გ. ღვინეფაძე

WEB-დაპროგრამება

I ნაწილი

HTML

ტექნიკური უნივერსიტეტი

საქართველოს ტექნიკური უნივერსიტეტი

გ. ღვინეფაძე

WEB-დაპროგრამება

I ნაწილი

HTML

დამტკიცებულია
სტუ-ს
სასწავლო-
მეთოდური
საბჭოს მიერ

თბილისი
2007

უაკ 681.3.06

განხილულია ჰიპერტექსტებთან მუშაობის სპეციალიზებული ენა HTML - ბაზისური საშუალება WEB-გვერდებისა და საიტების შესაქმნელად.

განკუთვნილია ინფორმატიკის სპეციალობათა შემსწავლელი სტუდენტებისა და ამ საკითხებით დაინტერესებული პირებისათვის.

რეცენზენტები: პროფ. ო. ნატროშვილი,
ასოც. პროფ. თ. სუხიაშვილი

ISBN 99940-14-59-5

H T M L

შესავალი

ინტერნეტში ინფორმაციის განთავსებისთვის საჭირო გახდა ახალი საშუალებების შემუშავება.

გლობალურმა ქსელმა არა მარტო დააკავშირა ერთმანეთთან ადამიანები მსოფლიოს ყველა კუთხიდან, არამედ აქცია ისინი ე. წ. ინტერნეტ-ტექნოლოგიების მომხმარებლებად. აღნიშნული ტექნოლოგიები უფრო ფართო გამოყენების ობიექტადაც კი იქცა, ვიდრე მათი უშუალო დანიშნულება გახლავთ. მაგალითად, ჩვენ შეგვიძლია ინტერნეტში ჩართვის გარეშეც გამოვიყვანოთ ბროუზერში, ვთქვათ, კომპაქტ-დისკოზე ჩაწერილი ინფორმაცია.

ინტერნეტ-ტექნოლოგიებიდან, პირველ ყოვლისა, შევისწავლით **HTML**-ს **(HyperText Markup Language) –** *ჰიპერტექსტის გაწყობის ენას.*

<u>ჰიპერტექსტი</u> განიმარტება, როგორც ჩვეულებრივ ტექსტზე უფრო მეტი ინფორმაციული და ფუნქციური მონაცემების შემცველი დოკუმენტი.

საერთოდ, როცა ტექნოლოგიებზეა საუბარი, ცხადია, აქ მხოლოდ დოკუმენტის შექმნის მეთოდებს არ გულისხმობენ. დიდ როლს ასრულებს ინფორმაციის გადაცემის მეთოდებიც, მაგრამ ეს უკვე სხვა სპეციალობის თემაა.

HTML გახლავთ ინტერნეტისთვის დოკუმენტების მომზადების ბაზისური ინსტრუმენტი, მაგრამ მისი შესაძლებლობები შეზღუდულია. ამის გამო, დღეისათვის ფართოდ გამოიყენება ჰიპერტექსტის შექმნის სხვა გზებიც *(ზოგიერთ ძირითად საშუალებას შემდგომში შევისწავლით).*

აღვნიშნოთ, რომ ჰიპერტექსტური დოკუმენტისთვის კოდის მომზადება ნებისმიერ ტექსტურ რედაქტორშია შესაძლებელი, თუმცა არსებობს სპეციალიზებული ჰიპერტექსტური რედაქტორებიც. ინფორმაციის გამოყვანა საჭირო სახით კი ხდება ბროუზერებსა ან იმავე სპეციალიზებულ რედაქტორებში.

შევნიშნოთ, რომ ჰიპერტექსტის კორექტირება შესაძლებელია ბროუზერის მეშვეობითაც – უმეტესწილად **Web**-ფურცლის კოდი **View→Source** ბრძანებით გამოგვყავს **Notepad** ტექსტურ რედაქტორში *(იყენებენ **File→Edit in MS Front Page** ბრძანებასაც).*

კორექტირების შემდეგ, **Refresh** ღილაკზე დაწკაპუნებით მოდიფიცირებული ფაილი ხელახლა გამოგვყავს ეკრანზე.

ამა თუ იმ ტექსტურ რედაქტორში შექმნილ ფაილს ვიმახსოვრებთ **htm** ან **html** გაფართოებით. შესრულებაზე მისი გაშვება *(მოცემულ შემთხვევაში ბროუზერში ჩატვირთვა)* ხდება ჩვენთვის კარგად ნაცნობი, ოპერაციულ სისტემაში მიღებული ხერხებით.

რაც შეეხება ინტერნეტიდან ფაილების კოპირებას, მათ ბროუზერის მეშვეობით ამა თუ იმ საქაღალდეში ვიმახსოვრებთ სტანდარტული, ასევე კარგად ნაცნობი ხერხით – **File→Save as** ბრძანებით. შესაძლებელია ფურცელზე არსებული ცალკეული ნახატის დამახსოვრებაც **Save Picture as** ბრძანების მეშვეობით. მაგრამ გაცილებით უფრო ხშირად იმახსოვრებენ **Web**-ფურცლის მხოლოდ მისამართს, რომელსაც, როგორც წესი, **Favorites** საქაღალდეში ათავსებენ.

Web – ფურცლის აგებულება

HTML-ფურცლის პირველი ვერსია შეიმუშავა ტიმ ბენერსლიმ 90-იანი წლების დასაწყისში *(წარსულში პოპულარული **Mosaic** ბროუზერისათვის).* მსოფლიო აბლაბუდის – **WWW**-ის პოპულარობამ ეს საქმე სწრაფად წასწია წინ და დღეისათვის ფართოდ გამოიყენება **HTML**-ის ის ვერსია, რომელსაც **Dinamic HTML**-საც უწოდებენ. იგი ეყრდნობა ისეთ ბროუზერებთან თანამშრომლობას, რომელთაც შეუძლია მუშაობა **HTML** კოდში ჩაშენებულ **Applet** და **Script** ელემენტებთან.

აპლეტი (**Applet**) წარმოადგენს პროგრამას, რომელიც **Web**-ფურცლის ჩათვალიერებისას დინამიკურად მიუერთდება **HTML**-კოდს ფაილის სახით.

Script ანუ სცენარი გახლავთ პროგრამა, რომელიც უმეტეს შემთხვევაში დაწერილია **JavaScript**-ის ერთ-ერთ ვერსიაში და წარმოადგენს **Web**-ფურცლის მაფორმირებელ **HTML**-კოდში ჩართულ დამატებით კომპონენტს.

