ბ. ღვინეფაძე

WEB-დაპროგრამება

HTML 5

“ტექნიკური უნივერსიტეტი”

საქართველოს ტექნიკური უნივერსიტეტი

გ. ღვინეფაძე

WEB-დაპროგრამება

HTML 5

დამტკიცებულია დამხმარე
სახელმძღვანელოდ სტუ-ს
სარედაქციო-საგამომცემლო
საბჭოს მიერ

თბილისი
2013

უაკ 681.3.06

განხილულია WEB-გვერდებისა და საიტების შესაქმნელად გამიზნული ბაზისური საშუალების HTML ენის ბოლო ვერსია – HTML 5, ასევე, – WEB-დაპროგრამებასთან დაკავშირებული სხვა სიახლეებიც, მათ შორის ინტერნეტის განვითარების მიმართულებანი.

განკუთვნილია ინფორმატიკის სპეციალობათა შემსწავლელი სტუდენტებისა და ამ საკითხებით დაინტერესებულ სხვა პირთათვის.

რეცენზენტები: სრული პროფესორი ო. ნატროშვილი,
სრული პროფესორი თ. სუხიაშვილი

ISBN 978-9941-20-227-8

http://www.gtu.ge/publishinghouse/

HTML 5

შესავალი

საყოველთაოდ ცნობილია, რომ ინტერნეტში განსათავსებელი ინფორმაციისათვის საჭირო სახის მისაცემად (ეს ინფორმაცია, ძირითადად, Web-საიტების სახით არსებობს) ბაზისურ ინსტრუმენტად რჩება HTML ენა. წინამდებარე ნაშრომში ვგულისხმობთ, რომ მკითხველი მეტ-ნაკლებად იცნობს ამ ენას (სასურველია ასევე, რომ იგი ფლობდეს ვებ-დაპროგრამებისათვის განკუთვნილ სხვა საშუალებებსაც). ჩვენი მიზანი გახლავთ, სტუდენტი გავაცნოთ ამ ენის ახალ ვერსიას – HTML 5-ს და დღეს ამ ენასთან მჭიდროდ დაკავშირებული *სტილების მაფორმირებელი კასკადური ცხრილების* – CSS-ის ახალ ვერსიას და ზოგიერთ სხვა საშუალებას.

HTML 5 (HyperText Markup Language, version 5) – ჰიპერტექსტის მონიშვნის ენა – წარმოადგენს საყოველთაოდ ცნობილი HTML (HTML 4.0 - 1997)-ის განვითარების შემდგომ საფეხურს. მასზე მუშაობა 2007 წელს დაიწყო სპეციალისტების ორმა ჯგუფმა[1].

ეს ჯგუფებია:

- Apple, Opera და Mozilla Fundation პოპულარული ბროუზერების დამპროექტებელი სპეციალისტებისაგან ჯერ კიდევ 2004 წელს შემდგარი მუშა ჯგუფი WHATWG (Web Hypertext Application Technology Working Group). მათ, ინტერნეტის განვითარების მიზნით, მუშაობა დაიწყეს Web Applications 1.0 სახელწოდების პროექტზე, რომლის იდეებიც შემდგომ გამოყენებული იქნა ამავე ჯგუფის მიერ შემუშავებულ ახალ სტანდარტში Web Forms 2.0. სწორედ ეს სტანდარტი იქცა HTML 5 ენის ერთ-ერთ წყაროდ.

[1] 2012 წლის სექტემბრის მონაცემებით ეს მუშაობა გრძელდება. სპეციალისტების ვარაუდით, იგი 2014 წლისათვის უნდა დამთავრდეს.

- W3C (World Wide Web Consortium) – ინტერნეტის სტანდარტებზე მომუშავე მსოფლიო ორგანიზაცია, რომლის ხელმძღვანელია მსოფლიოში ცნობილი სპეციალისტი ტიმოთი ჯონ ბერნერს-ლი. (აღსანიშნავია, რომ სწორედ ბერნერს-ლი გახლავთ HTML ენის შემქმნელი). აღნიშნული ჯგუფის დამსახურება არის ის, რომ HTML 5 ენა მთლიანად შეესაბამება პროგრამირების სფეროში დღეს ფრიად პოპულარული XML ენის სტანდარტებს.

HTML 5-ის შექმნის მიზანი გახლავთ HTML ენის გაუმჯობესება უახლესი მულტიმედიური საშუალებების მხარდაჭერის მიმართულებით, ამასთან, HTML-კოდის აღქმის გაადვილება როგორც ადამიანის, ისე – WEB-ბროუზერისათვის. ხაზგასასმელია კიდევ ერთი მნიშვნელოვანი მომენტი – HTML 5 თავსებადია თავის წინამორბედებთან HTML 4.01-სა და XHTML 1.1-თან, ხოლო ბროუზერების ძველი ვერსიების მიერ უბრალოდ იგნორირება ხდება HTML 5-ში შემოტანილი სიახლეების (ელემენტების).

HTML 4 ვერსიას უკვე გავეცანით. სანამ დეტალურად შევისწავლიდეთ საკითხს, თუ რა განასხვავებს HTML 5-ს მისი უშუალო წინამორ-ბედისაგან, სასარგებლოდ მიგვაჩნია, მკითხველი გაეცნოს XHTML სახელით ცნობილ ამ ორ ვერსიას შორის არსებულ, ასე ვთქვათ, შუალედურ რგოლს, რომელიც, ჯერ ერთი, დღესაც წარმატებით გამოიყენება საიტების შესაქმნელად და მეორეც – თავისი არსით იგი თავისთავადაც წარმოადგენს ვებ-დამპროექტებლებისთვის ინტერესების სფეროს.

XHTML

XHTML (***E**xtensible **H**yper**t**ext **M**arkup **L**anguage* — ჰიპერტექსტის მონიშვნის გაფართოებული ენა) გახლავთ HTML ტექნოლოგიების განვითარების მიმართულებით გადადგმული შემდგომი ნაბიჯი. თავისივე სახელწოდებიდან გამომდინარე, იგი განკუთვნილია HTML ენის (დავაზუსტოთ – HTML 4 ენის) შესაძლებლობების გაფართოებისათვის.

გარკვეული თვალსაზრისით, XHTML ენა შესაძლებელია განვიხილოთ, როგორც შუალედური გადაწყვეტილება აღნიშნული მიმართულებით, რადგან დღეს უკვე ნათლად ჩანს, რომ სულ მალე ამ ასპარეზზე სრულ უპირატესობას მოიპოვებს HTML 5 ტექნოლოგია *(ამ ენის შესაძლებლობების შესწავლას ქვემოთ განვაგრძობთ).*

XHTML ენის პირველი ვერსიის სპეციფიკაცია – XHTML-1.0 – ამ საუკუნის დასაწყისში გამოქვეყნდა W3C კონსორციუმის მიერ. შემდეგ მას მოჰყვა რიგი მოდიფიკაციებისა. 2010 წლისათვის ბოლო ვერსიას წარმოადგენდა XHTML-2.0. შემდგომ კი ახალ ვერსიებზე მუშაობა შეწყდა და მიზნად იქნა დასახული, პროგრამისტების ძალები და სხვა რესურსები დაესაქმებინათ HTML 5 ტექნოლოგიის შექმნაზე.

მთავარი ნიშან-თვისება, რომელიც XHTML-ს განასხვავებს წინამორბედისაგან – HTML 4-საგან, ის გახლავთ, რომ XHTML-ზე დაწერილი დოკუმენტის დამუშავება ხდება ისეთივე პარსერით, როგორიც განკუთვნილია XML-ენაზე შექმნილი კოდების გასაანალიზებლად (ამ ენას მომდევნო კურსზე შევისწავლით). აღვნიშნავთ, რომ ამ პროცესში დოკუმენტის დამპროექტებელთა მიერ დაშვებული შეცდომების გასწორება არ ხდება.

შენიშვნა: პარსერი (სინტაქსური ანალიზატორი) არის მოდული, რომლითაც ტარდება პროგრამული კოდის სინტაქსური ანალიზი – შეესაბამება თუ არა კოდის შემადგენელი ნაწილები (ლექსემები) მოცემული ენის ფორმალურ გრამატიკას. აქვე უნდა აღვნიშნოთ, რომ

პარსერის არჩევაზე არავითარ გავლენას არ ახდენს DOCTYPE დეკლარაციის შემცველობა.

ბროუზერი დოკუმენტის დასამუშავებლად პარსერის არჩევანს ახდენს სერვერისაგან მოწოდებული, content-type დასათაურებაში შემავალი შემდეგი ინფორმაციის საფუძველზე:

- HTML – text/html
- XHTML – application/xhtml+xml
- კლიენტზე ლოკალური დათვალიერების წესი განისაზღვრება ფაილის გაფართოების მიხედვით.
- Internet Explorer-ში XHTML დოკუმენტების დათვალიერება შესაძლებელია ბროუზერის მხოლოდ მე-9 ვერსიიდან.
- ხშირ შემთხვევაში, HTML ენისაგან განსხვავებით, XHTML-ის პარსერი ჩვენ მაგივრად არ იღებს გადაწყვეტილებას – W3C კონსორციუმის რეკომენდაციით, კოდში შეცდომის აღმოჩენის შემთხვევაში ბროუზერმა უნდა შეწყვიტოს დოკუმენტის დამუშავება და მომხდარი მოვლენის შესახებ შეტყობინება გასცეს.

ჩამოვთვალოთ ის მოთხოვნები, რომლებსაც უნდა აკმაყოფილებდეს XHTML დოკუმენტი (ფაქტობრივად, თავისებურებანი, რომლებიც მას განასხვავებს HTML დოკუმენტისაგან):

- ტეგებისა და ატრიბუტების სახელები იწერება მხოლოდ და მხოლოდ პატარა ასოებით, მაგალითად, ასე: და არა როგორც <IMG ALT="" />).
- ყველა ელემენტი აუცილებლად უნდა დაიხუროს ბოლო ტეგით. ამასთან, ცარიელი (დროებით მნიშვნელობის არმქონე) ან ერთტეგიანი ელემენტები შემდეგი სახით უნდა წარმოვადგინოთ:
, <p />.
- ბულის ტიპის ატრიბუტების სახელწოდებები და მნიშვნელობები იწერება მხოლოდ გაშლილი სახით. მაგალითად, ამგვარად: <option selected="selected"> ან <td nowrap="nowrap">.

- დუმილით, XHTML დოკუმენტისათვის კოდირების სისტემად არჩეულია UTF-8 (HTML ენაში კი ასეთად მიიჩნეოდა ISO 8859-1).

არის კიდევ სხვა, შედარებით ნაკლებად მნიშვნელოვანი ნიუანსებიც, მაგალითად, ის, თუ რა განასხვავებს ერთმანეთისაგან XHTML-დოკუმენტის ტიპებს (ამ ინფორმაციის მიღება ინტერნეტიდანაც შეიძლება).

შევნიშნავთ, რომ DTD – დასამუშავებელი დოკუმენტის ტიპის განსაზღვრება – დოკუმენტის თავშივე მოიცემა.

ქვემოთ მოყვანილია XHTML დოკუმენტისათვის განსაზღვრული ყველაზე გავრცელებული ტიპები:

XHTML 1.0 Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
```

XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

XHTML 1.0 Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
frameset.dtd">
```

