველი და ბოლო ტეგი, რომელიც მიუთითებს რომ აღნიშნული ფაილი არის web გვერდი და არა უბარლო ტექსტი.

<HEAD> ... </HEAD> - სათაურის ბლოკი, სადაც იწერება ზოგადი ინფორმაცია ვებ გვერდის შესახებ.

<TITLE> ... </TITLE> - ტეგებს შორის იწერება დოკუმენტის სახელი, რომელიც შემდგომ გამოისახება ბრაუზერის სათაურის ველში.

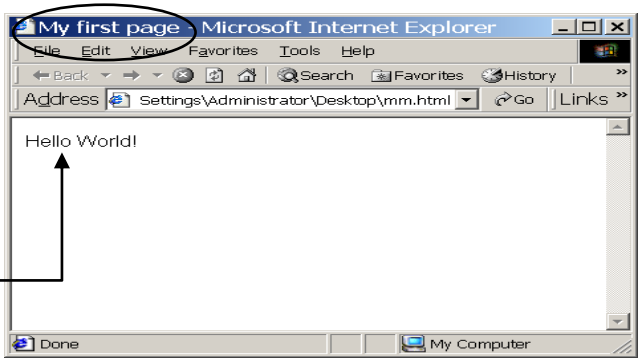
<BODY> ... </BODY> - ტეგი ხსნის დოკუმენტის ტანს, სადაც აღიწერება მთლიანად დოკუმენტი (web გვერდის შემცველობა).

html დოკუმენტის ზოგადი სტრუქტურა ასეთია:

<HTML>
<HEAD>
<TITLE> დოკუმენტის სახელი **</TITLE>**
</HEAD>
<BODY>
... დოკუმენტის შემცველობა **...**
</BODY>
</HTML>

მაგ:

```
<html>
<head>
<title>
My first page
</title>
</head>
  <body>
    Hello World!
  </body>
</html>
```



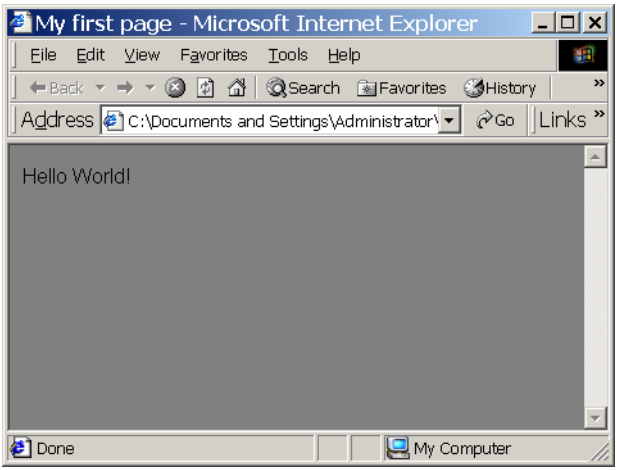
თუ BODY ტეგში არ მიუთითეთ Web გვერდის ფონის ფერი, ავტომატურად ირჩევს თეთრ ფონს. შესაძლებელია ფონის შეცვლა Bgcolor ატრიბუტით.

არსებობს ფერის მითითების ორი გზა:

1. ფერის ინგლისურენოვანი სახელის მითითებით
   <body **bgcolor=gray**>
2. ფერის ექვსციფრიანი კოდის მითითებით
   <body **bgcolor=#808080**>

მაგ:

```
<html>
 <title>My first page </title>
  <body bgcolor=gray>
    Hello World!
  </body>
</html>
```



ტეგი META – მეტა ტეგები

მეტა ტეგები არის ინფორმაცია ბროუზერისთვის. ეს ტეგები თავსდება <HEAD></HEAD> ტეგს შორის. მეტა ტეგის უმრავლესობა არ არის სავალდებულო.

კოდირების ტეგი (აუცილებელი ტეგია)

<meta http-equiv="content-type" content="text/html; charset= კოდირების სისტემის დასახელება">.

მაგალითი

<meta http-equiv="content-type" content="text/html; charset=Windows-1251">

ძირითადი ფაილის წინ დამატებითი ფაილის ჩვენება

<Meta name="Refresh" Content="წამების რაოდენობა; Url=ფაილის სახელი Htm გაფართოებით">.

მაგალითი

<Meta name="Refresh" Content="10; Url=INDEX.HTM">

დოკუმენტი INDEX.HTM ჩაიტვირთება 10 წამის შემდეგ

ინფორმაცია ავტორის შესახებ

<Meta name="Author" Content="avtori">

მაგალითი

<Meta name="Author" Content=" მაგდა ცინცაძე">

დოკუმენტის აღწერა

აქ დაწერილი ტექსტი გამოჩნდება საძებნი სისტემის მიერ. ტექსტის სიგრძე მაქსიმუმ 200 სიმბოლოა.

<Meta name="Description" Content="ა ღწ ე რა ">

მაგალითი

<Meta name="Description" Content="ინტერნეტ-ტექნოლოგიები (ლექციების კურსი)">

საკვანძო სიტყვები ინფორმაციულ-საძიებო სისტემისთვის.

სიის სიგრძე მაქსიმუმ 800 სიმბოლოა. სიტყვები ერთმანეთისგან გამოიყოფა მძიმით ან ინტერვალის ნიშნით
<Meta name="Keywords" Content="საკვანძო სიტყვები">

მაგალითი

<Meta name="Keywords"
Content="interneti,teqnologia,leqciebi">

E-Mail-ის მისამართი

<Meta name="Reply-to" Content="E-Mail-ი ს მ ი ს ა მ ა რთი ">

მაგალითი

<Meta name="Reply-to" Content="VINME@Rambler.ru">

საიტის შექმნის თარიღი

<Meta Name="Date" Content="თვ ე , დღე , წ ე ლი , დრო">
მაგალითი

<Meta Name="Data" Content="May 15 2007 14:35 Am">

ტეგი LINK ხორციელდება დოკუმენტების დაკავშირება. მისი ჩაწერის სინტაქსი შემდეგია:
<LINK [REL=კავშირის ტიპი] [HREF=URL]
[TYPE=დაკავშირების შინაარსი]>

აბზაცის ფორმატირება

<P> - ტეგი ხსნის ახალ აბზაცს, სადაც შეიძლება მოვათავსოთ როგორც ტექსტი, ასევე გრაფიკული გამოსახულება.