*შენიშვნა: **Script** სცენარების შექმნას შემდგომ შევისწავლით.*

ახლა კი, შევუდგეთ საქმეს. ქვემოთ მოყვანილია მარტივი **Web**-დოკუმენტის მაგალითი. იგი წარმოდგენას იძლევა დოკუმენტის ტიპური სტრუქტურის შესახებ. ავკრიბოთ ეს ჰიპერტექსტი **Notepad** რედაქტორში და ფაილი დავიმახსოვროთ, დავუშვათ, **Strukt.html** სახელით *(ყურადღება მიაქციეთ გაფართოებას!).*

```
<HTML>
<HEAD>
<TITLE>Web-ფურცლის სტრუქტურა</title>
<STYLE> H1, H2, H3, H4 {font-family: AcadNusx;}
H6, A, P  {font-family: LitNusx;}
</style>
<META http-equiv=”Content-Type” content=”text/html; charset=windows-1252”>
<META name=”Author” content=”Amiran Georgadze”>
</head>
<BODY bgcolor=#DFDFDF>
<!-- კომენტარი ფურცლის შესახებ -->
<A name=”top”> ფურცლის დასაწყისი </a> <P>
გადავდივართ ფურცლის <A href=”#bottom”> ბოლოში</a><P>
გადავდივართ ფურცლის <A href=”#middle”><B> შუაში</b></a><P>
<HR>
<H1>სათაური 1</h1>
<H2>სათაური 2</h2>
<H3>სათაური 3</h3>
<H4>სათაური 4</h4>
<H5>სათაური 5</h5>
<H6>სათაური 6</h6>
<HR>
<P>აქ მდებარეობს ფურცლის შუა ნაწილი <A name=”middle”></a>
```

<H1>სათაური 1**</h1>**
<H2>სათაური 2**</h2>**
<H3>სათაური 3**</h3>**
<H4>სათაური 4**</h4>**
<H5>სათაური 5**</h5>**
<H6>სათაური 6**</h6>**
<HR>
<A name="bottom"><P>
გადავდივართ ფურცლის **<A href="#top">** დასაწყისში****
</body>
</html>

ლისტინგში მოყვანილი ჰიპერტექსტი წარმოადგენს ცალკეული ელემენტების კრებულს.

ელემენტი გახლავთ **HTML**-ის კონსტრუქცია – ერთგვარი კონტეინერი, რომელიც შეიცავს ამა თუ იმ წესით დასაფორმატებელ *(ან რაიმე სხვა გზით დასამუშავებელ)* მონაცემებს.

ელემენტები ერთმანეთისაგან გამოიყოფა კუთხოვან ფრჩხილებში ჩასმული მარკერების, ანუ **ტეგების** (**tag**) მეშვეობით. ზოგჯერ ბოლო ტეგს არც იყენებენ - მის როლს მომდევნო ელემენტის საწყისი ტეგი ასრულებს. ასეთი შემთხვევების რიცხვი შედარებით ცოტაა. კიდევ უფრო იშვიათად, მაგრამ ხდება, რომ ზოგიერთ ელემენტს საერთოდაც არ გააჩნია ბოლო ტეგი.

სასურველია, საწყისი და საბოლოო ტეგები ერთმანეთისაგან დიდი და პატარა ასოებითაც განვასხვაოთ.

მოვიყვანოთ შესაბამისი ტეგების გარემოცვაში მოქცეული სათაურ-ელემენტის მაგალითი:

<**TITLE**> რწყილი და ჭიანჭველა </**title**>

Web-ფურცლის სტრუქტურიდან გამომდინარე, თანმიმდევრულად განვიხილოთ მისი ელემენტები.

<**HTML**> </**html**>

ყოველი **Web**-ფურცელი წარმოადგენს ერთმანეთში ჩალაგებული ელემენტების გარკვეულ კონსტრუქციას. <**HTML**> </**html**> "მატრიოშკის" გარსაცმში მოთავსებულია **Web**-ფურცლის ყველა სხვა ელემენტი: **HEAD, BODY** და ა.შ.

აქვე შემოვიტანოთ **ატრიბუტის** ცნება. იგი გახლავთ ელემენტის **პარამეტრი, თვისება** და წარმოგვიდგება ამ ელემენტთან დაკავშირებული, განსაზღვრული სახელის მქონე ცვლადის სახით, რომელსაც დუმილით ან ჩვენ მიერ რაიმე მნიშვნელობა მიენიჭება. ამასთან, ტექსტური მნიშვნელობები, რიცხვითისაგან განსხვავებით, ორმაგ ბრჭყალებში ჩაისმება.

ატრიბუტები თავსდება საწყისი ტეგის შიგნით და ერთმანეთისაგან შუალედებით გამოიყოფა.

ატრიბუტები აქვს <**HTML**> </**html**> ელემენტსაც (**version, lang** და **dir**), თუმცა მათ იშვიათად იყენებენ.

<HEAD> </head>

ეს ელემენტი გახლავთ **Web**-ფურცლის პირველი ნაწილი – მისი თავი. იგი შეიცავს **TITLE** ელემენტს, დასაშვებია მასში მრავალი სხვა ელემენტის მოთავსებაც, შეიძლება დახასიათდეს **lang** და **dir** ატრიბუტებით.

HEAD ელემენტში განსაზღვრულ პარამეტრებს ძალა აქვს მთელი **Web**-ფურცლისათვის.

<TITLE> </title>

ამ ელემენტში შეტანილი ტექსტი ფიგურირებს არა დოკუმენტში, არამედ ბროუზერის ფანჯრის სათაურის უბანში.

ეს ტექსტი ხშირად გამოიყენება **WWW**-ში დოკუმენტების ძიებისას. ამიტომ საჭიროა იგი ზუსტად ასახავდეს დოკუმენტის დანიშნულებას.

ზემოთ კოდში ჩვენ ფურცლისთვის სათაურად ავირჩიეთ ***Web**-ფურცლის სტრუქტურა.*

<STYLE> </style>

ვხედავთ, რომ იმავე კოდში სტილები დავუნიშნეთ **H1, H2, H3, H4, H6, A** და **P** ელემენტებს.

ამ დანიშვნების გარეშე მათ *(ისევე, როგორც **H5** ელემენტს)* სტილები დუმილით დაენიშნებოდა. ამრიგად, **<STYLE>**-ელემენტის გამოყენება აუცილებელი არ გახლავთ, მაგრამ ხშირად სასურველია.

<META>

ეს ელემენტი მხოლოდ სამსახურებრივი ინფორმაციის დასაფიქსირებლად გამოიყენება. მას არ სჭირდება ბოლო ტეგი, რადგანაც იგი არ შეიცავს **Web**-ფურცელზე გამოსაყვან ტექსტს.

META-ელემენტში აუცილებლად უნდა ფიგურირებდეს ორი ძირითადი ატრიბუტი, რომლებიც მიგვითითებს **META**-მონაცემების ტიპსა და შემცველობაზე.