XHTML 1.0 Mobile

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile
1.0//EN" "http://www.wapforum.org/DTD/xhtml-
mobile10.dtd">
```

XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

რაც შეეხება სახელების სივრცეს xhtml ტეგებისათვის, იგი შემდეგი სახით მოიცემა დოკუმენტის ფესვურ ტეგში:

"http://www.w3.org/1999/xhtm"

(სახელების სივრცის შესახებ ინფორმაცია იხ. დანართ №1-ში)

განვიხილოთ კიდევ ერთი ფრიად მნიშვნელოვანი საკითხი, რომელიც ეხება ინტერნეტის სფეროში ბოლო ხანებში გამოვლენილ ფუნდამენტური სახის ტენდენციებს – მხედველობაში გვაქვს ე.წ. **WEB 2.0** სახელით ცნობილი საიტების, ბიზნეს-გამოყენებების შექმნის ახალი კონცეფცია.

საერთოდ, როცა განიხილავენ HTML 5-სა და სხვა თანამედროვე WEB-ტექნოლოგიებს, მათ მოიაზრებენ, როგორც ე.წ. WEB 2.0 კონცეფციის განმახორციელებელ საშუალებებს (მეტიც, დღეს უკვე საუბარია WEB 3.0 კონცეფციაზეც). ბუნებრივია, დავინტერესდეთ, როგორ განმარტებას აძლევენ სპეციალისტები WEB 2.0 სიახლეს ინტერნეტის სფეროში.

WEB 2.0

დღეს ინტერნეტი ისე ფართოდ არის შემოჭრილი ადამიანის ცხოვრებაში, რომ სპეციალისტებმა დააყენეს საკითხი – მისი რესურსებით მომსახურებას რაც შეიძლება მასობრივი, ეფექტიანი და ამავე დროს სტანდარტული (აქ იგულისხმება მასთან ურთიერთობის წესების ადვილად ათვისების შესაძლებლობის უზრუნველყოფა) სახე მიეცეს. ასეთი მიდგომის სახელდებისათვის ბოლო ხანებში სპეციალური ტერმინებიც შემოიღეს – *პროგრამული უზრუნველყოფა, როგორც სერვისი* (Saas - software as a service) და ასეთი პოეტურიც კი – *ღრუბლისებრი გამოთვლები* (cloud computing). სწორედ, დროის ასეთი მოთხოვნების

გასათვალისწინებლად იქნა ბოლო წლებში შემუშავებული WEB 2.0-ის სახელით ცნობილი ახალი კონცეფცია ინტერნეტისათვის.

ცნება WEB 2.0-ისათვის ნათლია გახლავთ ამერიკელი ტიმ ო'რეილი – აღნიშნული ტერმინი პირველად მის მიერ იქნა გამოყენებული 2005 წლის ოქტომბერში გამოქვეყნებულ სტატიაში.

ო'რეილმა WEB 2.0 ტერმინის არსი ასე განმარტა:

WEB 2.0 არის კომპიუტერული სისტემების დაპროექტების ისეთი მეთოდი, რომელიც ორიენტირებულია ქსელურ ურთიერთქმედებებზე შემდეგი გათვლით – რაც მეტია პროექტის (გამოყენების) მომხმარებელი, მით უფრო ნათლად უნდა გამოიკვეთოს ამ პროექტის უპირატესობა მანამდე არსებულ სისტემებთან შედარებით.

ამრიგად, აქ საუბარია საკითხისადმი გლობალურ – პრინციპული სახის, შეიძლება ითქვას, ფილოსოფიურ მიდგომაზე და არა საიტების დამუშავების პროცესის ეფექტიანობის ასამაღლებლად რაიმე კონკრეტული სახის რეკომენდაციების გაცემაზე.

ახალი კონცეფციის შემუშავების საჭიროება განაპირობა მანამდე არსებული მიდგომის (მას, შესაბამისად, WEB 1.0 კონცეფცია დაარქვეს) კრახმა, რომელზე დაყრდნობითაც უმსხვილესი ინტერნეტ-კომპანიები ცდილობდნენ მთელი ეს უზარმაზარი ინტერნეტ-სივრცე ცენტრალიზებული, მკაცრად განსაზღვრული წესებითა და ასეთივე შეზღუდვების შემოღებით ემართათ. ამგვარი მიდგომის მარცხი ნათელი გახადა შემდეგი სიტუაციის წარმოშობამ – ინტერნეტ-სივრცის ცენტრალიზებული წესებით მართვის მოსურნე ინტერნეტ-კომპანიებმა ვერ მოახერხეს ბაზრის დაპყრობა, ვერ გაუწიეს კონკურენცია უამრავ მცირე მასშტაბის კომპანიას, რომელთაც ინტერნეტი აავსეს "წესების გარეშე" მოქმედი წვრილ-წვრილი საიტებით (რასაც ხატოვნად შემდგომ "გრძელი კუდის" კონცეფცია ეწოდა).

ჩამოვთვალოთ ის ძირითადი ტენდენციები (რეკომენდაციები), რომელთა ბაზაზე, WEB 2.0 კონცეფციის მიხედვით, უნდა მოხდეს საპროექტო გადაწყვეტილებების მიღება:

- ახალ პროექტთან (დანართთან) მუშაობა არ უნდა მოითხოვდეს მომხმარებლისაგან რაიმე დამატებითი ზომების მიღებას, ანუ როგორც აპარატურის, ისე პროგრამული ნაწილის ამ თუ იმ სახით წინასწარ მომზადებას – მასთან სამუშაოდ სავსებით საკმარისი უნდა იყოს მხოლოდ ბროუზერის შესაძლებლობანი.
- დანართები უნდა უზრუნველყოფდეს შემდეგი სახის WEB-სერვისს – მათ უნდა შეეძლოთ ერთმანეთთან "საერთო ენის" გამონახვა, რაც გულისხმობს ერთმანეთის შესაძლებლობების გამოყენებას.
- კონცეფცია იყენებს ე.წ. Mash – up მიდგომასაც – საჭირო WEB-სერვისის შექმნა შესაძლებელი უნდა იყოს რამდენიმე სხვა WEB-სერვისის ინტეგრაციის შედეგად.
- სურვილის შემთხვევაში შესაძლებელი უნდა იყოს მომხმარებლებს შორის დიალოგის უზრუნველყოფა, ე.წ. სოციალიზაცია – კოლექტიური ინტელექტისათვის ასპარეზის მიცემა. WEB 2.0 კონცეფციის მიხედვით შექმნილი დანართები ხშირად უბრალოდ შუამავლის როლში გამოდის მომხმარებლებს შორის, რომლებიც თვითონ განსაზღვრავენ მათთვის საინტერესო კონტენტს.
- იზრდება ტეგების როლი. მათი მეშვეობით ხდება კონტენტის ცალკეული უბნების დახარისხება-წვდომა და რელევანტური ინფორმაციის მოსაძიებლად საჭირო გარემოს შექმნა.

ზემოთ მოყვანილი ჩამონათვალი არ გახლავთ მოთხოვნების სრული სია – აქ ყურადღება გამახვილებულია მხოლოდ WEB 2.0 კონცეფციისათვის დამახასიათებელ ძირითად ტენდენციებზე.

აღვნიშნავთ, რომ WEB 2.0 კონცეფცია ინარჩუნებს ყველა იმ საღი აზრის მოთხოვნასაც, რომელთა გათვალისწინება რეკომენდებული იყო WEB-დაპროგრამების "ადრეულ ეპოქებშიც".

მოვიყვანოთ ამ მოთხოვნების სიაც:

- საიტების სტრუქტურის, დიზაინის იმგვარად სრულყოფა, რომ რაც შესაძლებელია მეტად გაადვილდეს უმთავრესი ინფორმაციის მოძიება, მომხმარებელთან ინტერფეისი იყოს მაქსიმალურად მეგობრული, საიტთან ურთიერთობის ფორმატები მომხმარებელს უნდა მოაგონებდეს უკვე ნაცნობ გარემოს, სიტუაციებს, რათა ზედმეტად ორიგინალურმა გადაწყვეტებმა მას მოცემულ საიტთან მუშაობის სურვილი არ გაუქროს.
- საიტი ბროუზერში მაქსიმალურად სწრაფად უნდა იტვირთებოდეს.
- საიტი, უპირველეს ყოვლისა, ოპტიმიზებული უნდა იყოს ძირითადი ამოცანის გადაწყვეტაზე.
- საიტის დიზაინი და შინაარსი ერთმანეთთან ორგანულად უნდა იყოს დაკავშირებული.
- ინფორმაცია უნდა იყოს, რამდენადაც შესაძლებელია, ადვილად აღქმადი და გრამატიკულად გამართული, ყოველ-გვარი ზედმეტი ელემენტებით გადატვირთვის გარეშე. შემჩნეულია, რომ საიტსაც "ტანსაცმლის (აქ გაფორმების) მიხედვით ხვდებიან".

დასასრულ, ვიტყვით, რომ WEB 2.0 კონცეფციის არსს მოკლედ სპეციალისტები ასეც განმარტავენ:

ამ კონცეფციის მიხედვით შექმნილი საიტი შესაძლებლობას უნდა იძლეოდეს, რომ მისი კონტენტი მთლიანად თუ არა, უმეტესწილად მაინც მომხმარებლის სურვილების მიხედვით განისაზღვრებოდეს.

ვაგრძელებთ საუბარს HTML 5 ენის შესახებ:

HTML 5-ში შემოტანილი სიახლეებიდან, უპირველეს ყოვლისა, უნდა აღინიშნოს:

- ლექსიკური გარჩევის ახალი წესები;
- ახალი ელემენტები და ატრიბუტები (მათ შორის უპრიანია, ცალკე გამოვყოთ ახალი ტიპის input-ელემენტები);
- HTML 5-ში რეგლამენტირებულია JavaScript-თან მუშაობა, რაც ხორციელდება დოკუმენტის ობიექტურ მოდელზე (DOM) დაყრდნობით.

ლექსიკური გარჩევის ახალი წესები

HTML 5 ენის დაპროექტებისას კოდის გარჩევისადმი გამოყენებული იქნა თვისებრივად ახლებური მიდგომა – ლექსიკური გარჩევის ახალ წესებზე დაყრდნობით HTML კოდთან მომუშავე ბროუზერები გაცილებით მოქნილად ახდენენ შეცდომების გამოვლენა-დამუშავებას. შედეგად, იმ შემთხვევაშიც კი, როდესაც HTML-ფაილის კოდი სინტაქსურად არცთუ მთლად გამართულად (კორექტულად) არის დაწერილი, კოდის ლექსიკური გარჩევა იმგვარად ხორციელდება, რომ სხვადასხვა ბროუზერი ერთნაირ სურათს იძლევა ეკრანზე.

ახალი ელემენტები და ატრიბუტები

თავდაპირველად აღვნიშნოთ, რომ შეთავსებადობის უზრუნველყოფის მიზნით, HTML 5-ში შესაძლებელია HTML 4-ის ყველა ელემენტის გამოყენება. თუმცა რეკომენდებული არ არის ზოგიერთი, მოძველებულად მიჩნეული ელემენტების კოდში ჩართვა, მაგალითად, ისეთი ელემენტებისა, რომლებიც განკუთვნილია ტექსტის ფრაგმენტების ადგილზე გადასა-ფორმატებლად: <b>, <i>, <u>, <sup>, <sub>, <font>, ასევე სხვა ელემენტებისაც (, , , strike), რომელთა ფუნქციების

შესრულება ასევე შესაძლებელია უფრო ეფექტიანი საშუალებებით განხორციელდეს.

აქვე შევნიშნავთ, რომ ზოგიერთმა ელემენტმა გარკვეული ცვლილებები განიცადა (<a>, <menu> და <cite>).

ქვემოთ მოგვყავს იმ ახალი ელემენტების სია, რომლების შემოტანითაც, ფაქტობრივად, ახალი პლატფორმა ჩამოყალიბდა WEB-გამოყენებების შესაქმნელად.

ესენია:

<video>, <audio>, <canvas>, SVG (Scalable Vector Graphics) – მასშტაბირებადი ვექტორული გრაფიკა, <section>, <article>, <header>, <footer>, <nav>, <progress>, <meter>, <time>.

ჩამოთვლილი ელემენტები WEB-საიტებისათვის უზრუნველყოფენ ახალ ფუნქციონალურობას. შედეგად შესაძლებელი ხდება WEB-ფურცლებზე უშუალოდ – საკუთარი პლაგინების გამოყენების და API-სადმი მიმართვების გარეშე – განხორციელდეს გრაფიკული და მულტიმედიური ობიექტების შემოტანა და მართვა.

გარდა აღნიშნულისა, ზოგიერთი ელემენტი განკუთვნილია WEB-რესურსების სტრუქტურის უკეთ წარმოჩენისათვის. მათი გამოყენების შედეგად WEB-ფურცლები (WEB-საიტები) გაცილებით უფრო ადვილად აღქმადი ხდება. ამასთან, საძიებო სისტემებსა და სხვა წამკითხველ პროგრამებს ეძლევათ საჭირო ინფორმაციის მოძიების დამატებითი საშუალებაც. ასეთებია, მაგალითად, <nav> (სანავიგაციო პანელი) და <footer> ელემენტები, რომლებიც ტექნიკურად ეკვივალენტურია ჩვენთვის ცნობილი და ელემენტების, მაგრამ მათ დამატებით სწორედ ეს აღნიშნული შესაძლებლობებიც ახასიათებს.

სტრუქტურული ელემენტების გარდა, HTML 5 გვთავაზობს ბლოკის დონის სემანტიკური ელემენტების კრებულსაც:

<aside>, <figure>, <dialog>.

მაგალითად, <aside> ელემენტს იყენებენ რჩევების, შენიშვნების და ციტატების, ანუ არაძირითადი ტექსტის ასახვისათვის.

ენის კიდევ ერთი სიახლე გახლავთ შემდეგი გლობალური ატრიბუტების შემოტანა:

id, tabindex, repeat.

შენიშვნა: *ჩამოთვლილი ელემენტების და ატრიბუტების, ასევე სხვა სიახლეების დანიშნულებას უფრო დაწვრილებით მომდევნო პარაგრაფებში გავეცნობით.*

JavaScript-თან რეგლამენტირებული მუშაობა, დოკუმენტის ობიექტურ მოდელზე (DOM) დაყრდნობით

HTML 5 ენაში სიახლეები არის შემოტანილი გამოყენებითი დაპროგრამების ინტერფეისის (API) უზრუნველყოფისა და DOM მოდელის გაფართოების გზითაც. მიუხედავად იმისა, რომ დამპროექტებელთა ზემოთ აღნიშნული ორი ჯგუფის მიერ რიგ შემთხვევაში საბოლოო გადაწყვეტილება ჯერ კიდევ მიღებული არ არის, ბევრი შემოტანილი შესაძლებლობის გამოყენება უკვე წარმატებით ხორციელდება, თუმცა სპეციალისტების მიერ რეკომენდებულია საიტის შექმნისას მოხდეს მიღებული შედეგების შემოწმება რამდენიმე ყველაზე პოპულარული ბროუზერის მეშვეობით მაინც.

დასასრულ, მოგვყავს html კოდში ახალი ელემენტების ჩართვის მაგალითები:

```
<!DOCTYPE html>
<html>
 <head>
   <title>HTML5-ზე WEB-ფურცლის შექმნის მაგალითი</title>
 </head>

 <body>
  <header>
     <h1>სათაური-კოლონტიტული</h1>
  </header>
  <section>
   <article>
```

```
    <h1>I სექციის სათაური</h1>
      <p>დაიწყო აბზაცი.....</p>
      <p>დაიწყო აბზაცი.....</p>
      <p>დაიწყო აბზაცი.....</p>

  </article>

  <article>
   <h1>II სექციის სათაური</h1>
      <p>დაიწყო აბზაცი.....</p>
      <p>დაიწყო აბზაცი.....</p>
      <p>დაიწყო აბზაცი.....</p>
  </article>

  <nav>
    <a href="http://www.rambler.ru">www.rambler.ru</a>
    <a href="http://www.worldsport.ge">www.worldsport.ge</a>
    <a href="http://ru.wikipedia.org/wiki/HTML5">wikipedia HTML5</a>
  </nav>

  </section>

  <footer>
     <h1>ფუძე-კოლონტიტული</h1>
  </footer>

 </body>
</html>
```

შენიშვნა: *ის, რაც HTML4-ში მიიღწეოდა div ელემენტების მეშვეობით (სხვადასხვა უბნების დანიშნულების განსაზღვრა), აქ ხორციელდება სპეციალური სტრუქტურიზებული უბნების – სექციების მეშვეობით, რომელთა არსი და დანიშნულება HTML 5-ისათვის უკვე ცნობილია, და ამრიგად, საჭირო აღარ არის ამ უბნების სპეციფირება სხვადასხვა კლასად div ელემენტების class ატრიბუტების მეშვეობით. თუმცა უნდა აღინიშნოს, რომ WEB-დოკუმენტში შესაბამის ადგილზე პანელის განსათავსებლად და მისთვის საჭირო სახის მისაცემად CSS-ის შესაძლებლობები აქაც გამოიყენება.*

მოვიყვანოთ კიდევ ერთი მაგალითი:

```
<!doctype html>
<html>

 <head>
     <meta charset=utf-8>
     <title>
        HTML5-ზე WEB-ფურცლის შექმნის კიდევ ერთი  მაგალითი
     </title>
 </head>

 <body>
  <header>

    <hgroup>
      <h1>"h1" სათაური hgroup-იდან</h1>
      <h2>"h2" სათაური hgroup-იდან </h2>
    </hgroup>

  </header>

  <nav>
    <a href=link1.html> "nav" ბლოკიდან პირველი დაყრდნობა </a>
    <a href=link2.html>"nav" ბლოკიდან მეორე დაყრდნობა </a>
  </nav>

  <section>

    <article>

      <h1>"article" ბლოკიდან სტატიის სათაური</h1>
      <p> "article" ბლოკიდან სტატიის აბზაცის ტექსტი</p>

        <details>
          <summary>"details" ბლოკი, "summary" ტეგის ტექსტი</summary>
          <p>"details" ბლოკიდან აბზაცი</p>
        </details>

    </article>

  </section>

  <footer>
    <time>"footer" ბლოკიდან "time" ტეგის შემცველობა</time>
    <p>"footer"  ბლოკიდან აბზაცი </p>
```