ALIGN ატრიბუტის საშუალებით ხდება მდებარეობის განსაზღვრა. ამ ატრიბუტს შეიძლება მიენიჭოს შემდეგი მნიშვნელობები:

LEFT-მარცხნივ**, CENTER**-ცენტრი, **RIGHT**-მარჯვნივ.

მაგ.: **<P ALIGN=CENTER>**

შენიშვნა: ALIGN ატრიბუტის გაჩუმებით (default) მნიშვნელობაა LEFT-მარცხნივ, ე.ი. თუ გვინდა აბზაცის მარცხნივ გასწორება, მაშინ შეგვიძლია პირდაპირ დავწერთ <P>.

**
** - ტეგის საშუალებით ხდება ახალ სტრიქონზე გადასვლა. ამ დროს უცვლელი რჩება წინა აბზაცში მითითებული მდებარეობა. ეს ტეგი ე.წ. თვითდახურვადი ტეგია

მაგ: This is
a para
graph with line breaks

<BLOCKQUOTE> - ტეგის საშუალებით ხდება ტექსტის ტაბულაცია.

მაგ:
<html>
<body>

Here comes a long quotation:
<blockquote>
This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation.
</blockquote>
Notice that a browser inserts white space before and after a blockquote element. It also insert margins for the blockquote element.

</body>
</html>

შედეგი იქნება ასეთი:

Here comes a long quotation:

> This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation.

Notice that a browser inserts white space before and after a blockquote element. It also insert margins for the blockquote element.

ტექსტის ფორმატირება

FONT – ტეგის საშუალებით ხდება შრიფტის აღწერა. ამ ტეგს გააჩნია შემდეგი ატრიბუტები:
FACE – შრიფტის დასახელება,
SIZE – ზომა,
COLOR – ფერი.

მაგ.: This is some text!</p>

შრიფტის გაჩუმებით (default) მნიშვნელობაა - Times New Roman. ე.ი. იმ შემთხვევაში თუ არ არის მითითებული შრიფტის დასახელება ავტომატურად იწერება ინგლისურად.

შესაძლებელია ერთდროულად მიუთითოთ რამოდენიმე შრიფტი. <font face="I შრიფტი, II შრიფტი ... "> იმ შემთხვევაში თუ მომხმარებლის კომპიუტერზე არ არის დაყენებული I შრიფტი, web გვერდის შესაბამის ნაწილის გამოიტანა მოხდება შემდეგი შრიფტით და. ა.შ. მაგ.: **<font face="AcadNusx, DumbaNusx">**

შრიფტის ზომის გაჩუმებით (default) მნიშვნელობაა – 3.

არსებობს ზომის მითითების ორი ხერხი: აბსოლიტური <font size=4> და შედარებითი <font size=+1>.

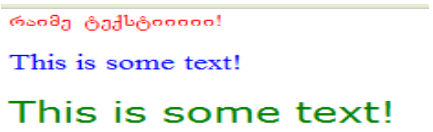
აბსოლიტური მნიშვნელობა	1	2	3	4	5	6	7
შედარებითი მნიშვნელობა	-2	-1	-	+1	+2	+3	+4
	6pt	8pt	10pt	12pt	14pt	18pt	27pt

შრიფტის გაჩუმებით (default) მნიშვნელობაა – შავი.

მაგ:

```
<html>
<body>

<p><font size="2" color="red" face="AcadNusx"> raime teqstiiii! </font></p>

<p><font size="4" color="blue">This is some text!</font></p>

<p><font face="verdana" color="green" size="5" >This is some text!</font></p>

</body>
```

</html>
შედეგად გვექნება:



შესაძლებელია HTML დოკუმენტის საბაზო პარამეტრების შეცვლა <BaseFont> ტეგის საშუალებით, რომელიც უნდა მოთავსდეს <Body> ტეგის შემდეგ.

მაგ:

```
<html>
<title> TSU </title>
<body>
<BaseFont face=AcadNusx size=4>
<p> Tbilisis saxelWifo universiteti
 <br/><Font face=arial> Tbilisi State University</font></p>
</body>
</html>
```

განხილულ მაგალითში ვინაიდან საბაზო შრიფტად არჩეული გვაქვს **AcadNusx**, ტექსტი ვებ საიტზე გამოჩნდება ქართულად.

შრიფტის სტილები

ტეგი	დასახელება	ფუნქცია
<I> ... </I> <EM> ... </EM>	Italic	დახრილი
<B> ... </B> <STRONG> ... </STRONG>	Bold	მუქი
<U> ... </U>	Underline	ხაზგასმული

_{...}	subscript	ზედაინდექსი
^{...}	superscript	ქვედაინდექსი

ჰორიზონტალური ხაზი

საჭიროების შემთხვევაში web გვერდის ჰორიზონტალური გაყოფა შესაძლებელია „ჰორიზონტალური ხაზის“ გამოყენებით, რომლის აღწერა ხდება **<HR/>** - Horizontal rule ტეგით, რომელიც ასევე თვითდახურვად ტეგს წარმოადგენს.

შესაძლებელია ხაზის ზომების მითითება შემდეგი ატრიბუტების დახმარებით:

SIZE - ხაზის სისქე;

WIDTH – ხაზის სიგრძე.

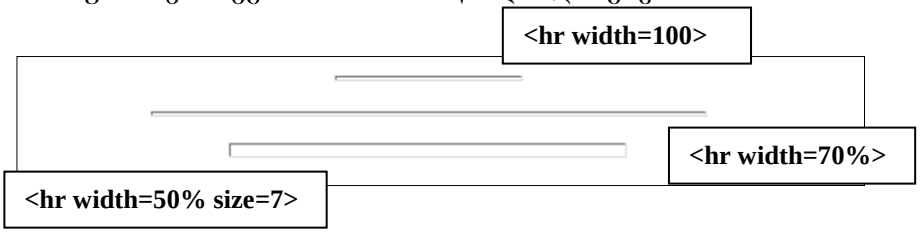
ALIGN - მდებარეობა

NOSHADE - ჩრდილის მოშორება, მკვეთრს ხდის

COLOR – ახალი თაობის ბრაუზერები აქვთ ამ ატრიბუტის მხარდაჭერა.