მოვიყვანოთ **META-**მონაცემების მაგალითები (*შევნიშნავთ, რომ ეს მონაცემები უშუალოდ **Web**-ფურცელზე არ აისახება).*

- **Web**-ფურცლის ავტორის სახელი:
  name = "**Author" content** = "სახელი, გვარი".
- **Web**-ფურცლის მოსაძებნი საკვანძო სიტყვები:
  name = "Keywords" content = "სიტყვა-1, სიტყვა-2, სიტყვა-3 **. . ."**
- **Web**-ფურცლის მოკლე შინაარსი:
  name = "Description" content = "ანოტაცია"
- **Web**-ფურცლის ტიპის და მახასიათებლების აღწერა:
  name = "Content-Type" content = "ტიპი, მახასიათებლები"

მაგალითად, **content = “text/html”**

- **HTML**–რედაქტორის ჩვენება, რომელშიც შეიქმნა მოცემული დოკუმენტი:
  name = “Generator” content = “HTML-რედაქტორის სახელი**”**
- დოკუმენტის ვარგისიანობის თარიღი:
  name = “Expires” content = “თარიღი”
- ელექტრონული ფოსტის მისამართი:
  name = “Reply-to” content = “სახელი**@**მისამართი”

Web-ფურცლის შესახებ ამ დამატებითი ინფორმაციის გამოყენება მომხმარებელს შეუძლია რომელიმე პროგრამის მეშვეობით.

თუ **name** ატრიბუტი შეცვლილია **http-equiv** ატრიბუტით, მაშინ ამ ინფორმაციით სარგებლობს სერვერი, რათა დამატებითი ველები შექმნას მოთხოვნების შესასრულებლად.

META-ელემენტი შეიძლება **URL**-საც შეიცავდეს:
URL = “**http : //** მისამართი”

<META> ელემენტების გამოყენება სათაურის სექციაში სავალდებულო არ გახლავთ, მაგრამ სასურველია. მისი მეშვეობით, მაგალითად, შეიძლება ავტომატურად განისაზღვროს **Web**-ფურცლის კოდირების სახე:

<META http-equiv = “Content-Type” content = “text/html; charset=windows-1252”>

მეტა-ელემენტებში მოცემული ინფორმაცია – ე.წ. პროფილი შეიძლება ფაილში შევინახოთ და საჭიროების დროს შემდეგნაირად მივუერთოთ **Web**-ფურცელს:

<HEAD profile=”URL”>

<BODY> <body>

სწორედ, ეს ელემენტი შეიცავს ეკრანზე გამოსაყვან ჰიპერტექსტს. მის საწყის ტეგში განსაზღვრული ატრიბუტების მნიშვნელობები მთელ ტექსტზე ვრცელდება. კერძოდ, ფონისათვის გამოყენებული ფაილის მისამართი ასე მოიცემა:

background=”ფაილის მისამართი”

ფონად შეიძლება გამოყენებულ იქნეს ერთი ტონალობის ფერიც, რომელიც განისაზღვრება:

ა) შესაბამისი სიტყვით ინგლისურ ენაზე:
bgcolor = “ფერი”, მაგალითად:
bgcolo r= “red”

ბ) სამი ორთანრიგა თექვსმეტობითი რიცხვით:
bgcolor = “**#RRGGBB**”
ეს რიცხვები გვიჩვენებს მოცემული ფერის შემადგენელი წითელი (**R**), მწვანე (**G**) და ლურჯი (**B**) ფერების ინტენსივობებს.
მაგალითად, იგივე წითელი ფერისათვის გვექნებოდა:
bgcolo = “**#FF0000**”

საინტერესოა, რომ ფონისათვის შეიძლება ერთდროულად გამოვიყენოთ ორივე საშუალება. თუ, ვთქვათ, ამა თუ იმ მიზეზით ნახატი ვერ მოიძებნა, მაშინ იმუშავებდა მეორე ვარიანტი.

Web-ფურცელზე მოცემული ინფორმაციის უკეთ აღსაქმელად ფერები შეიძლება შევურჩიოთ:

- ტექსტს – **text = "#RRGGBB",**
- ჰიპერდაყრდნობებს – **link = "#RRGGBB",**
- ჩათვალიერებულ ჰიპერდაყრდნობებს – **vlink = "#RRGGBB",**
- ბოლოს ნანახ ჰიპერდაყრდნობას – **alink = "#RRGGBB".**

<!-- კომენტარები -->

ტექსტი, რომელიც შეტანილია ამ ელემენტის შიგნით, ბროუზერის მიერ ეკრანზე არ გამოიტანება. იგი გვეხმარება, გავერკვეთ კოდის ცალკეული ფრაგმენტების დანიშნულებაში.

მაგალითად: <!-- ცხრილის გამოყვანის დასაწყისი -->

<H3> </h3>

HTML იყენებს ექვსი დონის სათაურს. სათაურის ელემენტში ნომერი *(მოცემულ შემთხვევაში რიცხვი 3)* მიგვითითებს, თუ რომელი დონის სათაურად უნდა იქნეს გამოტანილი ეკრანზე ტექსტი.

საწყის ტეგში შეიძლება გამოვიყვანოთ ტექსტის განლაგების განმსაზღვრელი **align** ატრიბუტიც, რომელმაც შეიძლება მიიღოს ერთ-ერთი ამ მნიშვნელობათაგანი:

"**left**", "**center**", "**right**".

<HR>

ჰორიზონტალური ხაზი (**horisontal rule**) ის ელემენტი გახლავთ, რომელიც მეორე ტეგს არ საჭიროებს. მისი მეშვეობით შესაძლებელია **Web**-ფურცელი დავყოთ პირობით ნაწილებად.

ამ ელემენტისათვის გათვალისწინებულია ატრიბუტები:

size = სისქე პიქსელებში (*რიცხვები ბრჭყალებს არ საჭიროებს*)

width = სიგრძე პიქსელებში

ან

width = სიგრძე პროცენტებში % *(ბროუზერის ფანჯრის სიგანესთან მიმართებაში)*

color = "ფერი**"**

გაითვალისწინება **align** ატრიბუტიც. მისთვის შესაძლებელია "**left**", "**center**", "**right**" და "**justify**" მნიშვნელობების მინიჭება.

**<A> **

A ელემენტს ორი განსხვავებული, მაგრამ ერთმანეთთან კავშირში მყოფი როლი ეკისრება:

ა) ***ჭდის შექმნა***. როცა **Web**-ფურცელი დიდი ზომისაა, საჭირო ხდება სწრაფად მოვძებნოთ მისი ესა თუ ის განყოფილება. ამ მიზნით, უპირველეს ყოვლისა, უნდა მოვნიშნოთ განყოფილების დასაწყისი, რისთვისაც შესაბამის სტრიქონში ვქმნით ჭდეს:

<**A name** = "ჭდე"> *აქ შესაძლებელია დამხმარე ტექსტის გამოყვანაც* ****
ჭდე შეიძლება გამოვიყენოთ სხვა ფურცელზე გადასვლის დროსაც.