```
    </footer>

  </body>
</html>
```

ვიწყებთ ზოგადად HTML ენის შესაძლებლობების შედარებით მოკლედ, ხოლო მისი ბოლო HTML 5 ვერსიის უფრო დეტალურად შესწავლას:

ინტერნეტში ინფორმაციის განთავსებისთვის საჭირო გახდა პრინციპულად ახალი საშუალებების შემუშავება. მსოფლიო მასშტაბის ამ გლობალურმა ქსელმა არა მარტო დააკავშირა ერთმანეთთან ადამიანები დედამიწის ყველა კუთხიდან, არამედ აქცია ისინი ე. წ. ინტერნეტ-ტექნოლოგიების მომხმარებლებად. საინტერესოა, რომ აღნიშნული ტექნოლოგიები უფრო ფართო გამოყენების ობიექტადაც კი იქცა, ვიდრე მათი უშუალო დანიშნულება გახლავთ. მაგალითად, შეგვიძლია ინტერნეტში ჩართვის გარეშეც გამოვიყვანოთ ბროუზერში, ვთქვათ, კომპაქტ-დისკოზე ჩაწერილი ინფორმაცია.

ინტერნეტ-ტექნოლოგიების შექმნისათვის ბაზისური საშუალება – საფუძველი – გახლავთ HTML ენა. ბუნებრივია, ამ ტექნოლოგიების შესწავლის მსურველმა პირველი ნაბიჯები სწორედ ამ მიმართულებით გადადგას.

HTML-ს **(HyperText Markup Language)** დარგის სპეციალისტები უწოდებენ ჰიპერტექსტის მონიშვნა-გაწყობისათვის გამიზნულ ენას, ხოლო თვით **ჰიპერტექსტს** კი განმარტავენ, როგორც ჩვეულებრივ ტექსტზე უფრო მეტი ინფორმაციული და ფუნქციური მონაცემების შემცველ დოკუმენტს. პირველ ყოვლისა, ეს გულისხმობს, რომ ჰიპერტექსტი წარმოადგენს განშტოებად ტექსტს, რომელზედაც შესაძლებელია შესრულდეს ჩვენი მოთხოვნების შესატყვისი ესა თუ ის მოქმედებანი (ტედ ნელსონი, 1965). აღნიშნული მიდგომა კი ხორციელდება ტექსტის ცალკეული ფრაგმენტების მონიშვნისა და დოკუმენტის სხვადასხვა უბნიდან მათზე გადასვლების შესაძლებლობის უზრუნველყოფის მეშვეობით. აღწერილმა

მიდგომამ ისეთი პოპულარობა მოიპოვა, რომ ერთი დოკუმენტის ფარგლებში მოქმედი შესაძლებლობიდან იგი შემდგომ გადაიზარდა მთელი ინტერნეტის მასშტაბით რეალიზებულ URL დამისამართების სისტემასა და მონაცემების გადაცემის HTTP ოქმში.

ტექსტის ჰიპერტექსტად გადაქცევა ხორციელდება არა მხოლოდ მისი ერთი ფრაგმენტიდან მეორეზე გადასასვლელად, რაც თავისთავად ჰიპერტექტისათვის დამახასიათებელ ძალიან მნიშვნელოვან ნიშან-თვისებად იქცა (მით უფრო იმ გარემოების გათვალისწინებით, რომ ეს მექანიზმი მთელი მსოფლიო აბლაბუდის ფარგლებში მუშაობს), არამედ – ტექსტისათვის უფრო მეტი გამომსახველობითი და სტრუქტურული სახის მისაცემადაც.

ქვემოთ გავიხსენოთ HTML 4 ენიდან ნაცნობი რამდენიმე მნიშვნელოვანი ელემენტი. ამასთან, გავეცნოთ მათდამი წაყენებულ ზოგიერთ ახალ მოთხოვნასაც:

ელემენტი

ტექსტის აბზაცებად დასაყოფად გამოიყენება ელემენტი. კიდევ ერთხელ გავუსვამთ ხაზს – HTML 5 ენა მოითხოვს იგი (და ნებისმიერი სხვა წყვილტეგიანი ელემენტი) დაბოლოვდეს ჩამკეტი ტეგით:

 მე ვარ აბზაცი

 ელემენტს აქვს ერთადერთი ატრიბუტი (პარამეტრი) align, რომელსაც ეკრანზე ტექსტის პოზიციის განსაზღვრისათვის შეუძლია მიიღოს შემდეგი მნიშვნელობები:

left (დუმილით)	სწორება მარცხნივ
right	სწორება მარჯვნივ
center	სწორება ცენტრზე
justify	სწორება სიგანეზე

 ელემენტი ბლოკური ელემენტია – იგი იკავებს მისთვის გამოყოფილი სტრიქონისა თუ უბნის მთელ სიგანეს იმ შემთხვევაშიც კი,

როცა მისი შემცველობა ბოლომდე არ ავსებს მას, ხოლო ბლოკის სიმაღლე განისაზღვრება ამ შემცველობის სიმაღლით.

 ელემენტი

ხშირად მოითხოვება ტექსტის შემდეგ სტრიქონზე გადატანა მოხდეს ახალი აბზაცის გარეშე (მაგალითად, ეკრანზე ლექსის გამოტანისას) – ასეთ შემთხვევაში იყენებენ ერთტეგიან
 ელემენტს:

```
<p> ქარი ქრის, ქარი ქრის, ქარი ქრის,<br>
    ფოთლები მიქრიან ქარდაქარ,<br>
    ხეთა რიგს, ხეთა ჯარს რკალად ხრის,<br>
    სადა ხარ, სადა ხარ, სადა ხარ...<br>
</p>
```

შენიშვნა: <p> *და*
 ელემენტებს შორის მსგავსება-განსხვავების დეტალებს უფრო დაწვრილებით ლაბორატორიული სამუშაოების შესრულებისას შევისწავლით, გავეცნობით აგრეთვე სხვა ელემენტებისათვის დამახასიათებელ ზოგიერთ ნიუანსაც.

სათაურები

ბლოკური ელემენტების ნაირსახეობას წარმოადგენს ერთმანეთისაგან შრიფტის ზომის მიხედვით განსხვავებული 6 დონის სათაურებიც (დალაგებულია კლების მიხედვით):

```
<H1> პირველი დონის სათაური </H1>
<H2> მეორე დონის სათაური  </H2>
.....
.....
.....
<H6> მეექვსე დონის სათაური  </H6>
```

სიები

ობიექტების ჩამოსათვლელად HTML იყენებს დანომრილ – და დაუნომრავ – სიებს, ცალკეული პუნქტებისათვის კი – ელემენტს:

```
<ul>                                (unordered list)
    <lh> უნომრო სია </lh>
    <li> სიის პუნქტი </li>
    <li> სიის პუნქტი </li>
    <li> სიის პუნქტი </li>
</ul>
<ol type="I">                       (ordered list)
    <lh> დანომრილი სია </lh>
    <li> პირველი პუნქტი </li>
    <li> მეორე პუნქტი   </li>
    <li> მესამე პუნქტი   </li>
 </ol>
```

ol დანომრილი სიის საწყის ტეგში type ატრიბუტმა შეიძლება მიიღოს მნიშვნელობები:

ატრიბუტები	*ნუმერაციის სახე*
type="1"	1, 2, 3, 4 . . .
type="i"	i, ii, iii, iv, . . .
type="I"	I, II, III, IV, . . .
type="a"	a, b, c, d, . . .
type="A"	A, B, C, D, . . .

ამასთან, სასტარტო სიდიდე შეიძლება ერთისგან განსხვავდებოდეს. მას განსაზღვრავს start ატრიბუტი. მაგალითად:

```
start=15;
```

საინტერესოა, რომ value ატრიბუტით უშუალოდაც შეიძლება განვსაზღვროთ სიის li ელემენტის ნომერი.

რაც შეეხება ul უნომრო სიებს, მისი პუნქტებისათვის დასაშვებია მარკერის სახის საკუთარი შეხედულებისამებრ არჩევა:

circle, square, disc, none

ამ მიზნით უნომრო სიის საწყის ტეგში type ატრიბუტს ვაძლევთ შესაბამის მნიშვნელობას. მაგალითად:

```
<ul  type = “square”>
  <li> პირველი პუნქტი </li>
  <li> მეორე პუნქტი    </li>
  <li> მესამე პუნქტი    </li>
</ul>
```

არსებობს სპეციალური ტიპის სიებიც, რომლებიც იქმნება შემდეგი შაბლონის მეშვეობით:

```
<dl>                                                (definition list)
    <dt>პირველი პუნქტი</dt>
        <dd>ვაშლი</dd>
        <dd>მსხალი</dd>
    <dt>მეორე პუნქტი</dt>
        <dd>სტაფილო</dd>
    <dt>მესამე პუნქტი</dt>
        <dd>არაჟანი</dd>
</dl>
```

ზემოთ მოყვანილი <dl> სახის სიებს ძირითადად იყენებენ ცნებების ჩამონათვალისა და მათი განმარტებების (განსაზღვრებების) ფორმირებისათვის. ასეთ შემთხვევაში თითოეულ <dt> პუნქტს მოსდევს ერთადერთი <dd> პუნქტი:

```
<dl>
  <dt> №1 ტერმინი </dt>
     <dd> №1 ტერმინის განმარტება </dd>
  <dt> №2 ტერმინი </dt>
```

```
    <dd> №2 ტერმინის განმარტება </dd>
  ----------------------------------------------------
</dl>
```

შევნიშნოთ, რომ განსაზღვრებების სიას იყენებენ ტექსტის მარჯვნივ გასაწევადაც.

აღსანიშნავია, რომ თითოეული უნომრო პუნქტი, თავის მხრივ, შეიძლება შეიცავდეს დანომრილი ქვეპუნქტების ნებისმიერ რიცხვს, ანუ შესაძლებელია შევქმნათ უნომრო სიაში ჩალაგებული ნომრიანი სიებიც (და პირიქითაც – ნომრიანში ჩალაგებული უნომრო), მაგალითად:

```
<ul>
  <li> პირველი პუნქტი
    <ol>
      <li> 1.1 ქვეპუნქტი </li>
      <li>1.2 ქვეპუნქტი </li>
    </ol>
  </li>
  <li> მეორე პუნქტი
   <ol>
      <li>2.1 ქვეპუნქტი </li>
      <li>2.2 ქვეპუნქტი </li>
      <li>2.3 ქვეპუნქტი </li>
   </ol>
  </li>
</ul>
```

დისპლეიზე გამოტანილი ტექსტის გამომსახველობის ასამაღლებლად HTML ენაში გამოიყენება მრავალი ელემენტი და მათთან დაკავშირებული ატრიბუტები, მაგრამ მათი მნიშვნელოვანი ნაწილის (მაგალითად, ტექსტის დაფორმატებისთვის განკუთვნილების - <font> elementi, <b>, <i>, <u>, <sup, <sub> და სხვ.) მიერ გენერირებული ეფექტების მიღწევა HTML 5 ენაში,

როგორც წესი, CSS ცხრილების გამოყენებით არის შესაძლებელი და რეკომენდებულიც.

ამასთან, ტექსტთან მუშაობისათვის შემოტანილია რიგი ახალი ელემენტებისა. ქვემოთ მოგვყავს მათგან ყველაზე მნიშვნელოვანთა ჩამონათვალი დანიშნულების აღწერით (შევნიშნავთ, რომ თითოეული მათგანი წყვილი ტეგისაგან შედგება):

- <section> – ამ ელემენტის შემცველობა (კონტენტი) ბროუზერის მიერ აღიქმება როგორც ბლოკი (სექცია).
- <header> – მისი მეშვეობით ფიქსირდება დოკუმენტის (სექციის) დასაწყისი.
- <footer> – ელემენტი კი აფიქსირებს დოკუმენტის (სექციის) დაბოლოებას.
- <hgroup> – აჯგუფებს სათაურებს.
- <time> – გამოიყენება თარიღისა და დროის ჩვენებისათვის.
- <nav> – ქმნის სანავიგაციო მენიუს (აფორმირებს დაყრდნობების დაჯგუფებას).
- <mark> – ახდენს ტექსტის განსაკუთრებით მნიშვნელოვანი ნაწილის მარკირებას (გამოყოფას)..
- <aside> – ამ ელემენტში, როგორც წესი, განათავსებენ დოკუმენტში მოყვანილი ძირითადი ტექსტისგან გვერდზე მდგომი რაიმე, დამატებითი სახის ინფორმაციის გასაცნობად განკუთვნილ დაყრდნობებსა და ჭდეებს.
- <article> – მოცემული ელემენტიც <section> ელემენტის მსგავსად ახორციელებს დოკუმენტის კონტენტის სექციონირებას. მისი თავისებურება კი ის გახლავთ, რომ ელემენტის შემცველობა შესაძლებელია წარმოადგენდეს დამოუკიდებლად გავრცელებისათვის გამიზნულ ინფორმაციას: ფრაგმენტს ფორუმიდან, სტატიას, ბლოგის ჩანაწერს და სხვ.

აღსანიშნავია, რომ ეს სიახლეები განკუთვნილია არა იმდენად ტექსტურ ფრაგმენტებზე ამა თუ იმ გარეგნული ეფექტის მოსახდენად,

არამედ – ბროუზერისა (და მომხმარებლის) ინფორმირებისათვის, თუ რა სახის ინფორმაცია არის მათში განთავსებული და როგორ შეიძლება შემდგომში მისი დამუშავება.

ელემენტების პარამეტრებს (ატრიბუტებს) HTML 4 ენის აღწერიდან უკვე ვიცნობთ. HTML 5-ში შემოტანილია ახალი ცნებაც: **გლობალური ატრიბუტი**. ასეთი სახის ატრიბუტი დაკავშირებულია ელემენტების უმეტესობასთან. უფრო დაწვრილებით მათ სახეობებს და გამოყენების წესებს მოგვიანებით შევისწავლით.

<img> ელემენტი

<img> გრაფიკული ელემენტი წარმოადგენს ერთტეგიან ჩადგმულ ელემენტს, რაც ნიშნავს, რომ იგი აბზაცისგან დამოუკიდებლად არ გამოიყენება. უნდა აღინიშნოს, რომ Web-ფურცელზე ყველანაირი გამოსახულება ვერ განთავსდება.

საერთოდ, რასტრული გამოსახულება ინახება jpg, gif, bmp, tiff, png, psd ტიპის, ხოლო ვექტორული – swf, cdr, max, ai ტიპის ფაილებში. მათგან Web-დანართებში იყენებენ JPEG, GIF, PNG რასტრული და SWF ვექტორული ტიპის გამოსახულებებს. შევნიშნავთ, რომ რასტრული ტიპის სამივე ფორმატში დასაშვებია მონაცემების შეკუმშვა.

<img> ელემენტი იყენებს აუცილებელ src ატრიბუტს (გრაფიკული ფაილის მისათითებლად) და შემდეგ არააუცილებელ ატრიბუტებსაც:

- align – განსაზღვრავს ეკრანზე გამოსახულების პოზიციას,
- alt – თუ გამოსახულება ეკრანზე ჯერ არ ჩატვირთულა, ეკრანზე გამოდის ამ ატრიბუტში მითითებული ტექსტი,
- border – განსაზღვრავს გამოსახულების ირგვლივ ჩარჩოს სისქეს;
- height – უჩვენებს გამოსახულების სიმაღლეს;
- hspace – განსაზღვრავს უახლოეს კონტენტამდე ჰორიზონტალური დაძვრის სიდიდეს,
- ismap – მიუთითებს, რომ მოცემული გამოსახულება წარმოადგენს სხვადასხვა უბნებზე გადასასვლელ რუკას,

- longdesc – გვიჩვენებს იმ დოკუმენტის მისამართს, რომელიც შეიცავს გამოსახულების ანოტაციას,
- vspace – განსაზღვრავს უახლოეს კონტენტამდე ვერტიკალური დაძვრის სიდიდეს,
- width – უჩვენებს გამოსახულების სიგანეს,
- usemap – განსაზღვრავს დაყრდნობას <map> ელემენტზე, რომელშიც მითითებულია მოცემული გამოსახულების შემადგენელი ნაწილების (ანუ რუკაზე "ქვეყნების") კოორდინატები.

მოვიყვანოთ გამოყენების მაგალითი:

```
<p><img src="sample.jpg"></p>
```

ახალი მულტიმედიური ელემენტები

გავეცნოთ HTML5-ში შემოტანილ მნიშვნელოვან სიახლეებს მულტიმედიური ხაზით.

ესენია ელემენტები:

<audio> და <video>.

პირველ ყოვლისა, უნდა აღინიშნოს, რომ ორივე ელემენტი თანამედროვე ბროუზერებისათვის "გარე სხეულებს" არ წარმოადგენს (მათი საკუთარი კომპონენტები), რაც მნიშვნელოვნად აადვილებს ამ ელემენტების გამოყენებას, ამაღლებს საიმედოობას, ამასთან, შესაძლებელი ხდება Web-სცენარიდან მათი მართვაც. თუმცა ისიც უნდა ითქვას, რომ დღეისდღეობით რიგი კოდეკებისა ვერ "ეწყობა" HTM L 5-ის მოთხოვნებს (HTML5-ის სპეციფიკაციაში, საერთოდ, მითითებული არცაა ამ შესაძლებლობების მხარდამჭერი კოდეკები).

აღნიშნულ პრობლემას ნაწილობრივ წყვეტს <source> ელემენტის გამოყენება, რომელშიც ხდება რამდენიმე მულტიმედიური წყაროს მითითება. ბროუზერს საშუალება ეძლევა მათ შორის ყველაზე შესაფერისი ამოირჩიოს

მოვიყვანოთ მაგალითი:

```
<audio>
  <source src="sound1.ogg">
  <source src="sound1.mp3">
</audio>
```

ხოლო ამ ელემენტის ჩასაყენებლად საბაზისო კოდს ასეთი მარტივი სახე შეიძლება ჰქონდეს:

```
<audio src="sound1.mp3"> </audio>
```

შენიშვნა: იმ შემთხვევაში, როდესაც მოცემული ბროუზერი <audio> *ელემენტს ვერ იყენებს, შესაძლებელია ეკრანზე გამოვიტანოთ ამ ელემენტში შეტანილი შესატყვისი ინფორმაციის ტექსტი.*

<audio> ელემენტისათვის დასაშვებია შემდეგი დანიშნულების ატრიბუტების გამოყენება:

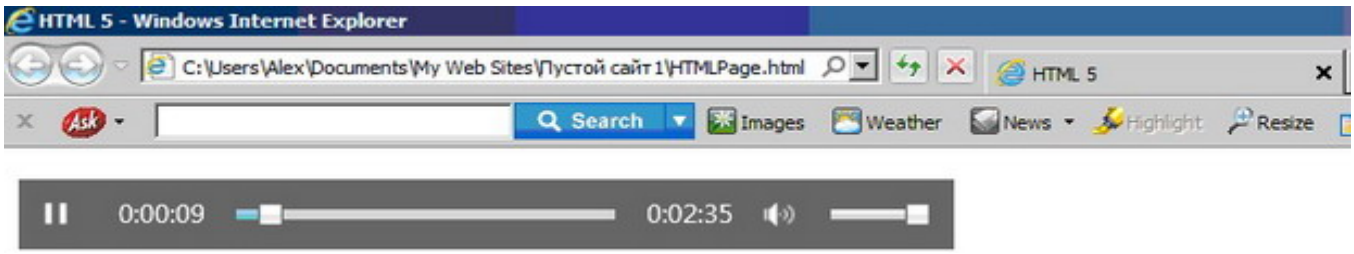
- autoplay – ბგერითი ფაილის გაშვება ხდება Web-ფურცლის ჩატვირთვისთანავე;
- controls – აუდიორგოლის გასაშვებ უბანს ეკრანზე ემატება მართვის პანელიც;
- loop – დამთავრების შემდეგ ხდება აუდიორგოლის განმეორებითი გაშვება;
- preload – ამ ატრიბუტის მეშვეობით Web-ფურცელთან ერთად ჩაიტვირთება მოცემული ფაილიც, მაგრამ, თუ ამავე დროს autoplay ატრიბუტიც გამოიყენება, ხდება preload ატრიბუტის იგნორირება;
- src – მიუთითებს წყაროს, საიდანაც უნდა ჩაიტვირთოს ფაილი.

მოვიყვანოთ მაგალითი:

```
<audio autoplay controls src="1.mp3">
  ვერ ხერხდება <audio> ელემენტის გაშვება!
</audio>
```

ქვემოთ ასახულია დისპლეიზე შექმნილი სიტუაციები ორივე – ელემენტის გამოყენება-ვერ გამოყენების შემთხვევისათვის:

ა)



ბ)

HTML კოდში <video> ელემენტის დამატებაც ანალოგიური წესით ხდება:

```
<video src="video1.avi"> </video>
```

<video> ელემენტისათვის დასაშვებია შემდეგი სახელწოდების და დანიშნულების ატრიბუტების გამოყენება:

- autoplay – ვიდეოფაილის გაშვება ხდება Web-ფურცლის ჩატვირთვისთანავე;
- controls – ვიდეოს ემატება მართვის პანელიც;
- height – განსაზღვრავს ვიდეოს ჩვენებისათვის განკუთვნილი უბნის სიმაღლეს;
- loop – ჩვენების დასრულებისთანავე ხდება ვიდეოიფაილის ხელახლა გაშვება;
- poster – უჩვენებს იმ გამოსახულების წყაროს, რომელიც ეკრანზე აისახება მაშინ, როდესაც ვიდეოფაილის ჩვენებისას ამა თუ იმ მიზეზით პაუზაა.
- preload – ამ ატრიბუტის მეშვეობით Web-ფურცელთან ერთად ჩაიტვირთება მოცემული აუდიოფაილიც, მაგრამ თუ ამავე დროს autoplay ატრიბუტიც გამოიყენება, ხდება preload ატრიბუტის იგნორირება;
- src – მიუთითებს წყაროს, საიდანაც უნდა ჩაიტვირთოს ფაილი,

- width – განსაზღვრავს ვიდეოს ჩვენებისათვის განკუთვნილი უბნის სიგანეს.

მართალია, დღესდღეობით, სპეციალისტების აღიარებით, ზემოთ განხილული ახალი მულტიმედიური <audio> და <video> ელემენტების შესაძლებლობები შეზღუდულია, მაგალითად, მათი მეშვეობით ვერ ხერხდება პროვაიდერისაგან ნაკადური მულტიმედია-მონაცემების მიღება (პირველ ყოვლისა, აქ ტელეგადაცემები იგულისხმება), მაგრამ, ამ სპეციალისტებისავე მტკიცებით, მდგომარეობა მალე გამოსწორდება და მომავალი სწორედ ამ მულტიმედიურ ელემენტებს ეკუთვნის.

ცხრილები

ცხრილებსა და მათი აგების წესებს დაწვრილებით გავეცანით HTML 4 ენის შესწავლისას. ცხრილებთან მიმართებით HTML 5 ენაში რაიმე განსაკუთრებულ სიახლეს ვერ ვხვდებით, საკმარისია მხოლოდ მათთვისაც დავიცვათ ელემენტებისადმი წაყენებული ადრე განხილული ზოგადი მოთხოვნები (მაგალითად, ელემენტების დახურვის მოთხოვნა და სხვ.).

ზემოთ თქმულიდან გამომდინარე, ქვემოთ ჯერ ჩამოვთვლით <table> ელემენტისათვის დამახასიათებელ ატრიბუტებს (მათი დანიშნულების აღწერით), მოვიყვანთ აგრეთვე მარტივი და შედარებით რთული ცხრილების აგების მაგალითებს მათი გრაფიკული სახის და შესაბამისი კოდების მოყვანით:

- align – განსაზღვრავს ცხრილის გასწორების ტიპს;
- background – მიუთითებს ფონურ გამოსახულებაზე;
- border – უჩვენებს პიქსელებით მოცემულ ცხრილის ჩარჩოს სისქეს;
- bordercolor – განსაზღვრავს ამ ჩარჩოს ფერს;
- cellpadding – იძლევა ჩარჩოდან უჯრედის შემცველობამდე დაძვრის სიდიდეს;
- cellspacing – განსაზღვრავს უჯრედებს შორის მანძილს;

- cols – განსაზღვრავს ცხრილში სვეტების რაოდენობას;
- frame – მიუთითებს ცხრილის გარშემო საზღვრების არსებობაზე. შეუძლია მიიღოს შემდეგი მნიშვნელობანი:
    - void – არ აისახოს საზღვრები ცხრილის გარშემო;
    - border –აისახოს ცხრილის გარშემო საზღვრები;
    - above – საზღვარი გაივლოს ცხრილის მხოლოდ თავზე;
    - below – საზღვარი გაივლოს ცხრილის მხოლოდ ბოლოში;
    - hsides – აისახოს მხოლოდ ჰორიზონტალური საზღვრები;
    - vsides – აისახოს მხოლოდ ვერტიკალური საზღვრები;
    - rhs – საზღვარი გაივლოს ცხრილის მხოლოდ მარჯვენა მხარეს;
    - lhs – საზღვარი გაივლოს ცხრილის მხოლოდ მარცხენა მხარეს;
- height – განსაზღვრავს ცხრილს სიმაღლეს;
- rules – უჯრედები განცალკევდება შიგა საზღვრებით, შეუძლია მიიღოს შემდეგი მნიშვნელობანი:
    - all – ეს მოხდეს თითოეული უჯრედისათვის;
    - groups – ეს მოხდეს <thead>, <tfoot>, <tbody>, <colgroup>, <col> ელემენტებით განსაზღვრული ჯგუფებისათვის;
    - cols – საზღვრები აისახოს ცხრილის სვეტებს შორის;
    - none – დაიმალოს ყველა საზღვარი;
    - rows – საზღვრები აისახოს ცხრილის სტრიქონებს შორის;
- summary – ცხრილი დანიშნულების მოკლე აღწერა;
- width – განსაზღვრავს ცხრილის სიგანეს.

სანამ ცხრილების აგებაზე გადავიდოდეთ, შევნიშნოთ, რომ მათ, გარდა თავისი პირდაპირი დანიშნულებისა, დამატებით ეკისრება დიზაინერის ფუნქციებიც. ცხრილების მეშვეობით შეიძლება მწყობრში მოვაქციოთ ფურცლის ნაწილები, გვერდიგვერდ განვათავსოთ ტექსტი და

ნახატი, შევქმნათ ფერადოვანი ხალიჩა და სხვ. ცხრილის ელემენტების ჩამოთვლა ხდება ზემოდან ქვემოთ და მარჯვნიდან მარცხნივ.

დავხაზოთ მარტივი ცხრილი, შემდეგ კი დავწეროთ მისთვის პროგრამული კოდი:

მარტივი ცხრილი

სათაური 1	**სათაური 2**
სტრიქონი 1. უჯრა 1.	სტრიქონი 1. უჯრა 2.
სტრიქონი 2. უჯრა 1.	სტრიქონი 2. უჯრა 2.

```
<table border=1 cellspacing=3>
    <caption> martivi cxrili </caption>
  <tr>
    <th bgcolor = “yellow”> saTauri 1</th>
   <th bgcolor = “yellow”> saTauri 2 </th>
  </tr>
 <tr>
    <td> striqoni 1. ujra 1.</td>
    <td> striqoni 1. ujra 2.</td>
  </tr>
  <tr>
    <td> striqoni 2. ujra 1.</td>
     <td> striqoni 2. ujra 2.</td>
  </tr>
</table>
```

ვხედავთ, რომ <table> ელემენტის შიგნით შესაძლებელია წარწერისათვის განკუთვნილი (<caption>), სტრიქონის მაფორმირებელი (<tr>), სათაურისა (<th>) და ჩვეულებრივი მონაცემების (<td>) შემცველი ელემენტების გამოყენება.

მათ გარდა, <table> ელემენტი შესაძლებელია მოიცავდეს <col> ელემენტს (განსაზღვრავს ცხრილში ერთი ან რამდენიმე სვეტის

მახასიათებლებს) და ცხრილის უბნებად მასტრუქტურიზებელ, სვეტებისა და სტრიქონების დამაჯგუფებელ ასეთ ელემენტებსაც:

- <colgroup> – განსაზღვრავს სტილს სვეტების ჯგუფისათვის;
- <tbody> – ინახავს ერთ ან რამდენიმე სტრიქონს. ცხრილში დასაშვებია მხოლოდ ერთი ასეთი ელემენტის არსებობა;
- <tfoot> – ეს ელემენტიც ინახავს ერთ ან რამდენიმე სტრიქონს, ოღონდ ისინი ცხრილის ქვედა ნაწილს იკავებენ;
- <thead> –ინახავს ერთ ან რამდენიმე სტრიქონს, რომლებიც განთავსდებიან ცხრილის ზედა ნაწილში.

მოვიყვანოთ ცხრილის მაგალითი, რომელშიც გამოყენებულია ზემოთ ჩამოთვლილი ელემენტებიც:

```
<table border = 1 cellspacing = 2>
   <caption>
     <b> <i> HTML5-ში ლექციების კურსის გეგმა.</i></b>
   </caption>
  <thead>
    <tr>
      <th> № </th>
      <th> ლექციის სახელწოდება </th>
      <th> ლექციაში განხილული საკითხები</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td align = center> 1 </td>
      <td> Web 2.0 – ვებ-სისტემების დაპროექტების ახალი მეთოდიკა </td>
      <td> "Web 2.0" ტერმინის წარმოშობა, კონცეფციის საფუძვლები,
        სიახლენი თანამედროვე საიტების საპროექტო
        გადაწყვეტილებებში.
      </td>
    </tr>
    <tr>
```

```
      <td align = center> 2 </td>
      <td> HTML-ის საფუძვლები. სტანდარტები, დონეები, ვერსიები.
      </td>
      <td> HTML5 - ენის განვითარების ბოლო ეტაპი. Microsoft-ის მიერ
           შემუშავებული, მასთან სამუშაო ინსტრუმენტარიუმი.
      </td>
   </tr>
   <tr>
      <td> ... </td>
      <td> ..... </td>
      <td> .....</td>
   </tr>
 </tbody>
 <tfoot>
   <tr>
     <th>ლექციების რაოდენობა</th>
     <th>სალექციო საათების რიცხვი</th>
     <th>ლაბორატორიული საათების რიცხვი</th>
   </tr>
   <tr align = center>
      <td>17</td>
      <td>17 საათი</td>
      <td>51 საათი</td>
    </tr>
  </tfoot>
</table>
```

<u>დავალება:</u>

გამოიყენეთ ზემოთ მოყვანილი კოდი, შეიტანეთ მასში ცვლილება-დამატებანი (გავიხსენოთ HTML4-ში ნასწავლი მასალა) ტექსტური უბნების ეკრანზე შესაბამისი სახით (აქ, პირველ ყოვლისა, იგულისხმება შრიფტი) ასახვის მიზნით.

აღვნიშნოთ, რომ შესაძლებელია კიდევ უფრო რთული სტრუქტურის ცხრილების შექმნაც, თუ უჯრებს გავაერთიანებთ როგორც ჰორიზონტალური ან ვერტიკალური, ისე ერთდროულად ორივე მიმართულებით. ამ მიზნით, <td> და <th> ტეგებში <colspan> და <rowspan> ატრიბუტებისათვის დუმილით გათვალისწინებულ ერთის ტოლ მნიშვნელობებს ცვლიან შესაბამისი სიდიდეებით.

<u>დავალება:</u>

დავწეროთ კოდი წარმოდგენილი ცხრილისათვის, შემდეგ შევადაროთ იგი ნახაზის ქვემოთ მოყვანილ კოდს:

	სათაური 1	
	სათაური 1.1	სათაური 1.2
სათაური 2	უჯრა 1	უჯრა 2
სათაური 3	უჯრა 3	უჯრა 4

```
<table  border = “4”  cellspacing=3 >
    <caption><i> ცხრილი გაერთიანებული უჯრებით </i></caption>
  <tr>
    <th  rowspan = “2”>  </th>
    <th  colspan = “2”>  სათაური 1</th>
  </tr>
  <tr>
    <th>სათაური 1.1</th>
    <th>სათაური 1.2</th>
  </tr>
  <tr>
    <th>სათაური 2</th>
    <td>უჯრა 1</td>
    <td>უჯრა 2</td>
  </tr>
```