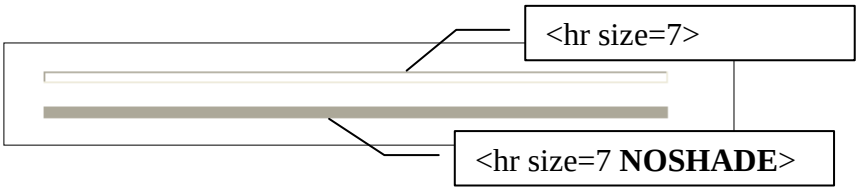
გამოიყენება ხაზის სიგრძის მითითების ორი ვარიანტი:

1. რიცხვითი მნიშვნელობა, რომელიც მიეთითება პიქსელებში (1 Pixel = 1 წერტილი ეკრანზე).
2. პროცენტული მნიშვნელობა (%), რომელიც მიუთითებს ეკრანის რა ნაწილი დაიკავოს ხაზმა.



ხაზის მდებარეობის არჩევა შესაძლებელია **ALIGN=Left / Center / Right** ატრიბუტის საშუალებით. (ავტომატურად ხაზის ჩასმა ხდება ეკრანის ცენტრში).

ხაზის სტილის შეცვლა ხდება **NOSHADE** ატრიბუტით, რომელიც ხსნის ხაზის ჩრდილის ეფექტს.



თავი 3 . CSS

CSS - Cascading Style Sheets

CSS –კასკადური სტილის ცხრილების დახმარებით შესაძლებელია ვებ გვერდის სრული კონტროლი გაცილებით მარტივად და მეტად ფუნქციონალურად, ვიდრე ეს HTML ენის საშუალებით ხდება.

არსებობს სამი სახის CSS:

1) INLINE STYLE SHEETS – ინლაინ სტილი.

ინლაინ სტილის გამოყენებისას სპეციალური ატრიბუტები (css თვისებები) იწერება პირდაპირ HTML ტეგში.

მაგ.

HTML-ში
mogesalmebiT yvelas

CSS-ში mogesalmebiT yvelas

როგორც ვხედავთ ინლაინ სტილების გამოყენებით მივიღეთ უფრო დიდი ტეგი ვიდრე ეს HTML ენის დროს იყო. ამიტომ მათი გამოყენება ხდება მხოლოდ იმ კონკრეტული შემთხვევისათვის, როცა გვინდა ისეთი თვისების გამოყენება რაც არ არის HTML-ში და არის CSS ენაში.

მაგ. *ასოებს შორის დაშორებაა 7*

 Hellooo this is my Page

2) GLOBAL STYLE SHEETS – გლობალური (შიდა) სტილი.

ამ შემთხვევაში სტილის თვისებების აღწერა ხდება ვებ დოკუმენტის დასაწყისში (<head>-ში) და გამოყენება ხდება მთლიანი დოკუმენტის ფარგლებში.

მაგ.

```
<html>
<title>gtu</title>
<style type=text/css>
   H1 {font-family: verdana; font-size: 16; font-weight: bold}
   H2 {font-family: arial; font-size: 16; color: white}
   body {background-color: gray}
</style>
<body>
<p align="center"> <h1> Hellooooooo </h1> </p>
<p align="center"> <h2> hi I'm White </h2> </p>
</p>
</body>
</html>
```

3) LINKED STYLE SHEETS - დაკავშირებული სტილები.

დაკავშირებული სახის სტილის გამოყენება მიზანშეწონილია და მოსახერხებელი, როცა ვქმნით ერტიდაიგივე სტილის რამოდენიმე ვებ გვერდს.

ამ შემთხვევაში სტილის აღწერა ხდება ცალკე ფაილში, რომელსაც შემდეგ ვიმახსოვრებთ css-ფორმატით ანუ file_name.css (მაგ. style.css) და შემდეგ ნებისმიერი ვებ გვერდის აღწერის დროს შეგვიძლია გამოვეყენოთ style.css ფაილში არსებული სტილები.
მაგ. თავდაპირველად ვქმნით ფაილს – **style.css** სადაც უნდა მოვახდინოთ სტილების აღწერა შემდეგნაირად:

```
body {background: gray; font-size: 12pt; font-family: Arial}
.geo {font-family: acadnusx; font-size: 16; color: blue; font-
      style: italic}
#bold {font-weight:bold}
```

სტილების აღმწერ ფაილზე (style.css) მიმართვა ხდება ასე: **<LINK rel="stylesheet" type="text/css" href="styles.css">** ტეგის საშუალებით (href ატრიბუტში მიეთითება სრული გზა).

შევქმნათ HTML ფაილი (მაგ. index.html), სადაც გამოყენებული იქნება სტილები style.css ფაილიდან

```
<html>
<title> gtu </title>
<LINK rel="stylesheet" type="text/css" href="styles.css">
<body>
<p class=”geo” id=”bold”>gamarjobaT! </p>
</body>
</html>
```

CSS სტრუქტურა

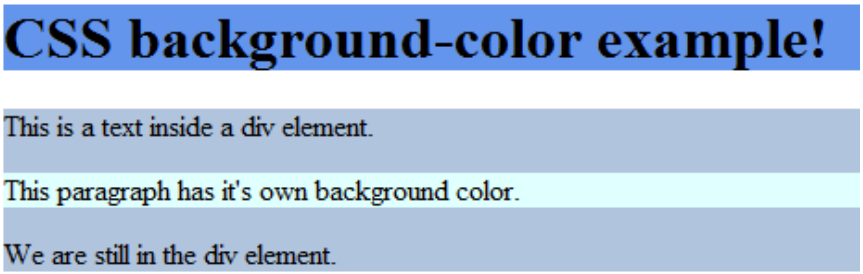
არსებობს სტილების გამოცხადების რამოდენიმე მეთოდი:

1) selector {property: value}

selector არის ჩვეულებრივ HTML ელემენტი (ტეგი ან ტეგთა ჯგუფი, კლასი, ინდივიდუალური სტილი (ID)), რომელის გარკვეულ თვისებებს (property) ენიჭერა კონკრეტული მნიშვნელობები (value).