ბ) ***ჰიპერდაყრდნობის შექმნა.*** ჭდით მონიშნულ ფრაგმენტზე ან სულაც სხვა **Web**-ფურცელზე გადასასვლელად შეიძლება შევქმნათ ჰიპერდაყრდნობები:

<**A href** = "#ჭდე"> *მოცემული ჭდით მონიშნულ განყოფილებაზე გადასასვლელად დააწკაპუნეთ აქ* </**a**>

<**A href** = "**Web**-ფურცლის **URL**-მისამართი" > *ამა და ამ **Web**-ფურცელზე გადასასვლელად დააწკაპუნეთ აქ* </**a**>.

<**A href** = "**Web**-ფურცლის **URL**-მისამართი#ჭდე"> *საჭირო **Web**-ფურცლის ჭდით მონიშნულ უბანზე გადასასვლელად დააწკაპუნეთ აქ* </**a**>.

<u>ტექსტის დაფორმატება</u>

ჩვენ მიერ ტექსტურ რედაქტორში აკრეფილი ტექსტის დაფორმატების სპეციფიკას ბროუზერი არავითარ ყურადღებას არ აქცევს – მას ეს ტექსტი ეკრანზე გამოჰყავს, როგორც თითო შუალედით დაცილებული სიტყვების ნაკადი.

ცხადია, სასურველია, **Web**-ფურცელს უფრო გამომსახველი სახე მივცეთ. ამ მიზნის მისაღწევად **HTML** ენაში გათვალისწინებულია დაფორმატების სპეციალური საშუალებები, რომლებიც ერთნაირად გამოიყენება ტექსტურ რედაქტორებში ჰიპერტექსტის მომზადების დროს და, ფაქტობრივად, ერთნაირ ეფექტს გვაძლევს სხვადასხვა ბროუზერში ტექსტის ეკრანზე გამოტანისას.

დაფორმატების საშუალებები ისევ ელემენტების სახით წარმოგვიდგება. გარდა ამისა, დამატებითს შესაძლებლობებს გვაწვდიან ე.წ. <u>**სტილების ცხრილები**</u> *(მათ ქვემოთ განვიხილავთ).*

გავეცნოთ ყველაზე უფრო ხშირად გამოყენებად დაფორმატების ელემენტებს:

<P> </p>

აბზაცის (**paragraph**) ელემენტის ბოლო ტეგს ხშირად არც იყენებენ. მის როლს ითავსებს მომდევნო აბზაცის საწყისი ტეგი. მაგალითად:

<**P**> *პირველი აბზაცის ტექსტი.*
<**P**> *მეორე აბზაცის ტექსტი.* </**p**>

აბზაცის ელემენტის საწყის ტეგში შეიძლება ჩართულ იქნეს **align** ატრიბუტი:
<**P align = "center"**> *აბზაცის ტექსტი*

ამ ელემენტს აქვს მხოლოდ საწყისი ტეგი. მის შემდეგ განლაგებული ტექსტი ეკრანზე ახალი სტრიქონიდან გამოდის.

<NOBR> </nobr>

ელემენტში მოქცეული ტექსტი ეკრანზე ერთი სტრიქონის სახით გამოდის. თუ იგი მასზე ვერ ეტევა, ვიყენებთ ჰორიზონტალური გადახვევის ზოლს.

<PRE> </pre>

ელემენტში მოთავსებული ტექსტი ბროუზერში ინარჩუნებს საწყის დაფორმატებას. ძირითადად იყენებენ პროგრამების ლისტინგების ან იმგვარი დოკუმენტების გამოსაყვანად, რომელთა ხელახლა დაფორმატებას შეუძლია დაამახინჯოს მათი შინაარსი.

<CENTER>

ეს ელემენტი აკეთებს იმასვე, რასაც **align** = "**center**" ატრიბუტის მნიშვნელობა.

<B>

ტექსტის გამსხვილების ამ ეფექტს ძალზე ხშირად მიმართავენ.

<BIG> </big> და <SMALL> </small>

ეს ელემენტები ერთი დონით ადიდებს (ამცირებს) შრიფტის ზომას.

<I> </i>

ტექსტი გამოიყოფა კურსივის მეშვეობით *(უკეთდება დახრა).*

<S> </s>

გამოჰყავს გადახაზული ტექსტი.

<U> </u>

განკუთვნილია ტექსტის ხაზგასმისათვის.

და

ეს ელემენტები ტექსტს გარდაქმნის ქვედა და ზედა ინდექსებად. ზოგჯერ მათ იყენებენ მთელი აბზაცის პატარა შრიფტში გამოსაყვანადაც.

<TT> </tt>

ამ ელემენტში შეტანილი ტექსტი გადაიყვანება ტელეტაიპის შრიფტში – სიმბოლოები ერთი სიგანის ხდება.

<BASEFONT size=შრიფტის ბაზური ზომა>

ელემენტი განსაზღვრავს შრიფტის ბაზურ ზომას მთელი **Web**-ფურცლისათვის.

size ატრიბუტი მნიშვნელობებს ღებულობს 1–7 დიაპაზონში. დუმილით გათვალისწინებულია 3-ის ტოლი ზომა.

ამ ელემენტის სხვა ატრიბუტები **FONT** ელემენტის ატრიბუტების ანალოგიურია.

<**FONT**> </**font**>

განსაზღვრავს შრიფტის მოხაზულობის სახეს, ზომასა და ფერს.

დავიწყოთ ფერით. იგი მოიცემა ატრიბუტით

color = "ფერი" *(იხ. წინა მასალები)*

შრიფტის სახის განმსაზღვრელი ატრიბუტი შეიძლება შეიცავდეს შრიფტების სიას. მაგალითად:

face = "**Arial, Verdana, Tahoma**"

თუ კომპიუტერზე არც ერთი ამ შრიფტთაგანი არ აღმოჩნდა დაყენებული, გამოიყენება დუმილით გათვალისწინებული თითო-თითო შრიფტი ტოლი სიგანის და პროპორციული შრიფტების ვარიანტებისათვის.

რაც შეეხება ზომას, იგი შეიძლება მოცემული იყოს აბსოლუტური მნიშვნელობით: **size**=1–7

ანდა, როგორც ფარდობითი, ზემოთ განხილული ბაზური ზომის მიმართ:

size= +n

size= -n

ცხადია, **n** ისე უნდა შეირჩეს, რომ **size**-ს მნიშვნელობა 1–7 დიაპაზონს არ გასცდეს.

FONT ელემენტს სათაურებისათვისაც იყენებენ, რათა შესაძლებელი გახდეს, ვთქვათ, შრიფტის ფერის განსაზღვრაც. **FONT** ელემენტს ასეთ შემთხვევაში გამოიყენებენ სხვა ელემენტებთან (**CENTER, B, I** და ა.შ.) კომბინაციებში.