```
<tr>
    <th>სათაური 3</th>
    <td>უჯრა 3</td>
    <td>უჯრა 4</td>
  </tr>
</table>
```

CSS – სტილების შემნახველი კასკადური ცხრილები

უკვე ვიცით, ეომ ეკრანზე გამოტანილი ინფორმაციისათვის უფრო მეტი გამომსახველობის მისაცემად, რომ HTML ენა იყენებს სხვადასხვა საშუალებას (font და style ელემენტებს და style ატრიბუტს,...). მათი შეტანა ხდებოდა და ხდება უშუალოდ დოკუმენტის აღმწერ კოდში, რაც მთლად მოსახერხებელი არ გახლავთ, განსაკუთრებით – დიდი მოცულობის მქონე საიტების შექმნისას.

არსებული სიტუაციის გამოსასწორებლად ვებ-დოკუმენტებისათვის (მათი გაფორმების გასაადვილებლად) სპეციალისტებმა შეიმუშავეს პრინციპულად ახალი მიდგომა – **CSS** (Cascading Style Sheets) – **სტილების შემნახველი კასკადური ცხრილები.**

CSS ერთდროულად სტანდარტიცაა და ენაც, რომელიც აფართოებს ტრადიციული HTML-ის შესაძლებლობებს. დღეს არსებობს მისი რამდენიმე სპეციფიკაცია: CSS1, CSS2 და დამუშავების სტადიაში მყოფი CSS3, რომლებიც გაცილებით მეტ ატრიბუტებს (*აქ მათ **თვისებები** ეწოდება*) შეიცავს, ვიდრე ეს უშუალოდ HTML-ში არის გათვალისწინებული. მაგრამ არანაკლებ მნიშვნელოვანია, რომ CSS-მა შესაძლებელი გახადა:

დოკუმენტების სიმრავლის ცენტრალიზებულად მართვა, ამასთან, შესაძლებელია ისინი ერთადერთი ცხრილიდანაც იმართოს;

ვებ-ფურცლების გარეგნული მხარის კონტროლის გამარტივება;

დიზაინერული გადაწყვეტების პროექტში ადვილად ჩართვა;

დოკუმენტის გამოტანისას წამკითხველი მოწყობილობის შესაძლებლობების ადვილად გათვალისწინება.

ქვემოთ გავეცნოთ CSS-კონცეფციის განვითარების ეტაპებსა და პერსპექტივებს:

ვერსია	მიღების თარიღი	შესაძლებლობანი
CSS1	17.01.1996	• ფურცელზე ელემენტის ასახვის წესის მართვა • ელემენტის და ტექსტის ურთიერთმიმართება • ელემენტის ზომების მართვა • შიდა და გარე დაძვრების მართვა • ცხრილებში ვერტიკალზე განლაგების მართვა • ელემენტის საზღვრების სტილის მართვა • სიების დაფორმატების მართვა • ტექსტის ფერის და ფონის მართვა • შრიფტის პარამეტრების მართვა • ტექსტის თვისებების მართვა • სტრიქონებს შორის ინტერვალების მართვა
CSS2	12.05.1998	*CSS1-ის შესაძლებლობანი და დამატებითი სიახლეები:* • ტექსტის მიმართულების მართვა • ელემენტის პოზიციონირების მართვა • უბნების ხილვადობის მართვა • საზღვრებიდან გასული ელემენტების მართვა • კურსორის გარეგნული სახის მართვა • ფენების მართვა • ელემენტის ზღვრული ზომების მართვა • ცხრილის უჯრედებს შორის დაცილებები • ელემენტის ცალკეული საზღვრების მართვა • ცხრილის ელემენტების ზომების მართვა • ბრჭყალების სტილის მართვა • ბეჭდვისას კონტენტის მართვა • ბგერის ხმამაღლობის, პაუზების მართვა

CSS2.1	8.09.2009	• მოხდა CSS2-ში დაშვებული შეცდომების გასწორება და ზოგიერთი მომენტის დაზუსტება, მათ შორის პერსპექტივის (CSS3 – იხ. ქვემოთ) გათვალისწინებით.
CSS3	დამუშავების სტადიაშია	• მომრგვალებული კუთხეების მხარდაჭერა • გრადიენტული საზღვრების მხარდაჭერა • ელემენტის ჩრდილების მართვა • არასტანდარტული შრიფტების გამოყენების შესაძლებლობა • მომხმარებლის ბლოკების ზომების მართვა • ტექსტის სვეტებად დაყოფა • და ზოგი სხვა გრაფიკული ეფექტი

დასასრულ, შევნიშნავთ, რომ CSS3 სტანდარტზე მუშაობა გრძელდება. ბოლო სიახლეების შესახებ ინფორმაცია შეგიძლიათ ინტერნეტში მოიძიოთ.

და კიდევ ცოტა რამ ისტორიიდან:

თავდაპირველად თითოეული ბროუზერი მასში ჩატვირთულ ვებ-დოკუმენტს (სტრუქტურას, შემცველობას) აღიქვამდა ამ დოკუმენტის შეთანადებით ბროუზერის დამპროექტებლების მიერ შემუშავებულ **ბროუზერის ობიექტურ მოდელთან**, რაც თითოეული მოდელის უნიკალობის გამო გარკვეულ სირთულეებს იწვევდა. ამის გამო იგი **შეცვლილი იქნა დოკუმენტის ობიექტური მოდელით** (Document Object Model – DOM). ეს მიდგომაც დოკუმენტს იერარქიული (ხისებრი) სტრუქტურის სახით წარმოგვიდგენს, მაგრამ შედეგი უკვე თითქმის აღარაა დამოკიდებული ბროუზერის "ვინაობაზე".

დოკუმენტის ობიექტური მოდელით წარმოდგენილი სტრუქტურა გულისხმობს დოკუმენტის შემცველობის კვალიფიცირებას **მშობელი, შვილობილი, თავისივე დონის, წინაპარი** და **შთამომავალი** ელემენტების

სახით (ორი უკანასკნელი მიგვითითებს ელემენტებს შორის ხეზე რამდენიმე დონით დაშორებაზე).

სტილების შემნახველი კასკადური ცხრილების შექმნა

სანამ ასეთი ცხრილის შექმნის საკითხებს განვიხილავდეთ, აღვნიშნავთ, რომ ამ მიზნის განმახორციელებელი კოდი შესაძლებელია და **უმჯობესი** გაფორმდეს ძირითადი, დასამუშავებელი HTML ფაილიდან ცალკე – css გაფართოების მქონე ფაილად (ზემოთ აღწერილი მოსაზრებებიდან გამომდინარე). მაგრამ პროგრამისტები არცთუ იშვიათად სტილების სახის განმსაზღვრელ კოდს ადგილს უთმობენ იმავე HTML ფაილში. მაშინ საქმე გვაქვს **სტილების შემნახველ შიგა ცხრილთან**.

<u>შენიშვნა:</u> HTML მოითხოვს ნებისმიერ შემთხვევაში სისტემას ეცნობოს კასკადური ცხრილების გამოყენების შესახებ, რისთვისაც head *უბანში უნდა ჩაირთოს შემდეგი მეტა-განსაზღვრება*

<meta http-equiv = “content-style-type” content = “text/css”>

შედეგად ბროუზერი მიიღებს შესაბამის ინფორმაცია. – რომელი ენა გამოიყენება.

თავდაპირველად განვიხილოთ სტილისათვის შიდა ცხრილის ფორმირების საკითხი. აქ მიდგომა – შიგა ცხრილის ფორმირების ხერხი (და სინტაქსიც) – შესაძლებელია სხვადასხვა სახის იყოს:

1. ცხრილი მოიცემა <head> უბანში განთავსებულ <style> ელემენტში:

```
<head>
   .............
   <style type = "text/css">
      p { color: red}
      ..............
   </style>
</head>
```

2. ცხრილი განისაზღვრება ცალკეული ელემენტის განლაგების ადგილზევე – მის საწყის ტეგში style ატრიბუტის მეშვეობით:

```
<p style="color: red"> ..... </p>
```

არსებობს უფრო რთული, თანამედროვე ვარიანტიც – კოდის დაწერისას ობიექტ-ორიენტირებული მიდგომის გამოყენება.

მოვიყვანოთ შესატყვისი მაგალითი:

```
<style  type = “text/css”>
    H1.red1 {color: RGB (215,40,40); text-align: center}
</style>
```

კოდის ფრაგმენტების სინტაქსზე ქვემოთ გვექნება საუბარი, ამჯერად კი აღვნიშნოთ “შიგა მიდგომის” დადებითი და უარყოფითი მხარეები:

- დადებითი მომენტი ის გახლავთ, რომ HTML ფაილი ეწევა “თვითმომსახურებას” – მას აღარ ესაჭიროება დამხმარე CSS ფაილის გამოძახება;
- უარყოფითი – შიგა სტილი მხოლოდ ერთი ფაილის ფარგლებში მოქმედებს და თუ იგი რამდენიმე ფაილშია კოპირებული, საჭიროების შემთხვევაში კოდის კორექტირება, ბუნებრივია, მეტ ძალისხმევას მოითხოვს; ასევე, არ სრულდება თანამედროვე პროგრამული პროდუქტებისადმი წაყენებული მოთხოვნა – არ ხდება დოკუმენტის შემცველობისა და მისი წარმოდგენის სახის აღწერის განცალკევება.

რაც შეეხება ასეთი ცხრილების გამოყენების უფრო მეტად გავრცელებულ შემთხვევას (შესაბამისი კოდის გაფორმებას css გაფართოების მქონე ცალკე ფაილად), დასამუშავებულ ფაილთან css ფაილის მიერთება ხორციელდება პირველის head სექციაში შემდეგი შემცველობის link ელემენტის ჩართვით:

```
<head>
 .....
   <link    href = “სტილების_ფაილის-სახელი.css”    rel = “stylesheet”
   type = “text/css”>
</head>
```

იმავე მიზნის განსახორციელებლად არსებობს სხვა გზაც – @import დირექტივისადმი მიმართვა ელემენტიდან:

```
<head>
   .....
   <style type = "text/css">
        @import url(style.css)
   </style>
</head>
```

გადავდივართ სტილების შემნახველი კასკადური ცხრილებისათვის კოდის დაწერის საკითხზე.

აღნიშნული კასკადური ცხრილებისათვის შაბლონი არის ამგვარი სახის:

ელემენტი.სტილის-სახელი {თვისება-1: მნიშვნელობა; **თვისება-2:** მნიშვნელობა; ...}

აღვნიშნოთ, რომ ასეთი კონსტრუქციები ობიექტ-ორიენტირებული დაპროგრამების ენებისთვის არის დამახასიათებელი და, ცხადია, მათ სინტაქსს რამდენადმე განსხვავებული სახე აქვს, ვიდრე საკუთრივ HTML ენისათვის განკუთვნილს.

დავუბრუნდეთ ზემოთ მოყვანილი სახელდებული სტილის შექმნისათვის გამიზნული კოდის ფრაგმენტს. ამჯერად უფრო დაწვრილებით განვიხილოთ მისი შემცველობა:

```
<style  type = “text/css”>
    h1.red1 {color: rgb(215,40,40); text-align: center}
</style>
```

type = “text/css” ატრიბუტი ბროუზერის ანალიზატორს ამცნობს, როგორი ტიპის მონაცემებს შეიცავს მოცემული <style> ელემენტი.

შემდეგ, h1 პირველი დონის სათაურებისათვის ფორმირდება red1 სახელის მქონე კლასი, რომლისთვისაც ფიგურულ ფრჩხილებში განსაზღვრულია ამა თუ იმ თვისების (თვისებების) გარკვეული მნიშვნელობები. შესაბამისად, h1 ელემენტებისათვის იქმნება კლასი red1 სახელით. ხოლო HTML კოდში რომელიმე ადგილას მყოფი h1 ელემენტისათვის red1 თვისებების მისაკუთვნებლად ვიყენებთ ასეთ კონსტრუქციას:

<h1 class = “red1”> *სათაურის ტექსტი* </h1>

ზემოთ მოყვანილ ფრაგმენტში, red1 კლასის გარდა, ცხადია, შეგვეძლო შეგვექმნა სხვა კლასებიც: red2, red3 და ა.შ., რომელთა მეშვეობითაც სხვადასხვა სტილი განესაზღვრებოდა ერთსა და იმავე ელემენტს *(მოცემულ შემთხვევაში პირველი დონის სათაურს).*

მოვიყვანოთ CSS-ფაილის შემცველობის სხვა მაგალითიც:

```
p.spec1 {color: green; font-variant: small-caps;}
p.new1 {color: maroon; font-style: italic;}
p.new2 {color: maroon; font-style: italic; letter-spacing: 2pt;}
```

ამრიგად, კოდის ამა თუ იმ ელემენტისათვის შეგვიძლია შევქმნათ სხვადასხვა სახელის მქონე სტილები (კლასები), რითაც ადგილზე ელემენტების გაფორმების მხრივ „მენიუ“ მრავალფეროვანი გახდება. მაგრამ CSS გვაწვდის უფრო მეტ შესაძლებლობასაც, კერძოდ, განსხვავებული ელემენტების *(მაგალითად, სათაურებისა და ჩვეულებრივი ტექსტის)* სასურველ სტილში ერთი ოპერატორით დაფორმატების წესსაც, რომლის მიხედვით style ელემენტში კონკრეტული სტილის სახელს უნდა წავუმძღვაროთ # სიმბოლო. შედეგად შევქმნით ე.წ. **უნივერსალურ კლასს**.

მოვიყვანოთ მაგალითი:

```
<style   type = “text/css”>
    #steel {color: rgb(155, 180, 190)}
</style>
```

ხოლო ადგილზე გამოყენებული უნდა იქნეს id ატრიბუტი:

```
<h2  id = “steel”> სათაური </h2>
<p   id = “steel”> აბზაცის ტექსტი </p>
```

ზემოთ ვაჩვენეთ, თუ როგორ უნდა მოხდეს WEB-ფურცლისათვის ამ თუ იმ სტილის გამოძახება CSS-ფაილიდან. ამჯერად ისიც აღვნიშნოთ, რომ ეს პროცესი შესაძლებელია განმეორდეს – მოხდეს სტილების განმეორებითი, ერთმანეთის მიყოლებით გამოძახებაც. სწორედ, ამიტომ ფიგურირებს მიდგომის სახელწოდებაში ტერმინი **კასკადური**. მაგალითად, შეიძლება ერთმანეთს დაედოს WEB-ფურცლის ავტორის, სერვერის და მომხმარებლის სტილი. შედეგად:

ყოველი ახლად გამოძახებული სტილის ცხრილი კორექტირებას უკეთებს წინას.

როგორც ვნახეთ, ელემენტების სტილის განსაზღვრა-შეცვლისათვის სხვადასხვა საშუალებები არსებობს, ამასთან, ხშირად მოცემული ელემენტის “ბედს” (ფერი, სისქე-სიგანე და ა.შ.) რამდენიმე “მბრძანებელი” განაგებს.

ელემენტებზე განსახორციელებელი ქმედებებისათვის დადგენილია შემდეგი სახის პრიორიტეტულობა:

1. თუ html კოდში გარე css ფაილების გამოძახება რამდენჯერმე მეორდება, წინებთან შედარებით უპირატესობა ეძლევა მომდევნო გამოძახებებს;
2. შიგა ცხრილებს ყოველთვის პრიორიტეტი აქვს;
3. ელემენტისათვის მისი ატრიბუტის მიერ განასაზღვრული სტილი “ამარცხებს” ნებისმიერ სხვას.
4. უფრო კონკრეტებულ სტილს, მაგალითად, p.classname {...}-ს პრიორიტეტი აქვს ნაკლებად დაკონკრეტებულ, მაგალითად, p

{..} სტილთან შედარებით. მაშასადამე, სტილის კლასს პრიორიტეტი აქვს ელემენტის თვისებების ხელახლა განსაზღვრებასთან შედარებით, ხოლო კომბინირებულ კლასს – ცალკეული სტილის კლასთან შედარებით.

5. თუ ელემენტთან რამდენიმე კლასია მიბმული, უპირატესობა ეძლევა მარჯვნივ მდგომთ.
6. თუ სტილის ატრიბუტი ცხადდება !important-ად, იგი "ხელშეუვალია"!

დასასრულ, მწყობრში მოვიყვანოთ ქვეთავში წარმოდგენილი მასალა:

CSS, ზოგადად, შემდეგი სახით მოიცემა:

<სელექტორი> {

<სტილის ატრიბუტი №1>: <ატრიბუტის მნიშვნელობა>;

<სტილის ატრიბუტი №2>: <ატრიბუტის მნიშვნელობა>;

...

<სტილის ატრიბუტი № >: <ატრიბუტის მნიშვნელობა>;

}

სელექტორი ეწოდება ელემენტის ან ელემენტების ჯგუფის აღწერას, რომლებისთვისაც გამოყენებული უნდა იქნეს მათი ატრიბუტებისათვის განსაზღვრული სტილი.

სტილის ატრიბუტი ვებ-ფურცლის ერთ-ერთი პარამეტრია. იგი ხშირად ელემენტის ატრიბუტთან არის გაიგივებული, მაგრამ მათ შორის არსებობს განმასხვავებელი ნიუანსებიც.

CSS სტილები ერთმანეთისაგან გამოიყოფა არა წერტილ-მძიმით, არამედ შუალედით (ანუ ხარვეზით) ანდა სტრიქონის გადატანის სიმბოლოთი.

რაც შეეხება სელექტორებს და სტილების სახელებს, მათ აღწერებში დაუშვებელია შუალედი-სიმბოლოების და სტრიქონის გადატანის სიმბოლოების არსებობა.

ქვემოთ მოცემულია სელექტორების სახეები, აღწერები და მიღებული შედეგების განმარტებანი:

სელექტორი	CSS-ში გაფორმება	კვალიფიცირებული ელემენტის გაფორმება	განმარტება
სელექტორი ელემენტებისათვის	P {*color*: *red* }	...	სტილი გამოყენებული იქნება ყველა აბზაცისათვის – შრიფტი მიიღებს წითელ ფერს.
სელექტორი კლასებისათვის	.classname {*color*: *red*}	...	სტილი გამოყენებული იქნება აბზაცებისთვის, რომელთა class ატრიბუტს ექნება შესაბამისი მნიშვნელობა.
სელექტორი იდენტიფიკა-ტორებისთვის	#clrRed { *color*: *red* }	...	გამოიყენება id ატრიბუტის შემცველი ელემენტისათვის, თუ ატრიბუტს ექნება შესაბამისი და, ამასთან, **უნიკალური** მნიშვნელობა.
სელექტორი შვილობილი ელემენტებისთვის	#clrRed>strong { *color*: *red*}	...	სტილი გამოიყენება ელემენტის შვილობილი თითოეული ელემენტისათვის.
კონტექსტური სელექტორი (კომბინირებული სტილი)	p strong {*color* : *red*}	...	სტილი გამოიყენება ელემენტში ჩადგმული ელემენტებისთვის.
	p.classname {*color*: *red*}	...	სტილი გამოიყენება იმ ელემენტებისათვის, რომელთა <class> ატრიბუტის მნიშვნელობა არის classname.

	p.classname { *color*: *red*}	...	სტილი გამოიყენება მხოლოდ ისეთ ელემენტებში ჩადგმული ელემენტებისთვის, რომელთათვისაც <class> ატრიბუტს მნიშვნელობად მიცემული ექნება classname.

სტილის დანიშვნისას კოდის კომპაქტურობის მიღწევის მიზნით დასაშვებია ისეთი სახის ნოტაციაც, რომლის მაგალითი ქვემოთ არის მოყვანილი:

```
p, .classname, td strong { color: red }
```

ზემოთ განვიხილეთ, ზოგადად, სტილების შემნახველი კასკადური ცხრილების რაობის და შექმნის საკითხები; გავარკვიეთ – მაშინ, როდესაც HTML ზრუნავს WEB-ფურცელზე გამოტანილი ინფორმაციის სტრუქტურული სახის ფორმირებაზე, CSS პასუხს აგებს ამავე დოკუმენტის გარეგნული სახის სრულყოფაზე. აქ კი, პირველი, რაც თვალში გვეცემა, ეს არის როგორც ეკრანზე ასახული უშუალოდ ინფორმაციისათვის, ისე მისი ფონისათვის ფერების შერჩევა. და ჩვენც სწორედ აქედან დავიწყოთ "ფრონტის წინა ხაზზე" განთავსებული CSS-ის საშუალებების შესწავლა.

ფერთა გამის განსაზღვრის ხერხები

ქვემოთ ჩამოთვლილია სტილების შემნახველ კასკადურ ცხრილებში გამოყენებული ის ატრიბუტები, რომლებიც ემსახურება ფერების განსაზღვრასა და ასევე, მათთან დაკავშირებული საკითხების გარკვევა-გადაწყვეტას ჩვენთვის ცნობილი HTML-ელემენტების უმეტესობისათვის:

- color – განსაზღვრავს ეკრანზე გამოტანილი ელემენტის წინა პლანის ფერს *(*color*: #00FF00);*

- background-color – განსაზღვრავს ელემენტის ფონის ფერს (background-color: brown);
- background-image – მიუთითებს იმ ფონურ გამოსახულებაზე, რომელიც ელემენტისათვის შექმნის ფონს (background-image: url("image.gif"));
- background-repeat – განსაზღვრავს წინა პუნქტში აღნიშნული გამოსახულების განმეორადობის ტიპს (background-repeat: no-repeat). შეუძლია მიიღოს შემდეგი მნიშვნელობები:
  - repeat-x – გამოსახულება მეორდება ჰორიზონტალზე;
  - repeat-y – გამოსახულება მეორდება ვერტიკალზე;
  - repeat – გამოსახულება მეორდება ჰორიზონტალზეც და ვერტიკალზეც;
  - no-repeat – გამოსახულება არ მეორდება (გათვალის-წინებულია დუმილით)
- background-attachment – განსაზღვრავს, ადგილზე რჩება თუ არა ელემენტის გადახვევისას background-attachment: fixed ფონური გამოსახულება. იღებს შემდეგ მნიშვნელობებს:
  - scroll – გადაიხვევა მასთან ერთად;
  - fixed – გადახვევა არ ხდება.
- background-position – მოიცავს ორ მნიშვნელობას: *პოზიციონირება ჰორიზონტალზე* და *პოზიციონირება ვერტიკალზე* (background-position: 5cm 4cm). გარდა რიცხვითისა, შეუძლია მიიღოს შემდეგი მნიშვნელობებიც:
  - left – ჰორიზონტალური პოზიციონირება "მიჯრა მარცხენა კიდესთან";

- center – ჰორიზონტალური პოზიციონირება "სწორება ცენტრზე";
- right – ჰორიზონტალური პოზიციონირება "მიჯრა მარჯვენა კიდესთან";
- top – ვერტიკალური პოზიციონირება "მაღლა აწევა";
- center – ვერტიკალური პოზიციონირება "შუაში განთავსება";
- bottom – ვერტიკალური პოზიციონირება "ქვემოთ ჩამოწევა".

CSS უშვებს შემდეგ შესაძლებლობასაც – ფონურ გამოსახულებასთან დაკავშირებული ყველა ატრიბუტს მნიშვნელობა განესაზღვროს კომპაქტური ნოტაციის მეშვეობით, მაგალითად, ამგვარად:

background: #00FF00 url("fig.gif") no-repeat fixed 5cm 4cm

შრიფტთან მუშაობა

ამთავითვე დავიხსომოთ: *მომხმარებლის დისპლეიზე აისახება მხოლოდ ის შრიფტები, რომლებიც დაყენებულია მის კომპიუტერზე, შესაბამისად, ჯობია, "ეგზოტიკური" შრიფტების გამოყენებისაგან თავი შევიკავოთ ან უკიდურეს შემთხვევაში, თუ მიგვაჩნია, რომ ეს აუცილებელია, ასეთი შრიფტებით გაკეთებული ჩანაწერები გრაფიკულ ფორმატში გადავიყვანოთ (შევნიშნავთ, რომ ნათქვამი არ ეხება უნიკოდს).*

ქვემოთ ჩამოთვლილია სტილების შემნახველ კასკადურ ცხრილებში გამოყენებული ის ატრიბუტები, რომლებიც ემსახურება შრიფტის მართვას:

- color – განსაზღვრავს ეკრანზე გამოტანილი ელემენტის წინა პლანის ფერს *(color: #00FF00)*. ფერი ფორმირდება 3 ბაზისური ფერის (წითელი – RR, მწვანე – GG, ლურჯი – BB) კომბინაციით, თითოეული მათგანის წილი განისაზღვრება ორთანრიგა 16-ითი რიცხვით 0-255 დიაპაზონში. კიდურა ვარიანტებია: #000000 – შავი და #FFFFFF – თეთრი ფერები. ამ და სხვა სტანდარტული

ვარიანტებისათვის დასაშვებია ფერის მითითება სიტყვიერადაც (ინგლისურ ენაზე):

Black	#000000
Maroon	#800000
Green	#008000
Olive	#808000
Navy	#000080
Purple	#800080
Teal	#008080
Gray	#808080
Silver	#C0C0C0
Red	#FF0000
Lime	#00FF00
yellow	#FFFF00
blue	#0000FF
Fuchsia	#FF00FF
Aqua	#00FFFF
White	#FFFFFF

- font-family – შრიფტის სახეს განსაზღვრავს. ამასთან, ეს შესაძლებელია მოხდეს როგორც შრიფტის სახელის მითითებით (Camria, Arial და ა.შ.), ისე – მისი სახეობის ჩვენებითაც
    - serif – ნაწიბურებიანი შრიფტები,
    - sans-serif – დაჩეხილი,
    - cursive – კურსივი,
    - fantasy – დეკორატიული,
    - monospace – ტოლი სიგანის მქონე,
    - font-style – სტილის აღმნიშვნელი (მაგ.: font-style: normal):
        - normal – ჩვეულებრივი,
        - italic – კურსივი,

        - oblique – დახრილი.
- font-variant – მიიღებს შემდეგ მნიშვნელობებს:
    - normal – სტრიქონული ასოები აისახება ჩვეულებრივად (საწყის რეგისტრში),
    - small-caps – სტრიქონული ასოები შეიცვლება ასომთავრულით, ოღონდ ზომა უმცირდებათ.
- font-weight :
    - normal – სტანდარტული სისქე;
    - bold – გასქელებული მოხაზულობის. აქვე შევნიშნოთ, რომ ბრაუზერების ნაწილს შეუძლია 100-900 დიაპაზონში მითითებული ინტენსივობით დისპლეიზე შრიფტის ასახვაც. სტანდარტულად ითვლება მნიშვნელობა 700.
- font-size – განსაზღვრავს შრიფტის ზომას (აბსოლუტური ან ფარდობითი სახით მითითებული კონსტანტების მეშვეობით), მაგალითად: (font-size*:* 12pt), (font-size*:* +3);

დასასრულ, აღვნიშნოთ, რომ ამ შემთხვევაშიც დასაშვებია კომპაქტური ნოტაციის გამოყენება:

font: normal bold 16pt acadnusx

ტექსტთან მუშაობა

CSS-ში ატრიბუტების ეს ჯგუფი განკუთვნილია მთელ ტექსტთან ან მის შედარებით დიდი ზომის ფრაგმენტებთან სამუშაოდ, კერძოდ, მათი პოზიციონირების, დაძვრების, სტრიქონების წყვეტის და სხვ. ამოცანების გადასაწყვეტად:

- text-align – იღებს მნიშვნელობებს ჰორიზონტალზე ტექსტის პოზიციონირებისათვის:
    - center, left, right, justify;
    - auto – გასწორების ტიპი უცვლელი რჩება;

- start – კიდეებთან ტექსტს ასწორებს მისი მიმართულების მიხედვით – მარჯვნიდან მარცხნივ მიმართულ ტექსტს მიაბჯენს მარჯვენა კიდეს და პირიქით;
- end – თუ ტექსტი მიმართულია მარცხნიდან მარჯვნივ, მაშინ სწორება ხდება მარჯვენა კიდესთან და პირიქით (საქმე გვაქვს წინა პუნქტში აღწერილის საწინააღმდეგო ქმედებასთან);

- text-*align*-last – გავლენას ახდენს ელემენტის ბოლო სტრიქონის გასწორების სახეზე მაშინ, როდესაც text-*align* ატრიბუტის მნიშვნელობა *justify*-ის ტოლია. ასეთ შემთხვევაში მას შესაძლებელია მიეცეს text-*align* ატრიბუტის ანალოგიური მნიშვნელობები.
- *text-decoration* – განკუთვნილია ტექსტის სხვადასხვა სახის ეფექტებით გაფორმებისათვის. მნიშვნელობებია:
  - *blink* – ციმციმა;
  - line-through – გადახაზული;
  - *overline* –ტექსტის თავზე ხაზის გავლება;
  - *underline* – ხაზგასმა (ქვემოდან);
  - none – ეფექტების გარეშე.
- *text-indent* – ტექსტის პირველი სტრიქონის დაძვრა პროცენტების ან კონკრეტული რიცხვითი სიდიდეების მითითებით, მაგალითად, *text-indent*: 10%;
- text-*overflow* – არის გადავსებისას “ზედმეტი” ტექსტის ჩამოჭრა (text-*overflow*: *clip*), ანდა მისი მრავალწერტილით დაბოლოება (text-*overflow*: *ellipsis*);
- *text-shadow* – ტექსტს უმატებს გარკვეული პარამეტრების მქონე ჩრდილს, მაგალითად, *text-shadow*: *red* 5 5. პარამეტრებმა შეიძლება მიიღონ შემდეგი მნიშვნელობანი:
  - none – ჩრდილის გარეშე;
  - ფერი – ნებისმიერი შესაძლო ფერი;

- დაძვრა ჰორიზონტალზე – დადებითი მნიშვნელობა ჩრდილს მარჯვნივ დაძრავს, ხოლო უარყოფითი – მარცხნივ.
- დაძვრა ვერტიკალზე – დადებითი მნიშვნელობა ჩრდილს ქვემოთ ჩამოსწევს დაძრავს, ხოლო უარყოფითი – მაღლა ასწევს;
- გათქვეფის რადიუსი – მეტი მნიშვნელობა ჩრდილს **ნაკლებ** სიმკვეთრეს ანიჭებს, დუმილით კი პარამეტრის მნიშვნელობა ნოლის ტოლია.

- *text-transform* – ტექსტის ტრანსფორმირება სხვადასხვა სახესა და რეგისტრებში (WORD-ის ანალოგიურად):
  - none;
  - *capitalize*;
  - lowercase;
  - *uppercase*.

ინტერნეტის მომავალი

ჯერ კიდევ ბოლო საუკუნეების მიჯნაზე მეცნიერებმა დაიწყეს სერიოზული მსჯელობა ინტერნეტის – ამ მსოფლიო გლობალური ქსელის განვითარება-სრულყოფის გზების შესახებ.

რა თქმა უნდა, საერთოდ, როდესაც ადამიანის მოღვაწეობის ამა თუ იმ სფეროს პერსპექტივებზეა საუბარი, ძალიან ხშირად საქმის ნაკლებად მცოდნე ხალხი გასაქანს აძლევს საკუთარ ფანტაზიას და ხშირად არარეალური მიზნების განხორციელებაზე ოცნებობს, მაშინ როდესაც მათი ყურადღების მიღმა რჩება ნაკლებად ეფექტური, მაგრამ, კაცობრიობისათვის სიკეთის მოტანის თვალსაზრისით, გაცილებით საინტერესო, ამასთან, რეალურად განხორციელებადი პროექტები.

რაც შეეხება ინტერნეტის მომავალს, სწორედ ამ კუთხითაც იქცევს ჩვენს ყურადღებას დარგის საყოველთაოდ აღიარებული სპეციალისტების

მიერ გამოთქმული მოსაზრება, რომ აღნიშნულ სფეროში მომავალი ეკუთვნის ე.წ. *სემანტიკურ ვებ-სივრცეს.*

აი, მათ მიერ აღნიშნული ცნებისათვის მოცემული განმარტება:

"სემანტიკური ვებ-ი წარმოადგენს იმ შესაძლებლობების ერთიანობას, რომელიც კომპიუტერებს საშუალებას მისცემს, გაერკვეს დოკუმენტების თუ მონაცემების სემანტიკაში და არა ადამიანის მეტყველებაში ან, ვთქვათ, მის მაგივრად შეძლოს აზრის ჩამოყალიბება".

ტ. ბერნერს-ლი, ჯ. ჰენდლერი, ო. ლასილა

სემანტიკური ვებ-ი, 2001

საერთოდ, როდესაც ასეთ გლობალურ პრობლემას განიხილავენ, ძალიან მნიშვნელოვანი ხდება ტერმინების დაზუსტების საკითხი.

თავდაპირველად დავსვათ კითხვა:

რა განსხვავებაა (ხშირად იდენტურად მიჩნეულ) ორ ცნებას – ინტერნეტსა და ვებ-ს – შორის?

მოგვყავს განმარტებები:

ინტერნეტი არის მსოფლიო მასშტაბით ერთმანეთთან არხებით დაკავშირებული ისეთი კომპიუტერების ქსელი, რომლებიც ამ კავშირის უზრუნველსაყოფად იმართება TCP/IP პროტოკოლებით. მას საფუძველი ჩაუყარეს სპეციალისტებმა: Vint Cerf და Robert Kahn-მა.

ვებ (WWW) წარმოადგენს საიტების ქსელს (მსოფლიო მასშტაბით), რომლებიც ერთმანეთთზე გადასასვლელად (ცალკეული ფურცლების თუ მონიშნული ადგილების დონეზეც) იყენებენ ჰიპერკავშირებს. მისი გამომგონებელია ტიმ ბერნერს-ლი.

მაინც, რა ფუნქციების დაკისრებას ვარაუდობენ მეცნიერები სემანტიკური ქსელისათვის?

უმარტივესი სცენარის მაგალითია ვებ-აგენტისათვის ასეთი დავალების მიცემა:

ესა და ეს წიგნები შეუკვეთე ჩემთვის უახლოეს ბიბლიოთეკაში (ბიბლიოთეკებში).

მოვიყვანოთ ვებ-აგენტისათვის მიცემული სხვა, რამდენადმე უფრო რთული დავალების მაგალითიც:

შეუკვეთე თეატრში 2 ბილეთი ამა და ამ სპექტაკლზე; ჩემი სამუშაოს დამთავრების დროისათვის უზრუნველყავი სატრანსპორტო მომსახურება ისე, რომ სპექტაკლზე დროზე მივიდეთ მე და ჩემი მეგობარი, რომელიც მანამდე საქმის კურსში უნდა ჩააყენო და თხოვე მას ჩემთან დროულად დაკავშირება.

კიდევ ერთი მაგალითი, რომლის შესრულების საჭიროებას იმედია, მომავალში ჩვენც გავაცნობიერებთ:

რესტორანში შეკვეთილ მენიუში შემავალი კერძებისათვის ცალ-ცალკე გაარკვიე, რომელი მათგანისათვის როგორი მარკის ღვინოა რეკომენდებული. ამასთან, გაითვალისწინე, რომ მათ შორის "ტოკაი" გამორიცხული უნდა იყოს.

დავალება: *მოიფიქრეთ სხვა შეკვეთილი სცენარების მაგალითები.*

ზოგი რამ ქრონოლოგიიდან:

- 1994: W3C კონსორციუმის ჩამოყალიბება. მისი წევრების მიერ შემუშავებული იქნა შემდეგი სტანდარტები:
- HTML, URL, XML, HTTP, PNG, SVG, CSS.
- 1998: ტიმ ბერნერს-ლი აქვეყნებს სემანტიკური ვების (Semantic Web Road map) გეგმას.
- 1999: W3C კონსორციუმი აყალიბებს ჯგუფებს დასახული მიზნის განსახორციელებლად. ქვეყნდება სემანტიკური ვების სინტაქსის პირველი ვერსია RDF.
- 2000: ამერიკელი სამხედროები იწყებენ მუშაობას ონტოლოგიების აღწერის მიმართულებით (DAML+OIL project)
- 2001: ჟურნალ Scientific American-ში ქვეყნდება სემანტიკური ვების აღწერა

- 2004: გამოქვეყნდა RDF-ის ახალი ვერსია, წარმოდგენილი იქნა ონტოლოგიების აღწერის ენა OWL.
- 2006: წარმოდგენილი იქნა მოთხოვნების ფორმირების ენის ვერსია SPARQL

დავალება: ეცადეთ, თვითონ შექმნათ კაცობრიობის მიერ უხსოვარი დროიდან დაგროვილი ცოდნის კომპიუტერში დასაფიქსირებლად განკუთვნილი ინსტრუმენტარიუმის მოდელი. არა უშავს, თუ ცდა წარუმატებლად დამთავრდება (დაგამშვიდებთ, რომ თქვენამდეც იყო ყველასათვის, მათ შორის, ცხადია, პირველ რიგში კომპიუტერები იგულისხმება, გასაგები მეგა-ენის შექმნის მცდელობები, რომელთაც დიდი წარმატება არ მოჰყოლია). გავიხსენოთ, რომ "უარყოფითი შედეგიც შედეგია", რომელიც შემდგომში ახალ გზებს დაგვანახებს. სწორედ, ერთი ასეთი გზისკენს მიგვითითა იმავე ტიმ ბერნერს-ლიმ, მაგრამ სანამ ამ საკითხზე გადავიდოდეთ, გავიხსენოთ (ან ვისწავლოთ) ზოგი რამ ტერმინოლოგიიდან:

სინტაქსი – ენის ფრაზების, **ფორმალური** თვალსაზრისით, სწორად აგებისათვის გამიზნული წესების კრებული.

ცნების არსი განვმარტოთ შემდეგი სინტაქსურად გამართული და სინტაქსურად არასწორი ფრაზების მაგალითეზე:

სწორია: *"მოსწავლე წავიდა სკოლაში".*

სწორია: "*სკოლა წავიდა მოსწავლეში".*

არასწორია: "დაროზს რამდი დარი".

სემანტიკა – ენის ფრაზების, **აზრობრივი** თვალსაზრისით, სწორად აგებისათვის გამიზნული წესების კრებული **ცალკეული ენობრივი კონსტრუქციებისათვის**.

საერთოდ, სინტაქსისაგან განსხვავებით, სემანტიკური სისწორის გლობალურად შემმოწმებელი წესების ჩამოყალიბება, ფაქტობრივად შეუძლებელი ამოცანა გახლვთ, რის გამოც განმარტებაში ფიგურირებს ფრაზა "ცალკეული ენობრივი კონსტრუქციებისათვის", რაც მიგვითითებს,

რომ წესების ქმედითობას განვიხლავთ მხოლოდ შეზღუდული საგნობრივი სფეროსათვის.

დავუბრუნდეთ ზემოთ განხილულ მაგალითს: "*სკოლა წავიდა მოსწავლეში*". ცხადია, სემანტიკური თვალსაზრისით, იგი მცდარია.

დაპროგრამების ადრე შექმნილი საშუალებები, როგორც წესი, აქცენტირებას აკეთებდნენ კოდის სინტაქსურად გამართულობაზე, მაგალითად, ისეთი თანამედროვე ენისათვისაც კი, როგორიც არის XML, სწორად და კორექტულადაც (იხ. ამ ენის შესახებ სახელმძღვანელო) მიიჩნევა, ვთქვათ, შემდეგი კონსტრუქცია:

```
<Order OrderNo="1234">
    <OrderDate>2011-01-01</OrderDate>
    <Gvari>ორი ტონა კარტოფილი</Gvari>
    <Saxeli>არქტიკა</Saxeli>
</Order>
```

აქვე ისიც უნდა აღინიშნოს, რომ ჩვეულებრივი მეტყველებაც ძალიან შორს დგას მკაცრად დადგენილი სემანტიკური წესების დაცვისაგან. სწორედ, ამის გამო მხოლოდ მიახლოებული სიზუსტით ხორციელდება კომპიუტერული პროგრამების მიერ ერთი ენიდან მეორეზე ტექსტების თარგმანი და ისიც მეტნაკლებად ღირებული გახლავთ მხოლოდ "მშრალი" საგაზეთო სტილის ფრაზებისათვის.

მაინც რა შემოგვთავაზა ტიმ ბერნერს-ლიმ? მისი მიდგომა საკითხის გაადაწყვეტისადმი შემდეგი გახლავთ – ჰიპოთეზური მეგა-ენისათვის ცალ-ცალკე დამუშავდეს სინტაქსი და სემანტიკა:

RDF (Resource Description Framework) – სემანტიკური ვებ-ის დოკუმენტებისათვის სინტაქსი, რომელიც დაეყრდნობა ონტოლოგიების სტრუქტურას (შევნიშნოთ, რომ ონტოლოგია გულისხმობს რაიმე შემოზღუდული სფეროსათვის, მაგალითად, ავტომაღაზიისათვის, ობიექტების კლასების აღწერა თავისი თვისებებით, მეთოდებით, ინტერფეისის თავისებურებებით).

OWL (Ontology Web Language) – ონტოლოგიების აღწერის ენა.

ტიმ ბერნერს-ლის მიერ დასახული გეგმა RDF-ისა და OWL-ის შექმნის მიმართულებით მეტ-ნაკლებად განხორციელდა და დაიწყო მუშაობა შემდეგი მიმართულებებით:

- ვებ-სერვისების აღწერის (WSDL, OWL-S) ენის შესაქმნელად;
- სემანტიკური ვებ-ის დოკუმენტების კითხვა-დამუშავების ინსტრუმენტების (Jena, Haystack, Protege) დასამუშავებლად;
- RDF-ში ჩაწერილი ცოდნის ბაზიდან ინფორმაციის ამომკრეფი მოთხოვნების SPARQL ენის შესაქმნელად.
- არის კიდევ რიგი სამუშაოებისა, რომელთა შესრულებაც საჭიროა საბოლოო მიზნის განსახორციელებლად, ამასთან, მუშაობის პროცესში მოსალოდნელია ბევრი სხვა გადა-საწყვეტი საკითხის წამოჭრაც.

დანართი № 1

სახელების სივრცე

სახელების სივრცე განიმარტება, როგორც მოცემულ სიმრავლეში ამა თუ იმ წესით ერთმანეთთან დაკავშირებული და ამასთან, ამ სიმრავლეში უნიკალური სახელების, ტერმინების, იდენტიფიკატორების ერთობლიობა.

სახელების სივრცის შემოტანა-გამოყენება განპირობებულია შემდეგი გარემოებით:

შესაძლოა, WEB-დოკუმენტში (და, ცხადია, არა მარტო მასში) ფიგურირებდეს ერთნაირი სახელის, მაგრამ განსხვავებული დანიშნულების (შესაბამისად, განსხვავებული ტიპის მონაცემების შემცველი) ელემენტები.

მაგალითად, XML-დოკუმენტში წიგნის მაღაზიაში წიგნებზე მოთხოვნას შეიძლება ასეთი სახე ჰქონდეს:

```
<?xml version="1.0"?>
<BookOrder OrderNo="1234">
    <OrderDate>2001-01-01</OrderDate>
    <Customer>
        <Title>Mr.</Title>
        <FirstName>Graeme</FirstName>
        <LastName>Malcolm</LastName>
    </Customer>
    <Book>
        <Title>Treasure Island</Title>
        <Author>Robert Louis Stevenson</Author>
    </Book>
</BookOrder>
```

ვხედავთ, რომ დოკუმენტი შეიცავს სხვადასხვა დანიშნულების მქონე ორ Title ელემენტს. ერთი მათგანია მიმართვა მყიდველისადმი, მეორე – წიგნის სახელწოდება.

გაურკვევლობის თავიდან ასაცილებლად ასეთ შემთხვევაში XML იყენებს ე.წ. **სახელების სივრცეებს,** რომელთა არსი შემდეგია:

სახელების სივრცის მეშვეობით ხდება XML-დოკუმენტის ელემენტების და ატრიბუტების დაკავშირება (მიბმა) რაიმე უნივერსალურ იდენტიფიკატორთან – URI-სთან. აქვე აღვნიშნავთ, რომ Universal Resource Indentifier, ანუ **რესურსის უნივერსალური მაჩვენებელი** შეიძლება წარმოადგენდეს URL-ს, მაგრამ შესაძლოა, იგი იყოს რაიმე სხვა უნიკალური იდენტიფიკატორიც (ე.ი არ წარმოადგენდეს რესურსს ინტერნეტიდან).

სახელების სივრცე შესაძლებელია ნებისმიერ ელემენტში გამოცხადდეს.

ამ მიზნით XML-ში გამოიყენება xmlns ატრიბუტი. ბუნებრივია, რომ ამ ატრიბუტის მნიშვნელობა დუმილით ვრცელდება კვალიფიცირებული ელემენტის შემადგენელ ქვეელემენტებზეც.

წიგნებზე მოთხოვნა შემდეგნაირად დავაზუსტოთ:

```
<?xml version="1.0"?>
<BookOrder OrderNo="1234">
    <OrderDate>2001-01-01</OrderDate>
    <Customer
xmlns="http://www.northwindtraders.com/customer">
        <Title>Mr.</Title>
        <FirstName>Graeme</FirstName>
        <LastName>Malcolm</LastName>
    </Customer>
    <Book xmlns="http://www.northwindtraders.com/book">
        <Title>Treasure Island</Title>
        <Author>Robert Louis Stevenson</Author>
```

```
    </Book>
</BookOrder>
```

ერთნაირი სახელების მქონე ელემენტების პრობლემა შემდეგნაირად გადაწყდა:

Customer და შესაბამისად მასში ჩადგმული ელემენტები, მაშასადამე, Title ელემენტიც, მიეკუთვნება ინტერნეტის ერთ-ერთ მისამართზე, მაგალითად, http://www.northwindtraders.com/customer-ზე არსებულ სახელების სივრცეს, ხოლო Book ელემენტი და, ცხადია, მასში ჩადგმული ასევე Title სახელის მქონე სხვა ელემენტი http://www.northwindtraders.com/book მისამართზე ფიქსირებულ სახელების სივრცეს.

ვხედავთ, რომ სახელთა სივრცის ამგვარი წესით გამოცხადება მთლად მოხერხებული არ არის, განსაკუთრებით მაშინ, როცა მას უკავშირდება ბევრი ელემენტი და ატრიბუტი. ასეთ შემთხვევებში მიმართავენ საკითხის ალტერნატიულ გადაწყვეტას - ფესვურ ელემენტში ცხადდება ყველა ასეთი სივრცე. ერთ-ერთი მათგანი დუმილით გაითვალისწინება პრეფიქსით მოუნიშნავი ელემენტებისა და ატრიბუტებისათვის, დანარჩენი სივრცეებისთვის კი გამოცხადდება აბრევიატურები, რომელთა მეშვეობითაც მოინიშნება საჭირო ელემენტები და ატრიბუტები (აბრევიატურა მათი სახელის წინ პრეფიქსის როლში მოგვევლინება).

რადგანაც აბრევიატურა სულ რამდენიმე სიმბოლოსაგან შედგება, კოდის უკეთ და სწრაფად აღსაქმელად მას წაუმძღვარებენ ქვეელემენტებს იმ შემთხვევაშიც კი, როცა XML-ანალიზატორს ასეთი გადაწყვეტის გარეშეც შეუძლია ვითარებაში გარკვევა.

მოვიყვანოთ მაგალითი:

```
<?xml version="1.0"?>
<BookOrder xmlns="http://www.northwindtraders.com/order"
    xmlns:cust="http://www.northwindtraders.com/customer"
    xmlns:book="http://www.northwindtraders.com/book"
```

```
        OrderNo="1234">
    <OrderDate>2001-01-01</OrderDate>
    <cust:Customer>
        <cust:Title>Mr.</cust:Title>
        <cust:FirstName>Graeme</cust:FirstName>
        <cust:LastName>Malcolm</cust:LastName>
    </cust:Customer>
    <book:Book>
        <book:Title>Treasure Island</book:Title>
        <book:Author>Robert Louis Stevenson</book:Author>
    </book:Book>
</BookOrder>
```

დოკუმენტი ეყრდნობა სამ სახელთა სივრცეს:

- http://www.nothwindtraders.com/order - დუმილით;
- http://www.nothwindtraders.com/customer - აღნიშნული სივრცისთვის გამოცხადდა cust აბრევიატურა;
- http://www.nothwindtradesr.com/book - ამ სივრცისთვის კი - book აბრევიატურა.

დოკუმენტის ელემენტებისა და ატრიბუტების წინ დასმულია სათანადო პრეფიქსები, ხოლო ელემენტები და ატრიბუტები, რომელთა სახელწოდებებში პრეფიქსი არ ფიგურირებს (მოცემულ მაგალითში ესენია: BookOrder, OrderNo და OrderDate), მიეკუთვნება დუმილით განსაზღვრულ სახელების სივრცეს.

დანართი № 2

TCP/IP დამისამართება, ქვექსელის ნიღბები

ინტერნეტში (და არა მარტო ინტერნეტში) კომპიუტერებს (ფაქტობრივად, ქსელურ კვანძებს) შორის ურთიერთობის დასამყარებლად საჭირო ხდება მანქანისათვის გასაგებ ენაზე მათი იდენტიფიცირება – დანომრვა, რაზეც ზრუნავს **TCP/IP** ოქმის ერთ-ერთი შემადგენელი **IP** ნაწილი (აქვე შევნიშნავთ, რომ **IP** ოქმი პასუხისმგებელია დანომრილი პაკეტის ქსელში გადაცემაზეც, ხოლო **TCP** ოქმს ევალება გადასაცემი შეტყობინების პაკეტებად დაყოფა და დანიშნულების ადგილზე მათი მიღების შემდეგ ამ შეტყობინების ხელახლა პირვანდელი სახით აწყობა).

შემდეგ, კომპიუტერის დამისამართებისათვის **IP** ოქმი იყენებს 4 ბაიტს, რითაც შესაძლებელი ხდება, დღეს ინტერნეტში ჩართული აბსოლუტურად ყველა კომპიუტერი უნიკალური სახით იქნეს იდენტიფიცირებული (აქვე დავაზუსტოთ ზოგი დეტალიც: არა მარტო მარშრუტიზატორს აქვს, თავისი ფუნქციიდან გამომდინარე, სხვადასხვა ქსელთან დამაკავშირებელი თითოეული პორტისათვის განსაზღვრული უნიკალური **IP** მისამართი, არამედ, ასეთი შესაძლებლობის არსებობა დასაშვებია ქსელური კვანძ-კომპიუტერისთვისაც, როდესაც უკავშირდება რამდენიმე ქსელს. ამრიგად, რეალურად, **IP** ამისამართებს არა ცალკეულ კომპიუტერს ან მარშრუტიზატორს, არამედ მათ თითოეულ ქსელურ კავშირს).

მოვიყვანოთ კომპიუტერის სახელდების მაგალითი – მის **IP** მისამართს შეიძლება ასეთი სახე ჰქონდეს:

24. 0. 64. 18

IP მისამართში თითოეული ბაიტი მნიშვნელობებს იღებს 0 – 255 დიაპაზონში, ანუ სულ – 256 განსხვავებულ მნიშვნელობას.

კომპიუტერის 4-ბაიტიან მისამართს, როგორც წესი, შემდეგი ორნაწილიანი სტრუქტურის სახით წარმოადგენენ:

- კომპიუტერის გარემომცველი ქსელის ნომერი (*მისამართის პირველი ნაწილი);*
- ზემოაღნიშნულ ქსელში საკუთრივ კომპიუტერის ნომერი *(ბოლო ნაწილი).*

დასაშვებია, კონკრეტულ მისამართებში ამ ნაწილებს დაეთმოს თანრიგების განსხვავებული რაოდენობები.

4 ბაიტი ზომის მეხსიერება თეორიულად საშუალებას იძლევა დამისამართდეს დაახლოებით 4 მილიარდი კომპიუტერი. მთელი ეს სიმრავლე კი დაყოფილია კლასებად. პრაქტიკული მიზნებიდან გამომდინარე, მათგან განვიხილავთ ჩვენთვის განსაკუთრებით საინტერესო შემდეგ სამ კლასს:

A კლასი. იგი მსხვილი ქსელების კლასია, რომელშიც დამისამართებულია 126 ქსელი *(შესაბამისად, მათი ნომრები ვარირებს 1 – 126 დიაპაზონში).*

B კლასს იმავე პირველ ბაიტში ეთმობა დიაპაზონი 128 – 191 (127-ე ნომერი გამოიყენება სპეციალური მიზნებისათვის). მაგრამ, რადგანაც **B** კლასის ქსელების რიცხვი გაცილებით მეტია, ვიდრე **A** კლასისა, ქსელის ნუმერაცია გრძელდება მომდევნო ბაიტშიც (იხ. ქვემოთ).

C კლასისათვის კი პირველ ბაიტში დათმობილი დიაპაზონი არის 192 – 223. ქსელების რაოდენობის კიდევ უფრო მეტად გაზრდის გამო წინა კლასებთან შედარებით, ნუმერაციისათვის უკვე გამოყოფილია 3 ბაიტი.

A კლასი განკუთვნილია უმსხვილესი ორგანიზაციებისათვის. ამ ტიპის 126 ქსელიდან თითოეულ მათგანში შეიძლება განთავსდეს (256*256*256 - 2) რაოდენობის კომპიუტერი. **IP**-მისამართებიდან ორი მათგანი, კერძოდ, 0 და 255 მნიშვნელობები, ამ და სხვა კლასებშიც სპეციალური მიზნებისათვის არის განკუთვნილი (იხ. ქვემოთ).

B კლასი ეთმობა საშუალო მასშტაბის ორგანიზაციებს, კორპო-რაციებს, უნივერსიტეტებს... რადგანაც ასეთი ორგანიზაციების რიცხვი გაცილებით მეტია, როგორც უკვე აღვნიშნეთ, თვით ქსელების

იდენტიფიცირებისას მათი რაოდენობის გასადიდებლად, პირველის გარდა, იყენებენ მომდევნო – მეორე ბაიტსაც. მთლიანობაში B კლასის ქსელების რიცხვი 16 384-ით განისაზღვრება (64*256), თითოეულ მათგანში კი შეიძლება ფუნქციონირებდეს 65 534-მდე კომპიუტერი (256*256 - 2).

C კლასი ლოკალურ ქსელებზეა ორიენტირებული. კომპიუტერის იდენტიფიცირებისათვის საჭირო მეხსიერების კიდევ უფრო გაზრდით C კლასის ქსელების რიცხვი 532 676 608-ს აღწევს, თითოეულში 254 კომპიუტერის გაერთიანების შესაძლებლობით.

სამივე ტიპის ქსელები, ძირითადად, ინტერნეტში მუშაობისთვის არის განკუთვნილი და ზემოთ აღწერილი წესით დამისამართება მსოფლიოს ნებისმიერ კუთხეში არსებულ კომპიუტერებს შორის "გაუგებრობებს" გამორიცხავს.

ისეთ შემთხვევაში კი, როცა დარწმუნებული ვართ, რომ ქსელს ინტერნეტთან კავშირი არ ესაჭიროება, შეიძლება მივმართოთ კომპიუტერების ე.წ. კერძო დამისამართებასაც.

კერძო დამისამართება

თუ ლოკალურ კომპიუტერულ ქსელს ინტერნეტში გასვლა არ სჭირდება, დასაშვებია მასში გაცილებით მეტი კომპიუტერები ჩავრთოთ, ვიდრე ეს ხელმისაწვდომი პროვაიდერის მიერ გამოყოფილ სეგმენტშია შესაძლებელი.

ამ მიზნით, კომპიუტერების იდენტიფიცირებისათვის იყენებენ კერძო დამისამართების ხერხს, რომლის არსია შემდეგი:

კომპიუტერის მისამართის პირველი ნაწილი *(რომელიც, როგორც უკვე ვიცით, თვით ქსელის დამისამართებას ახდენს)* ამ შემთხვევაში წარმოადგენს თავისებურ, ამოქოლილ გასასვლელს, რისთვისაც გამოიყენება სპეციალური მისამართები:

A კლასისთვის 10.0.0.0–10.255.255.255 დიაპაზონში; ქვექსელის ნიღაბი (იხ. ქვემოთ) გახლავთ 255.0.0.0, რაც საშუალებას იძლევა გამოიყოს 16 მლნ-ზე მეტი მისამართი.

B კლასს ეთმობა 172.16.0.0–172.31.255.255 დიაპაზონი ანუ 16 ქსელი. ამ შემთხვევაში ქვექსელის ნიღბის მნიშვნელობაა 255.255.0.0 და ითვალისწინებს თითოეულ ასეთ ქსელში 65 ათასზე მეტი მისამართის დანიშვნას.

C კლასისათვის გამოიყენება 192.168.0.0–192.168.255.255 დიაპაზონი; ქვექსელის ნიღაბი 255.255.255.0 საშუალებას იძლევა გამოიყოს 254 მისამართი.

ასეთი მისამართები კონფლიქტებს გამორიცხავს მაშინაც კი, როცა ლოკალური ქსელი ინტერნეტს მიუერთდება, რადგანაც კერძო მისამართი ინტერნეტით არ გადაიცემა.

სტანდარტული და მომხმარებლის მიერ დანიშნული ქვექსელის ნიღბები

ამრიგად, სტანდარტული ნიღბებია:

A კლასისთვის – 255.0.0.0;

B კლასისთვის – 255.255.0.0;

C კლასისთვის – 255.255.255.0.

ქვექსელის ნიღაბი ზღუდავს იმ კომპიუტერების რაოდენობას, რომლებსაც უშუალოდ შეუძლია მიმართოს მოცემულმა კომპიუტერმა. როდესაც ქსელში კომპიუტერს ვამატებთ, მისთვის უნდა შევირჩიოთ იგივე ნიღაბი, რომელიც გამოიყენება ამ ქსელში უკვე ჩართული კომპიუტერებისათვის.

სტანდარტული ნიღბების გარდა, შეიძლება გამოყენებული იქნეს მომხმარებლების მიერ შექმნილი ნიღბებიც. ამ საკითხში უკეთ გასარკვევად რომელიმე ნიღაბი, მაგალითად, 255.255.255.0, წარმოვადგინოთ ორობით სისტემაში:

11111111.11111111.11111111.00000000

ბოლო ბაიტში, როგორც უკვე ვიცით, შეიძლება განვათავსოთ 254 მისამართი, მაგრამ თუ მიზანშეწონილად მიგვაჩნია 254 კომპიუტერის შემცველი ერთი ლოკალური ქსელი სეგმენტებად დავყოთ, შეგვიძლია

ბოლო ბაიტში თანრიგების ნაწილი ქსელის მისამართს დავუთმოთ. მაგალითად, ავირჩიოთ შემდეგი ნიღაბი:

11111111.11111111.11111111.11110000

ანუ

255.255.255.240

შედეგად, ერთი ლოკალური ქსელი დაიყოფა $2^4 = 16$ სეგმენტად, რომელთაგან თითოეულში შესაძლებელი იქნება $2^4 - 2 = 14$ კომპიუტერის განთავსება.

შევნიშნოთ, რომ ზოგჯერ ნიღაბს სხვა სახითაც წარმოადგენენ, კერძოდ, მის ორობით გამოსახულებაში არსებული ერთიანების რიცხვით. მოცემული შემთხვევისთვის ნიღბის წარმოდგენის კომპაქტურ ფორმას ექნებოდა რიცხვ 28-ის სახე.

მოვიყვანოთ IP ოქმში მიღებული ზოგიერთი შეთანხმებაც:

თუ მისამართი მხოლოდ ნულებისაგან შედგება, იგულისხმება, რომ მითითებულია იგივე კვანძი, რომელმაც მოახდინა პაკეტის გენერირება;

მისამართში ფიგურირებს მხოლოდ ერთიანები – პაკეტი უნდა მიუვიდეს მოცემულ ლოკალურ ქსელში არსებულ ყველა კომპიუტერს კომუნიკაციის ასეთ ხერხს ეწოდება შეზღუდული ფართომაუწყებლობა (limited broadcast);

თუ ერთიანებითაა შევსებული IP მისამართის მხოლოდ მეორე ნაწილი, ხოლო პირველში მითითებულია კონკრეტული ქსელის მისამართი, მაშინ პაკეტი გადაეგზავნება მითითებულ ქსელში არსებულ ყველა კომპიუტერს. ასეთ შემთხვევაში საქმე გვაქვს „სუფთა“ ფართომაუწყებლობასთან (broadcast).

დასასრულ, კომპიუტერის ქსელში ჩართვა შესაძლებელია გახორციელდეს მისთვის სტატიკური IP მისამართის მიკუთვნების გარეშეც – დინამიკურად, რაზეც ზრუნავს **DHCP** ოქმი): ამ ქვექსელისათვის გამოყოფილი და მოცემულ მომენტში თავისუფალი მისამართების

„საწყობიდან“ მას დროებით, სეანსის დასრულებამდე, გამოეყოფა ერთი ასეთი მისამართთაგანი.

ლიტერატურა

1. გ. ღვინეფაძე. WEB-დაპროგრამება. HTML. სახელმძღვანელო. თბილისი. “ტექნიკური უნივერსიტეტი”. 2005. ISBN 99940-14-59-5. http://gtu.ge/books.php/
2. http://www.intuit.ru/department/internet/html5fwcd/0/
3. გ. ღვინეფაძე. WEB-დაპროგრამება. Javascript. სახელმძღვანელო. თბილისი. “ტექნიკური უნივერსიტეტი”. 2009. ISBN 99940-14-80-3. http://gtu.ge/books.php/ 681.3(06) /203
4. Освой самостоятельно JavaScript за 24 часа. Майкл Монкур,
5. http://www.webmasterwiki.ru/jQuery
6. WEB-ტექნოლოგიების სტანდარტების საიტი http://www.w3schools.com

შინაარსი