მაგ.

```
<html>
<head>
<style type="text/css">
h1{background-color:#6495ed;}
p{background-color:#e0ffff;}
div{background-color:#b0c4de;}
</style>
</head>
<body>
<h1>CSS background-color example!</h1>
<div>This is a text inside a div element.
<p>This paragraph has it's own background color.</p>
We are still in the div element.
</div>
</body>
</html>
```



2) .class_name {property: value}

კლასის სახელად შეგვიძლია მიუთითოდ ნებისმიერი სიტყვა. კლასის გამოძახება ხდება **class=”clas_name”** ატრიბუტით.

მაგ.

```
.center  { text-align:center;}
```

გამოძახება:

```
<h1 class="center">Center-aligned heading</h1>
<p class="center">Center-aligned paragraph.</p>
```

ნუ კლასები მოსახერხებელია როცა გვინდა სხვადასხვა ტეგისათვის ერთიდაიგივე თვისების(თვისებების) მინიჭება.

3) #id_name {property: value}

კლასისაგან განსხვავებით ID სელექტორი გამოიყენება მხოლოდ ერთი უნიკალური ელემენტისათვის სტილის განსაზღვრისათვის.

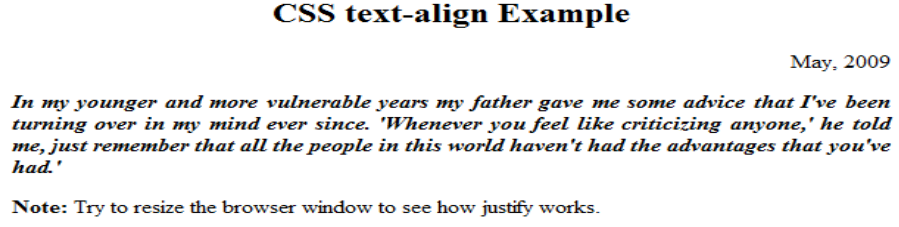
აგრეთვე შესაძლებელია უკვე აღწერილ კლასს დავამატოთ რომელიმე თვისება, მაშინ ვქმნით ე.წ. ინდივიდუალურ სტილს – ID, რომლის გამოყენება HTML ელემენტში ხდება **ID=id_name** ატრიბუტით.

მაგ.

```
<html>
<head>
<style type="text/css">
.center {text-align:center}
.date {text-align:right}
.main {text-align:justify}
#text {font-weight: bold; font-style: italic}

</style>
</head>
```

```
<body>
<h2 class="center">CSS text-align Example</h2>
<p class="date">May, 2009</p>
<p class="main" id="text">In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, just remember that all the people in this world haven't had the advantages that you've had.'</p>
<p><b>Note:</b> Try to resize the browser window to see how justify works.</p>
</body>
</html>
```

გვექნება:



CSS text-align Example

May, 2009

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, just remember that all the people in this world haven't had the advantages that you've had.'

Note: Try to resize the browser window to see how justify works.

Background Properties – ფონის თვისებები CSS-ში

CSS –ში გამოიყენება ფონის შემდეგი თვისებები:

background-color
background-image
background-repeat
background-attachment
background-position

თვისება	აღწერა	მნიშვნელობა	CSS
background	აცხადებს ფონის თვისებებს	*background-color* *background-image* *background-repeat* *background-attachment* *background-position* inherit	1
background-attachment	იძლევა ფონის ფიქსირების ან სქროლის საშუალებას	scroll fixed inherit	1
background-color	ფერის ჩასმა ფონად	*color-rgb* *color-hex* *color-name* transparent inherit	1
background-image	ფონად სურათის ჩასმა	url(*URL*) none inherit	1
background-position	ფონად ჩასმული სურათის მდებარეობის არჩევა	top left top center top right center left center center center right bottom left bottom center bottom right *x% y%* *xpos ypos* inherit	1
background-repeat	გამეორდეს თუ არა ფონი და თუ კი რომელი მიმართულებით	repeat repeat-x repeat-y no-repeat inherit	1

Fonts Properties - შრიფტის თვისებები

თვისება	აღწერა	მნიშვნელობა	CSS
font	აცხადებს ფონტის თვისებებს	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar inherit	1
font-family	შრიფტის არჩევა	*family-name* *generic-family* inherit	1
font-size	შრიფტის ზომა	xx-small x-small small medium large x-large xx-large smaller larger *length* % inherit	1
font-style	შრიფტის სტილი	normal italic oblique inherit	1
font-variant	პატარა ასოები	normal small-caps inherit	1
font-weight	შრიფტის სისქე	normal bold bolder lighter 100 200 300 400	1

Text Properties - ტექსტის თვისებები

თვისება	აღწერა	მნიშვნელობა	CSS
color	ფერი	*color*	1
direction	მიმართულება	ltr rtl	2
line-height		normal *number* *length* %	1
letter-spacing	ასოებს შორის ადგილი	normal *length*	1
text-align	ტექსტის სწორება	left right center justify	1
text-decoration	ტექსტის დეკორაცია	none underline overline line-through blink	1
text-transform	ტექსტის გარდაქმნა	none capitalize uppercase lowercase	1
word-spacing	სიტყვებს შორის ადგილი	normal *length*	1
vertical-align	ვერტიკალურად სწორება	baseline sub super top text-top middle bottom text-bottom *length* %	1

ფსევდო კლასები(მაგალითი)

```
<html>
<head>
<style type="text/css">
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
</head>

<body>
<p>Mouse over the links to see them change layout.</p>

<p><b><a class="one" href="decorate.html"
 target="_blank">This link changes color</a></b></p>
<p><b><a class="two" href="decorate.html"
 target="_blank">This link changes font-
 size</a></b></p>
<p><b><a class="three" href="decorate.html"
 target="_blank">This link changes background-
 color</a></b></p>
<p><b><a class="four" href="decorate.html"
 target="_blank">This link changes font-
 family</a></b></p>
```

```
<p><b><a class="five" href="decorate.html"
 target=" blank">This link changes text-
 decoration</a></b></p>
</body>

</html>
```

This link changes color

This link changes font-size

This link changes background-color

This link changes font-family

This link changes text-decoration

ფსევდო ელემენტები(მაგალითი)

```
<html>
<head>
<style type="text/css">
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
font-variant:small-caps;
}
</style>
</head>

<body>
```

```
<p>You can combine the :first-letter and :first-line
 pseudo-elements to add a special effect to the first
 letter and the first line of a text!</p>
</body>
</html>
```

You can combine the :first-letter and :first-line pseudo-elements to add a special effect to the first letter and the first line of a text!