<u>დაფორმატების დამატებითი საშუალებები სპეციფიკური ტექსტებისათვის</u>

ქვემოთ განხილული ელემენტები განკუთვნილია სპეციფიკური შინაარსის მქონე ტექსტური ფრაგმენტების ეკრანზე ასახვისათვის. აქ ხაზი უნდა გაესვას შემდეგ გარემოებას – სხვადასხვა ბროუზერი თითოეული ელემენტისათვის დაფორმატების საკუთარ ვარიანტს ირჩევს, თუმცა ძირითადი მიზანი მიღწეულია – ყველა, ვთქვათ, ციტატა ეკრანზე გამოდის ერთნაირი, ტექსტის სხვა ნაწილებისგან განსხვავებული სახით.

<**BLOCKQUOTE**> </**blockquote**>

ტექსტი წარმოგვიდგება ციტატის სახით – აბზაცი დაიძვრება. სხვა მხრივ, ტექსტი არ იცვლება. თუ საჭიროდ მიგვაჩნია ციტატის ბრჭყალებში ჩასმაც, ჩვენ თვითონ უნდა შევიტანოთ ისინი საწყის ტექსტში.

ელემენტი აუცილებელია მეორე ტეგით დაბოლოვდეს. საწყის ტეგში შესაძლებელია ციტირების წყაროს მითითებაც **URL** სახით:

cite=”**URL**”

<Q> </q>

წინა ელემენტებისაგან იმით განსხვავდება, რომ ციტატა იქმნება არა ცალკეული აბზაცის სახით, არამედ მოცემულ აბზაცში.

მოვიყვანოთ კიდევ ზოგიერთი, შედარებით ნაკლებმნიშვნელოვანი ელემენტების სია მათი დანიშნულების შესახებ მოკლე ინფორმაციით:

<CODE> </code> - განკუთვნილია პროგრამული ფრაგმენტებისათვის,
<VAR> </var> - ცვლადების აღსანიშნავად,
<SAMP> </samp> - მაგალითის სახის მოყვანილი მონაცემებისათვის,
<KBD> </kbd> - მომხმარებლის მიერ კლავიატურიდან შეტანილი ტექსტის აღსანიშნავად,
<ABBR> </abbr> - აბრევიატურებისათვის,
<ACRONYM> </acronym> - აკრონიმებისათვის.

კვლავ სტილის შესახებ. სტილების ცხრილები

ჩვენ უკვე შევისწავლეთ **<STYLE> </style>** ელემენტი, რომელიც თავსდებოდა **HEAD** ელემენტში და მოდიფიცირებას უკეთებდა დუმილით გათვალისწინებულ სტილებს ამა თუ იმ დონის სათაურებისა თუ **Web**-ფურცლის სხვა შემადგენელი ნაწილებისთვის. პრაქტიკაში ხშირად მოითხოვება, **Web**-ფურცელზე კონკრეტული უბანი გადავაფორმატოთ **HEAD** უბანში განსაზღვრული ან დუმილით გათვალისწინებული სტილისაგან განსხვავებული სახით. ამ მიზანს ემსახურება სტანდარტული **style** ატრიბუტი, რომელიც გამოიყენება უშუალოდ რომელიმე კონკრეტული უბნისათვის *(აბზაცისათვის):*

<P style = “font-size: 10pt; font-style: italic; color: blue”>

როგორც ვხედავთ, მოცემულ შემთხვევაში სტილის ცალკეული ატრიბუტების მნიშვნელობების განსაზღვრისათვის განსხვავებული წესები გამოიყენება.

მოვიყვანოთ **STYLE** ელემენტის გამოყენების შედარებით რთული მაგალითი:

<STYLE>
H1 {border-width: 1; border: groove; text-align: center; color: green}
H2 {color: maroon; font-style: italic}
P {color: violet; font-size: 16pt}
CODE {font-family: LitNusx; font-size: 14pt }
P CODE {font-weight: bold; background: blue }
</style>

განსაკუთრებული ყურადღება მივაქციოთ **P CODE** კონსტრუქციისათვის სტილის განსაზღვრის წესს. ამ სტილს კოდისათვის ბროუზერი მაშინ გამოიყენებს, როცა ეს კოდი ჩადგმული იქნება აბზაცში, როგორც მისი შიდა ელემენტი. მაშინ იმ ატრიბუტების მნიშვნელობები, რომლებიც უშუალოდ **P CODE** კონსტრუქციის კოდისათვის განსაზღვრული არ გახლავთ, მემკვიდრეობით გადმოვა ცალკე **P**-სა და ცალკე **CODE**-სათვის განსაზღვრული ატრიბუტებიდან.

ამრიგად, ვხედავთ, რომ **STYLE** ელემენტს შეუძლია ატრიბუტების მნიშვნელობის ერთდროულად განსაზღვრა ყველა აბზაცისათვის ან რომელიმე დონის ყველა სათაურისათვის და ა.შ., ხოლო ატრიბუტი **style** ადგილზე ახდენს რომელიმე კონკრეტული სათაურის თუ აბზაცის გადაფორმატებას.

სტილების კასკადური ცხრილები

სტილების გამოყენებაში შემდგომი ნაბიჯი იქნა გადადგმული ე.წ. **სტილების ცხრილების** შემოღებით. მათი არსი ის გახლავთ, რომ ტექსტის ამა თუ იმ ელემენტისათვის ჩვენ შეგვიძლია შევქმნათ სხვადასხვა სახელის მქონე სტილები, ანუ კლასები, რითაც „მენიუ“ უფრო მრავალფეროვანი გახდება.

HTML მოითხოვს, რომ სისტემას ეცნობოს სტილების ცხრილების გამოყენების შესახებ, რისთვისაც **HEAD** უბანში უნდა ჩავრთოთ ასეთი მეტა-განსაზღვრება:

<**META http-equiv = “Content-Style-Type” content = “text/css”**>

შედეგად ბროუზერი მიიღებს ინფორმაციას, სტილების განსაზღვრის რომელი ენა გამოიყენება. **css** ამ შემთხვევაში აღნიშნავს ე.წ. **სტილების კასკადურ ცხრილს** (**Cascading Style Steets**). იგი ერთდროულად სტანდარტიცაა და ენაც, რომელიც აფართოებს ტრადიციული **HTML**-ის შესაძლებლობებს. დღეისათვის არსებობს ორი სპეციფიკაცია: **css**1 და **css**2, რომლებიც გაცილებით მეტ ატრიბუტებს (*აქ მათ **თვისებები** ეწოდება*) შეიცავს, ვიდრე ეს უშუალოდ **HTML**-ში არის გათვალისწინებული.

სტილების ცხრილის შამბლონი ამგვარი სახის არის:

ელემენტი.სტილის-სახელი {თვისება1: მნიშვნელობა; თვისება2: მნიშვნელობა;. . .}

აღვნიშნოთ, რომ ასეთი კონსტრუქციები ობიექტზე ორიენტირებული დაპროგრამების ენებისთვის არის დამახასიათებელი და, ცხადია, რომ სინტაქსსაც რამდენადმე განსხვავებული სახე აქვს.

მოვიყვანოთ სახელდებული სტილის შექმნის მაგალითი:

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

პირველი დონის რომელიმე სათაურისათვის ადგილზე **red1** სტილის გამოსაყენებლად ვსარგებლობთ კონსტრუქციით:

<**H1 class = “red1”**> *სათაურის ტექსტი* </**h1**>

მოცემული შესაძლებლობის მქონე სტილის ზოგადი სახელია *კლასი*. ჩვენ შეგვიძლია შევქმნათ სხვა კლასებიც: **red2**, **red3** და ა.შ., რომელთა მეშვეობითაც შეგვეძლება სხვადასხვა სტილი განვუსაზღვროთ ერთსა და იმავე ელემენტს *(მოცემულ შემთხვევაში პირველი დონის სათაურს).*

იმისათვის კი, რომ სასურველ სტილში განსხვავებული ელემენტები დავაფორმატოთ *(მაგალითად, სათაურები და ჩვეულებრივი ტექსტი)*, **STYLE** ელემენტში

კონკრეტული სტილის სახელს წინ უნდა წავუმძღვაროთ # სიმბოლო. შედეგად შევქმნით ე.წ. *უნივერსალურ კლასს.* მაგალითად:

<STYLE type = "text/css">
#steel {color: RGB(155, 180, 190)}
</style>

ხოლო ადგილზე უნდა გამოვიყენოთ **id** ატრიბუტი:

<H2 id = "steel"> *სათაური* **</h2>**
<P id = "steel"> *აბზაცის ტექსტი* **</p>**

სტილების ცხრილები **Web**-ფურცლისათვის ე.წ. **css**-ფაილიდანაც შეიძლება გამოვიძახოთ, თანაც ერთმანეთის მიყოლებით *(სწორედ, ამიტომ უწოდეს მათ კასკადური).* მაგალითად, შეიძლება ერთმანეთს დაედოს **Web**-ფურცლის ავტორის, სერვერის და მომხმარებლის სტილები. ყოველი ახლად გამოძახებული სტილის ცხრილი კორექტირებას უკეთებს წინას.

css-ფაილის შემცველობა შეიძლება იყოს ასეთი:

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;}

ფურცელთან ფაილის მიერთება კი ხდება **HEAD** სექციაში **LINK** ელემენტის ჩართვით:

<LINK href = "*ფაილის-სახელი*.**css" rel = "stylesheet" type = "text/css"**>

<DIV> და <SPAN> ელემენტები

ეს ორი ელემენტი სპეციალურად სტილების გამოყენებისათვის არის შექმნილი. პირველი ასრულებს კონტეინერის როლს მასში მყოფი ყველა ელემენტისათვის, რომელთაც შესაბამისი სტილი განესაზღვრება.

მეორე შეიძლება გამოვიყენოთ მოცემული ელემენტის *(მაგალითად, აბზაცის)* ნაწილისთვის სტილის მოდიფიცირების მიზნით:

ა) **<HEAD>**
<STYLE type = "**text/css**"> **DIV**.*სახელი { თვისება: მნიშვნელობა; . . .}*
</style>
</head>
<BODY>
<DIV class = "*სახელი***">**
<H1> *სათაური* **</h1>**
<P> *პირველი აბზაცი*
<P> *მეორე აბზაცი*
** **
</body>

ბ) **<HEAD>**
<STYLE type = "text/css">

SPAN.*სახელი { თვისება: მნიშვნელობა; . . .}*
</style>
</head>
<BODY>
<H1> *ქვესათაური* **</h1>**
<P> <SPAN class = "სახელი"**>** *ტექსტი 1* **** *ტექსტი 2* **</p>**
</body>

ამრიგად, სტილი შეიძლება განესაზღვროს: დოკუმენტის ერთ ელემენტს *(მაგალითად, აბზაცს)*; ყველა ერთსახელიან ელემენტს *(დავუშვათ, მეორე დონის სათაურებს და ჩვეულებრივ ტექსტს);* დაბოლოს, **css**-ფაილის გამოყენებით – სხვადასხვა დოკუმენტების რიგს.

სიები

სიებს (**list**) ჩვენ ჯერ კიდევ **Word**-იდან ვიცნობთ. აქაც ვიყენებთ უნომრო და დანომრილ სიებს.

შაბლონებს აქვს შემდეგი სახე:

<**UL**> (**unordered list**)
<**LH**> **უნომრო სია**
<**LI**> სიის პუნქტი
<**LI**> სიის პუნქტი
<**LI**> სიის პუნქტი
</**ul**>

<**OL type**="**I**"> (**ordered list**)
<**LH**> **დანომრილი სია**
<**LI**> პირველი პუნქტი
<**LI**> მეორე პუნქტი
<**LI**> მესამე პუნქტი
</**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 1-სგან 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 ვანიჭებთ შესაბამის მნიშვნელობას. მაგალითად:

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

აღსანიშნავია, რომ თითოეული ნომრიანი პუნქტი, თავის მხრივ, შეიძლება შეიცავდეს დაუნომრავი ქვეპუნქტების ნებისმიერ რიცხვს. ასეთი სია იქმნება შემდეგი შაბლონის მეშვეობით:

```
<DL>                                        (definition list)
 <DT> პირველი პუნქტი
   <DD> ვაშლი
   <DD> მსხალი
 <DT> მეორე პუნქტი
   <DD> სტაფილო
 <DT> მესამე პუნქტი
    <DD> არაჟანი
</dl>
```

შეიძლება შევქმნათ უნომრო სიაში ჩალაგებული ნომრიანი სიებიც:

```
<UL>
 <LI> პირველი პუნქტი
   <OL>
     <LI>1.1 ქვეპუნქტი
     <LI>1.2 ქვეპუნქტი
   </ol>
 <LI> მეორე პუნქტი
   <OL>
     <LI>2.1 ქვეპუნქტი
     <LI>2.2 ქვეპუნქტი
     <LI>2.3 ქვეპუნქტი
   </ol>
</ul>
```

ცხრილები

ცხრილებს, გარდა თავისი პირდაპირი დანიშნულებისა, დამატებით დიზაინერის ფუნქციებიც ეკისრება. მათი მეშვეობით შეიძლება მწყობრში მოვაქციოთ ფურცლის ნაწილები; გვერდიგვერდ მოვათავსოთ ტექსტი და ნახატი, შევქმნათ ფერადოვანი ხალიჩა და სხვ.

ცხრილის ელემენტების ჩამოთვლა ხდება ზემოდან ქვემოთ და მარჯვნიდან მარცხნივ.

დავხატოთ მარტივი ცხრილი, შემდეგ კი დავწეროთ მისთვის პროგრამული კოდი.