CSS3

CSS3 მთლიანად უკუთავსებადია, ანუ უკვე არსებული დიზაინის გადაკეთება არ მოგიწევთ, ბრაუზერებს სულ ექნებათ CSS2-ის მხარდაჭერა. CSS3 დაყოფილია მოდულებად ძველი მოდიფიკაცია დაყვეს უფრო მომცრო ნაწილებად და ახალი თვისებები დაუმატეს. CSS3-ის რამოდენიმე მნიშვნელოვანი მოდულები შემდეგია: Selectors, Box Model, Backgrounds and Borders, Text Effects, 2D/3D Transformations, Animations, Multiple Column Layout, User Interface.

შენიშვნის სახით გავითვალისწინოთ რომ CSS3 სპეციფიკაციები ჯერ ისევ შექმნა/განვითარების პროცესშია, თუმცა მისი სიახლეები თანამედროვე ბრაუზერებში უკვე შეტანილია (არა ყველა და არა ყველგან ☺)

მომრგვალებული კუთხეების ეფექტი

CSS3-ის border-radius თვისება საშუალებას აძლევს დიზაინერებს მოამრგვალონ სასურველი ელემენტის კუთხეები, სკრიპტების, სურათების და მრავალჯერადი div ტეგების გამოყენების გარეშე. დაწყებული 2005-დან boder-radius თვისება ფართოდ გავცელდა და ბრაუზერებმაც უზრუნველყვეს მისი მხარდაჭერა.

განვიხილოთ მაგალითი:

შევნიშნოთ რომ ის იმუშავებს Firefox, Safari/Chrome, Opera and IE9 ვერსიებისათვის.

კოდი საკმაოდ მარტივია:

```
#example1 {
border-radius: 15px;
}
```

შევნიშნოთ რომ Firefox -ისათვის დაგჭირდებათ - moz – პრეფიქსი:

```
#example1 {
-moz-border-radius: 15px;
border-radius: 15px;
}
```

<u>მუშაობის პრინციპი</u>

მომრგვალებული კუთხეები შესაძლებელია შეიქმნას ერთმანეთისაგან სრულიად დამოუკიდებლად ოთხი border-*-radius თვისების გამოყენებით (border-bottom-left-radius, border-top-left-radius, ა.შ) ან ოთხივესათვის ერთდროულად შემოკლებული border-radius თვისების მეშვეობით.

განვიხილოთ სინტაქსი border-*-radius თვისების და შემდეგ გადავიდეთ border-radius შემოკლებაზე.

border-bottom-left-radius,

border-bottom-right-radius,

border-top-left-radius,

border-top-right-radius

border-*-radius თვისებას შესაძლებელია მივანიჭოთ მნიშვნელობები როგორც პიქსელებში ისე პროცენტებში, მნიშვნელობების რაოდენობა შეიძლება იყოს ერთი ან ორი იქნება

სინტაქსი:

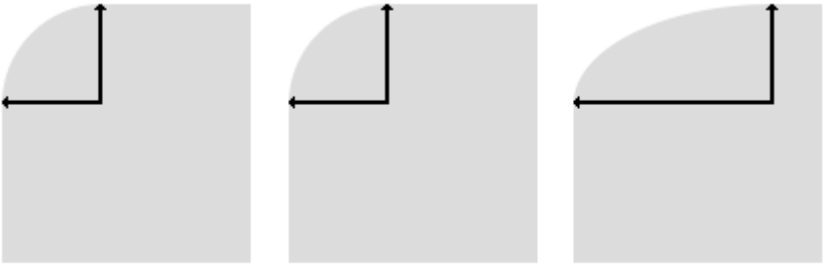
border-*-*-radius: [<*სიგრძე*> | <%>] [<*სიგრძე*> | <%>]?

მაგალითი:

```
border-top-left-radius: 10px 5px;
border-bottom-right-radius: 10% 5%;
border-top-right-radius: 10px;
```

როდესაც გვაქვს ორი მნიშვნელობა, ისინი შესაბამისად განსაზღვრავენ ჰორიზონტალურ და ვერტიკალურ რადიუსებს, როდესაც მხოლოდ ერთი მნიშვნელობაა მითითებული, ის გამოიყენება როგორც ჰორიზონტალური ისე ვერტიკალური რადიუსის განსაზღვრისათვის.

მაგალითში ნაჩვენებია სხვადასხვა რადიუსის შემთხვევაში როგორი ეფექტი გვექნება:



თუ მნიშვნელობა იქნება 0, კუთხე იქნება იქნება და არა მომრგვალებული

border-radius

border-radius შემოკლებული თვისება შესაძლებელია გამოყენებულ იქნას ოთხივე კუთხის თვისების ერთდროულად განსაზღვრისათვის. იგი იღებს ერთ ან ორ წყვილს მნიშვნელობებისა (წყვილი შედგება ერთი-ოთხი მნიშვნელობისაგან) რომლებიც განსაზღვრავენ ჰორიზონტალურ და ვერტიკალურ რადიუსებს

სინტაქსი:

```
border-radius: [ <სიგრძე> | <პროცენტულობა> ]{1,4} [ / [
<სიგრძე> | <პროცენტულობა>  ]{1,4} ]?
```

Examples:

```
border-radius: 5px 10px 5px 10px / 10px 5px 10px 5px;
border-radius: 5px;
border-radius: 5px 10px / 10px;
```

პირველი ოთხეული განსაზღვრავს ჰორიზონტალურ რადიუსებს ოთხივე კუთხისათვის , '/'-ის შემდგომ არსებული არააუცილებელი ოთხეული კი ვერტიკალურ რადიუსებს ოთხივე კუთხისათვის, თუ მხოლოდ ერთი წყვილია მითითებული, ისინი თანაბრად განსაზღვრავენ როგორც ჰორიზონტალურ ისე ვერტიკალურ რადიუსებს.