მარტივი ცხრილი

სათაური 1	სათაური 2
უჯრა 1	უჯრა 2
უჯრა 3	უჯრა 4

<**TABLE border=4 cellspacing=3** >
<**CAPTION**> მარტივი ცხრილი </**caption**>
<**TR**><**TH bgcolor = "yellow"**> სათაური 1
<**TH bgcolor = "yellow"**> სათაური 2
<**TR**><**TD**> უჯრა 1
<**TD**> უჯრა 2
<**TR**><**TD**> უჯრა 3
<**TD**> უჯრა 4
</**table**>

განვმარტოთ კოდის შემადგენელი, ჩვენთვის უცნობი დეტალების არსი:

border ატრიბუტი ცხრილის გარე ჩარჩოს სისქეს განსაზღვრავს. შეიძლება მას ნულის ტოლი მნიშვნელობაც მივანიჭოთ. ჩარჩოს დანარჩენი ნაწილის სისქე კი **cellspacing** ატრიბუტით განისაზღვრება. თუ მის მნიშვნელობას ნულის ტოლს ავიღებთ, ჩარჩო დაწვრილდება, მაგრამ მაინც დარჩება ეკრანზე.

<**CAPTION**> ელემენტი *(წარწერა)* ქმნის ცხრილის სათაურს.

TR-ელემენტები სტრიქონებს აგებენ. მათ შიგნით სვეტებისათვის *(შესაძლოა სტრიქონებისთვისაც)* განლაგდება უჯრა-სათაურები (*მოცემულ ცხრილში "სათაური 1" და "სათაური 2"-ის შემცველი უჯრები).* ამ უჯრა-სათაურებს ქმნის **TH**-ელემენტები. დაბოლოს, **TD** ელემენტი აგებს ცხრილის ჩვეულებრივ უჯრას.

ცხრილისათვის შეიძლება გამოყენებულ იქნეს რიგი ატრიბუტებისა.

განვმარტოთ ზოგიერთი ატრიბუტის დანიშნულება.

valign ატრიბუტის "**top", "center"** და **"bottom"** მნიშვნელობების მიხედვით ხდება უჯრაში ტექსტის ზედა, შუა ან ქვედა პოზიციაში გამოყვანა. ამ ატრიბუტს შეუძლია მიიღოს "**baseline"** მნიშვნელობაც. მაგალითად:

<**TR valign="baseline"**> <**TD**> სტრიქონი 1 <**br**> სტრიქონი 2 <**TD**> წიგნი

პირველ უჯრაში გვაქვს ტექსტის 2 სტრიქონი. **valign** ატრიბუტის "**baseline**" მნიშვნელობა უზრუნველყოფს მომდევნო უჯრის შემცველობის ("წიგნი") გამოყვანას წინა უჯრის პირველი სტრიქონის ("სტრიქონი 1") დონეზე.

ატრიბუტების **rowspan=m** და **colspan=n** მეშვეობით შეიძლება უჯრების გაერთიანება ვერტიკალსა და ჰორიზონტალზე.

დასაშვებია უჯრების ზომების პირდაპირი განსაზღვრაც:

width=სიგრძე

height=სიმაღლე

დავწეროთ კოდი შემდეგი ცხრილისთვის:

ცხრილი გაერთიანებული უჯრებით

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

უჯრაში, რომელიც არ შეიცავს ტექსტს, ვითვალისწინებთ ე.წ. უწყვეტი შუალედის სიმბოლოს – ** ,** რითაც თავიდან ვიცილებთ ცხრილის ბადის დამახინჯებას.

<**TABLE border = "4" cellspacing=3 background = "**ნახატის ფაილი">
<**CAPTION**><**I**> ცხრილი გაერთიანებული უჯრებით </**i**></**caption**>
<TR><TH rowspan = "2"> <TH colspan = "2"> სათაური 1
<**TR**><**TH**>სათაური 1.1<**TH**> სათაური 1.2
<**TR**><**TH**>სათაური 2<**TD**> უჯრა 1<**TD**> უჯრა 2
<**TR**><**TH**>სათაური 3<**TD**> უჯრა 3<**TD**> უჯრა 4
</**table**>

სტრიქონების დაჯგუფება <THEAD>, <TFOOT> და <TBODY> ელემენტების მეშვეობით

ეს ხერხი საშუალებას იძლევა, თითოეულ ჯგუფს ცალკე განესაზღვროს სტილი.

ასეთი ცხრილების შექმნისას გავითვალისწინოთ, რომ:

1. ჯერ უნდა ვუჩვენოთ თითო-თითო **THEAD** და **TFOOT** ელემენტი, შემდეგ კი – ერთი ან რამდენიმე **TBODY** ელემენტი *(ცხრილის აგებისას, ცხადია, ბროუზერი **TFOOT** ბლოკს ქვემოთ მოაქცევს).*
2. ყოველი ბლოკი უნდა შეიცავდეს სვეტების ერთნაირ რაოდენობას.

ავაგოთ კოდი შემდეგი ცხრილისთვის:

სათაური 1	სათაური 2
სტრიქონი 1	უჯრა 1.2
სტრიქონი 2	უჯრა 2.2
სტრიქონი 3	უჯრა 3.2
სტრიქონი 4	უჯრა 4.2
სტრიქონი 5	უჯრა 5.2
ქვედა ბლოკი	

<**TABLE border=2**>
<THEAD>
<TR><TD>სათაური 1**<TD>**სათაური 2
<TFOOT>
<TR> <TD>ქვედა ბლოკი**<TD> **
<TBODY>
<**TR**> <**TD**>სტრიქონი 1<**TD**> უჯრა 1.2
<**TR**> <**TD**>სტრიქონი 2<**TD**> უჯრა 2.2
<TBODY>
<**TR**> <**TD**>სტრიქონი 3<**TD**> უჯრა 3.2
<**TR**> <**TD**>სტრიქონი 4<**TD**> უჯრა 4.2
<**TR**> <**TD**>სტრიქონი 5<**TD**> უჯრა 5.2
</**table**>

ვხედავთ, რომ <**TBODY**> ელემენტი ცხრილში რამდენიმეც შეიძლება იყოს *(განსხვავებული დაფორმატების უზრუნველსაყოფად).*

სვეტების დაჯგუფება <COLGROUP> და <COL> ელემენტებით

<**COLGROUP**> ელემენტი საშუალებას იძლევა შევქმნათ ერთნაირი სტილის მქონე სვეტების ჯგუფები. განვიხილოთ მაგალითი:

1-1	1-2	1-3	1-4	1-5
2-1	2-2	2-3	2-4	2-5

<**TABLE border=4**>
<COLGROUP span=1 width = “30” bgcolor = “lime”>
<COLGROUP bgcolor = “yellow”>
<COl span=2 width = “30”>
<COl width = “60”>
<COLGROUP bgcolor = “aqua”>
<Col width = “50”>
<TR><TD> 1-1 **<TD>** 1-2 **<TD>** 1-3 **<TD>** 1-4 **<TD>** 1-5
<TR><TD> 2-1 **<TD>** 2-2 **<TD>** 2-3 **<TD>** 2-4 **<TD>** 2-5
</**table**>

თითოეული **COLGROUP** ელემენტი **span** ატრიბუტის მეშვეობით განსაზღვრავს ჯგუფში იმ სვეტების რაოდენობას, რომლებზეც ვრცელდება აღნიშნულ ელემენტში დანიშნული თვისებები.