თითოეული წყვილის კომპონენტებისათვის სამართლიანია შემდეგი:

თუ წყვილის ოთხივე კომპონენტი მითითებულია ე.ი შესაბამისად განსაზღვრულია: top-left, top-right, bottom-right and bottom-left რადიუსები. თუ bottom-left გამოტოვებული ე.ი ის იგივეა რაც top-right, ხოლო თუ bottom-right არის გამოტოვებული, იგივე იქნება რაც top-left, თუ მხოლოდ ერთი მნიშვნელობა იქნა მითითებული, ის გამოიყენება ოთხივე რადიუსისათვის.

<u>ბრაუზერების მხარდაჭერა</u>

თანამედროვე ბრაუზერები Opera (ვერსია 10.5 -ზევით), Safari (ვერსია 5 და ზემოთ) და Chrome (ვერსია 5 და ზემოთ) უზრუნველყოფენ ინდივიდუალური border-*-radius თვისების

მხარდაჭერას ისე როგორც განსაზღვრულია W3C სპეციფიკაციით (თუმცა გარკვეული ბაგები შეიმჩნევა მაინც).

Safari და Chrome უზრუნველყოფენ border-radius მხარდაჭერას -webkit- პრეფიქსით დაწყებული მე-3 ვერსიიდან, მე-5- დან პრეფიქსი აღარ ჭირდება.

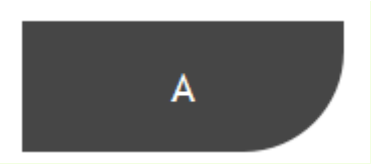
-moz- პრეფიქსი

Mozilla's Firefox გააჩნია border-radius თვისების მხარდაჭერა, -moz- პრეფიქსით დაწყებული 1.0. ვერსიიდან, თუმცა მხოლოდ 3.5 დაწყებული უზრუნველყოფს ელიფსური კუთხეების მხარდაჭერას. ანუ კუთხისათვის ორ მნიშვნელობას ჰორიზონტალური და ვერტიკალური რადიუსებისათვის

W3C სპეციფიკაცია	Mozilla სპეციფიკაცია
border-radius	-moz-border-radius
border-top-left-radius	-moz-border-radius-topleft
border-top-right-radius	-moz-border-radius-topright
border-bottom-right-radius	-moz-border-radius-bottomright
border-bottom-left-radius	-moz-border-radius-bottomleft

შევნიშნოთ რომ მოზილა ცოტა განსხვავებულად აღიქვამს კიდევაც ზემოაღნიშნულ თვისებებს, შეგიძლიათ დატესტოთ ☺

ქვემოთ მოყვანილი მაგალითი იმუშავებს ბრაუზერების თანამედროვე ვერსიებში:




```
#Example_A
 {
height: 65px;
width:160px;
-moz-border-radius-bottomright: 50px;
border-bottom-right-radius: 50px;
}
```


```
#Example_B

{
height:65px;
width:160px;
-moz-border-radius-bottomright: 50px 25px;
border-bottom-right-radius: 50px 25px;
}

#Example_C
```


```
{
height: 65px;
width: 160px;
-moz-border-radius-bottomright:25px 50px;
```

```
border-bottom-right-radius: 25px 50px;
}
```


```
#Example_D

{
height: 5em;
width: 12em;
-moz-border-radius: 1em 4em 1em 4em;
border-radius: 1em 4em 1em 4em;
}
```


```
#Example_E

{
height:65px;
width:160px;
-moz-border-radius: 25px 10px / 10px 25px;
border-radius: 25px 10px / 10px 25px;
}
```


```
#Example_F
```

```
{
height:70px;
width:70px;
-moz-border-radius:35px;
border-radius:35px;
}
```

CSS3-ში, box-shadow თვისება გამოიყენება box-ებისათვის ჩრდილის დასამატებლად:
მაგალითად:

```
div
 {
 box-shadow: 10px 10px 5px #888888;
 -webkit-box-shadow: 10px 10px 5px #888888; /* Safari-სათვის */
 }
```

სინტაქსი ასეთია:
box-shadow: h-shadow v-shadow blur spread color inset;

Value	Description
h-shadow	Required. The position of the horizontal shadow. Negative v
v-shadow	Required. The position of the vertical shadow. Negative valu
blur	Optional. The blur distance
spread	Optional. The size of shadow
color	Optional. The color of the shadow. Look at CSS Color Values possible color values
inset	Optional. Changes the shadow from an outer shadow (outs

CSS3 border-image საშუალებას იძლევა სურათის მეშვეობით შექმნათ ჩარჩო:

```
div
 {
 border-image:url(border.png) 30 30 round;
 -moz-border-image:url(border.png) 30 30 round; /* Firefox */
```

-webkit-border-image:url(border.png) 30 30 round; /* Safari და Chrome */
}
სინტაქსი ასეთია:
border-image: source slice width outset repeat;

Value	Description
border-image-source	The path to the image to be used as a border
border-image-slice	The inward offsets of the image-border
border-image-width	The widths of the image-border
border-image-outset	The amount by which the border image area exte box
border-image-repeat	Whether the image-border should be repeated,

Browser Support

Property	Browser Support				
border-radius					
box-shadow				-webkit-	
border-image		-moz-	-webkit-	-webkit-	

Internet Explorer 9 აქვს რამოდენიმე border თვისების მხარდაჭერა.

border-image-ისავის Firefox -ი ითხოვს პრეფიქსს -moz-
Chrome და Safari კი პრეფიქსს -webkit
Safari-ს ჭირდება პრეფიქსი -webkit- box-shadow-ისათვის.
Opera-ს სრული მხარდაჭერა აქვს ☺

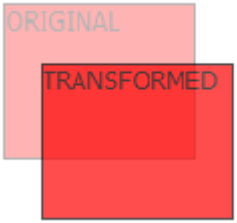
CSS3 ტრანსფორმი

ამ თვისების მეშვეობით შესაძლებელია ელემენტის წაძვრა, მობრუნება, გადიდება დაპატარავება, გაწელვა. ელემენტების ტრანსფორმაცია შესაძლებელია 2D და 3D ტრანსფორმირების მეშვეობით

გადა ცინცაძე, ტატა წილოსანი

განვიხილოთ 2D ტრანსფორმირების მეთოდები:
translate()
rotate()
scale()
skew()
matrix()

translate()მეთოდის მეშვეობით ელემენტი გადაიწევს თავისი მიმდინარე მდებარეობიდან იმ პარამეტრების შესაბამისად რომელიც მიეთითება X და Y ღერძთან მიმართებაში



მაგ:

```
div
 {
 transform: translate(50px,100px);
 -ms-transform: translate(50px,100px); /* IE 9 */
 -webkit-transform: translate(50px,100px); /* Safari and Chrome */
 -o-transform: translate(50px,100px); /* Opera */
 -moz-transform: translate(50px,100px); /* Firefox */
 }
```

rotate() მეთოდის შედეგად ელემენტი ტრიალებს საათის ისრის მიმართულებით მოცემული გრადუსით, უარყოფითი მნიშვნელობების შემთხვევაში მოტრიალება განხორციელდება საათის ისრის საწინააღმდეგო მიმართულებით:

მაგ:

```
div
 {
 transform: rotate(30deg);
 -ms-transform: rotate(30deg); /* IE 9 */
 -webkit-transform: rotate(30deg); /* Safari and Chrome */
 -o-transform: rotate(30deg); /* Opera */
 -moz-transform: rotate(30deg); /* Firefox */
 }
```

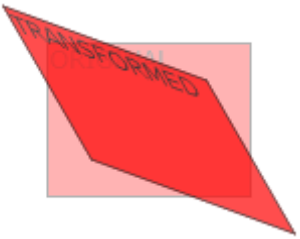
scale() მეთოდი შესაძლებლობას იძლევა ელემენტი გაიზარდოს ან დაპატარავდეს თავის ორიგინალ ზომასთან მიმართებაში შესაბამისი პარამეტრების (სიგრძე =Yღერძი , სიგანე= Xღერძი)შესაბამისად:

მაგ:

```
div
 {
 transform: scale(2,4);
 -ms-transform: scale(2,4); /* IE 9 */
 -webkit-transform: scale(2,4); /* Safari and Chrome */
 -o-transform: scale(2,4); /* Opera */
 -moz-transform: scale(2,4); /* Firefox */
 }
```

scale(2,4) მნიშვნელობა მოახდენს სიგანის 2-ჯერ, ხოლო სიგრძის 4-ჯერ ზრდას (ორიგინალურ ზომასთან მიმართებაში).

skew() მეთოდის მეშვეობით ელემენტი ბრუნდება მითითებული კუთხით, აბსცისთა და ორდინატთა ღერძებთან მიმართებაში, მითითებული პარამეტრების მიხედვით:




```
div
 {
 transform: skew(30deg,20deg);
 -ms-transform: skew(30deg,20deg); /* IE 9 */
 -webkit-transform: skew(30deg,20deg); /* Safari and Chrome */
 -o-transform: skew(30deg,20deg); /* Opera */
 -moz-transform: skew(30deg,20deg); /* Firefox */
 }
```

ელემენტი მობრუნდება 30 გრადუსით X ღერძის გარშემო და 20 გრადუსით Y-ის გარშემო.
matrix() მეთოდი ახდენს ყველა 2D ტრანსფორმირების მეთოდების ერთად გაერთიანებას, ის იღებს 6 პარამეტრს, შეიცავს რა მათემატიკურ ფუნქციებს, საშუალებას იძლევა მოახდინოთ ელემენტის მობრუნება, გადაადგილება, გაწელვა.
მაგალითი:

```
div
 {
 transform:matrix(0.866,0.5,-0.5,0.866,0,0);
 -ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
 -moz-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Firefox */
 -webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Safari and Chrome */
 -o-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Opera */
 }
```

შესავალი JavaScript-ში

რა არის JavaScript?

JavaScript შეიქმნა რომ HTMLგვერდებისათვის ინტერაქტიულობა შეეძინა
JavaScript წარმოადგენს სკრიპტულ ენას, რომელიც პროგრამირების ენის ასე ვთქვათ „მსუბუქი“ ვარიანტია ანუ პროგრამირების ენა როგორც ასეთი არაა,ის ინტერპრეტირების ენაა მისი გაშვება წინასწარი კომპილაციის გარეშე ხდება და არც ლიცენზიას საჭიროებს გამოყენებისათვის.

რა შეუძლია JavaScript-ს რომ გააკეთოს?

HTML დიზაინერებს JavaScript აძლევს პროგრამირების ინსტრუმენტს, როგორც წესი HTML-ის ავტორები არ არიან პროგრამისტები, მაგრამ JavaScript ხომ ნამდვილი პროგრამირების ენა არაა შესაბამისად საკმაოდ მარტივი სინტაქსი გააჩნია.
JavaScript შეუძლია HTML გვერდზე დინამიური ტექსტის ჩასმა, ამისათვის გამოიყენებს document.write("<h1>" + name + "</h1>") ცვლადი ტექსტის ჩასმის შესაძლებლობისათვის .
JavaScript-ს შეუძლია მოვლენებზე „რეაგირება“, მაგალითად როდესაც გვერდი ჩაიტვირთება ან მომხმარებელი დააქლიქებს რაიმე ლინქს.
JavaScript-ს შეუძლია წაიკითხოს ან ჩაწეროს HTML ელემენტები, აგრეთვე შეცვალოს HTML ელემენტის შიგთავსი
JavaScript შესაძლებელია გამოვიყენოთ რომ გავარკვიოთ მომხმარებელი რომელ ბრაუზერს იყენებს, შესაბამისად ჩავტვირთოთ სხვა გვერდი შემუშავებული იმ კონკრეტული ბრაუზერისათვის. იგი აგრეთვე შესაძლებელია cookies-ის გასაკეთებლად, რომ მომხმარებლის კომპიუტერზე შეინახოს და შემდგომ გამოიყენოს ინფორმაცია.

სკრიპტის ჩასმის მაგალითი

```
<html>
<body>

<script type="text/javascript">
document.write("This is my first JavaScript!");
</script>
```

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

სად თავსდება JavaScript

ბრაუზერში გვერდის ჩატვირთვისთანავე ხდება JavaScripts-ის გაშვება, შესაბამისად შესაძლებელია ჩვენ ეს არ გვინდოდეს და გვინდოდეს სკრიპტის გაშვება რაღაცა სხვა მოვლენის დროს, მაგალითად ღილაკზე თითის დაჭერის შემთხვევაში. ასეთ შემთხვევაში სკრიპტი უნდა განთავსდეს ფუნქციაში.

განვიხილოთ სკრიპტები რომლებიც თავსდებიან გვერდის <head> ნაწილში (ფუნქციაში) და გამოიძახებიან კონკრეტული მოვლენის დროს:

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>
//მისი გამოძახება შესაბამისად მოხდება<body>-ში ასე:
 <body onload="message()">
</body>
</html>
```

თუ არ გინდათ რომ სკრიპტი მოათავსოთ ფუნქციაში მისი განთავსება <body>-ში მოგიწევთ:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
```

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

შესაძლებელია შემოუსაზღვრელი რაოდენობის სკრიპტების განთავსება როგორც <head> ისე <body> ნაწილში როგორც ცალ-ცალკე ისე ერთდროულად:

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>

<body onload="message()">
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>

</html>
```

შესაძლებელია აგრეთვე ცალკე გქონდეთ " *.js "ფაილი და ჩასვათ ის სასურველ გვერდში შემდეგნაირად:

```
<html>
<head>
<script type="text/javascript" src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

გაითვალისწინეთ რომ Javascript მგრძნობიარეა დიდი და პატარა ასოების მიმართ

კომენტარები javascript-ში:

```
// ერთხაზიანი
```

/*
რამოდენიმე
სტრიქონიანი
*/
ცვლადების აღწერა: გვჭირდება გასაღები სიტყვა var :
var x;
var carname;
var carname="Volvo"; // შესაძლებელია ცვლადისათვის მნიშვნელობის
//განსაზღვრა მისი აღწერისთანავე

მინიჭების ოპერატორი „=“, შეკრების ოპერატორი „+“ გამოკლება „-„„ გამრავლება „*“, გაყოფა „/“ ინკრემენტი „++“ დეკრემენტი „- -„ სტრინგთან მიმართებაში „+“ წარმოადგენს კონკატენაციის ოპერაციას (გადაბმას).
შედარების ოპერატორები
ცვლადებისათვის მნიშვნელობით x=6 და y=3
&& and (x < 10 && y > 1) ჭეშმარიტი
|| or (x==5 || y==5) მცდარი
! not !(x==y) ჭეშმარიტი
პირობითი ოპერატორი: სინტაქსი
variablename=(condition)?value1:value2
მაგალითი: greeting=(visitor=="PRES")?"Dear President ":"Dear ";

პირობითი ოპერატორები:

if - გამოიყენება თუ კონკრეტული პირობის ჭეშმარიტებას ვადგენთ
<script type="text/javascript">
//დაწერს "Good morning" თუ დრო ნაკლებია 10-ზე

var d=new Date();
var time=d.getHours();

if (time<10)
 {
 document.write("Good morning");
 }
</script>

if...else - გამოიყენება თუ კონკრეტული პირობის ჭეშმარიტებას ვადგენთ და უარყოფითი შედეგის შემთხვევაშიც გვსურს გარკვეული ოპერაციის ჩატარება

```
<script type="text/javascript">
//თუ დრო ნაკლებია 10-ზე გექნებათ "Good morning"
//სხვა შემთხვევაში "Good day"

var d = new Date();
var time = d.getHours();

if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

if...else if....else - რამოდენიმე პირობითი ბლოკისათვის

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
```

</script>

switch statement - მრავალი პირობითი ბლოკისათვის

```
<script type="text/javascript">
//მიიღებთ დღეების მიხევით განსზვავებულ მისალმებებს,
გაითვალისწინეთ //რომ Sunday=0,Monday=1, Tuesday=2, და ასე შემდეგ

var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

Alert Box

ის გამოიყენება მომხმარებლისათვის შეტყობინების გამოსატანად, როცა ის გამოჩნდება მომხმარებელს მოუწევს რომ დააჭიროს ღილაკს "OK" რომ გააგრძელოს მუშაობა გვერდთან, შესაბამისად ჩვენც დარწმუნებულები ვიქნებით რომ მან ჩვენი შეტყობინება ნახა
სინტაქსი: alert("sometext");
მაგალითი:

```
<html>
<head>
<script type="text/javascript">
function show_alert()
```

```
{
alert("I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

მსგავსია Confirm Box და Prompt Box მაგალითები:

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show confirm box"
/>

</body>
</html>
```

და

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

ფუნქციის განსაზღვრა:

სინტაქსი ასეთია:

```
function functionname(var1,var2,...,varX)
{
რაიმე კოდი
}
```

მაგალითი:

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
```

```
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

ციკლის ოპერატორები

ციკლის ოპერატორი for, ის გამოიყენება როცა წინასწარ იცით რამდენჯერ გინდათ რომ სკრიპტი დაატრიალოთ, სინტაქსი ასეთია:

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
გასაშვები კოდი
}
```

მაგალითი:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

ციკლის ოპერატორი while

While ოპერატორი ატრიალებს ციკლს სანამ გადაცემული პირობა ჭეშმარიტია. სინტაქსი ასეთია:

```
while (var<=endvalue)
  {
გასაშვები კოდი
  }
```

მაგალითი:

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
 {
 document.write("The number is " + i);
 document.write("<br />");
 i++;
 }
</script>
</body>
</html>
```

ციკლის ოპერატორი do...while არის while-ის სახეობა რომელიც ერთხელ მაინც შესრულდება იმისგან დამოუკიდებლად გადაცემული პირობა ჭეშმარიტია თუ მცდარი, შემდგომ კი მხოლოდ იმ შემთხვევაში თუ პირობა ჭეშმარიტია. სინტაქსი ასეთია:

```
do
 {
 გასაშვები კოდი
 }
while (var<=endvalue);
```

მაგალითი:

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
 {
 document.write("The number is " + i);
 document.write("<br />");
 i++;
 }
while (i<=5);
</script>
```

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