დასაშვებია **span** ატრიბუტის გამოყენება არა **COLGROUP** ელემენტში, არამედ მის შიგნით განსაზღვრულ **COL** ელემენტებში. საერთოდ, როცა გამოიყენება თუნდაც ერთი **COL** ელემენტი, **COLGROUP**-ში **span**-ის გამოყენება აზრს კარგავს. მასში სვეტების რაოდენობა განისაზღვრება შემადგენელი **COL** ელემენტების რაოდენობით *(რომელთაგან ზოგი, როგორც აღვნიშნეთ, შეიძლება* ***span*** *ატრიბუტსაც შეიცავდეს).*

COL ელემენტების მეშვეობით შესაძლებელია მოხდეს სვეტისა და სვეტების ჯგუფებისათვის თვისებების ნაწილის მოდიფიცირება.

ცხრილის სტრუქტურის განსაზღვრის შემდეგ ხდება ცალკეული უჯრების შემცველობათა ჩვენება.

<u>სინტაქსი HTML -ში</u>

ახლა, როცა გავეცანით საკმაო მასალას, შეიძლება მწყობრში მოვიყვანოთ ჩვენი ფრაგმენტული ცოდნა **HTML**-ენის სინტაქსისის შესახებ.

ვხედავთ, რომ **HTML**, **HEAD, TITLE** და **BODY** ელემენტების განლაგება ერთნაირად უნდა ხდებოდეს ნებისმიერ ფურცელზე (*გარდა ფრეიმების გამოყენების შემთხვევისა, რასაც ქვემოთ განვიხილავთ. აქ* ***BODY****-ის ნაცვლად გამოიყენება* ***FRAMESET*** *ელემენტი).*

აუცილებელია ვიცოდეთ, თუ რომელი ელემენტი რომელში შეიძლება ჩაიდგეს. კოდის ამ უბანზე პროგრამისტებს საკმაოდ ხშირად მოსდით შეცდომები. მოვიყვანოთ ტიპური შეცდომის შემთხვევა:

<**H**1> *სათაური 1* <**H2**> *სათაური 2* </**h1**> *სათაური 3* </**h2**>

ზოგჯერ ბროუზერი შეცდომას თავად ასწორებს, რაც შესაძლოა, ყოველთვის არც ემთხვეოდეს დამპროექტებლის ჩანაფიქრს.

ზოგიერთი ელემენტი სხვადასხვა ადგილას შეიძლება განლაგდეს. მაგალითად, ცხრილი შეიძლება მოთავსდეს **BODY**-ში, როგორც დამოუკიდებული ელემენტი და შეიძლება ჩაიდგეს აბზაცშიც, როგორც მისი ნაწილი.

სიმბოლოების კოდირება

ინტერნეტში სიმბოლოების ასახვისთვის სტანდარტულ რეჟიმად ითვლება **ISO Latin 1** კოდირების გამოყენება. ბროუზერს პრობლემები არ ექმნება როგორც **MS DOS**-ში, ასევე **Windows**-ში მომუშავე ტექსტური რედაქტორებით მომზადებული ჰიპერტექსტების გამოყვანისას.

მაგრამ, როცა საჭიროა ეროვნული ალფაბეტის გამოყენებაც, მაშინ აუცილებელია ვუჩვენოთ **charset** ატრიბუტი და სწორად განვსაზღვროთ მისი მნიშვნელობა. მაგალითად, რუსული ტექსტებისათვის ფაქტიურად საკმარისია ორი ყველაზე პოპულარული კოდირების *(კირილიცა* ***Windows*** *და კირილიცა* ***KOI8-R****)* გამოყენება.

კოდური ცხრილის ჩვენება ასე ხდება:

charset=windows-1252

charset=КОИ-8

შესაძლოა, ინტერნეტიდან გადმოგზავნილ **Web**-ფურცლის კოდში **charset** ატრიბუტი გამორჩენილი ან არასწორად ნაჩვენები იყოს *(მაგალითად, **Word-**სა და **Front Page Express**-ში ჰიპერტექსტების მომზადებისას ფურცლის კოდს ავტომატურად ემატება **charset** ატრიბუტის მნიშვნელობა, რაც კირილიცის გამოყენების საშუალებას არ იძლევა).* ასეთ შემთხვევაში ინფორმაცია ეკრანზე დამახინჯებულად აისახება. მდგომარეობის გამოსწორება შეიძლება ბროუზერშივე **View → Source** ბრძანებაში შესაბამისი კოდირების არჩევით.

აღსანიშნავია, რომ ბროუზერებს **ISO Latin 1** კოდირების გარდა მუშაობა შეუძლიათ **Unicode 2.0** კოდირების სისტემასთანაც. ამ უკანასკნელში თითოეულ სიმბოლოს ორი ბაიტი ეთმობა, რაც ნაციონალური ალფაბეტების პირდაპირი წარმოდგენის შესაძლებლობას იძლევა *(საჭირო აღარ გახლავთ **charset** ატრიბუტისადმი მიმართვა).* მაგრამ **Unicode 2.0**-ის მასიურ გამოყენებას ხელს უშლის ის გარემოება, რომ ბევრი ტექსტური რედაქტორი მისადაგებული არ გახლავთ ამ კოდთან მუშაობისათვის.

სპეცსიმბოლოების გამოყენება

ნებისმიერი სიმბოლო ჰიპერტექსტში შეიძლება შეტანილ იქნეს რიცხვითი კოდის სახითაც. მაგალითად, დასაშვებია ლათინური **A** სიმბოლო წარმოვადგინოთ &#65 აღნიშვნით. ზოგიერთი სპეცსიმბოლოსათვის კი, კოდის დამახსოვრების გაადვილების მიზნით, იყენებენ ე.წ. **მნემონიკურ კოდსაც:**

კოდი	რიცხვითი კოდი	მნემონიკური კოდი	დასახელება	სიმბოლო
34	**&#34**	**"**	ბრჭყალი	**"**
38	**&#38**	**&**	ამპერსანდი	**&**
60	**&#60**	**<**	ნაკლებობის ნიშანი	**<**
62	**&#62**	**>**	მეტობის ნიშანი	**>**
160	** **	** **	უწყვეტი შუალედი	
167	**&#167**	**§**	პარაგრაფის ნიშანი	**§**
172	**&#172**	**¬**	უარყოფის ნიშანი	**¬**
176	**&#176**	**°**	გრადუსი	**°**
178	**&#178**	**²**	2-ის ხარისხი	**²**
215	**&#215**	**×**	გამრავლ. ნიშანი	*****
247	**&#247**	**÷**	გაყოფის ნიშანი	**/**

ასეთი შესაძლებლობის გამოყენებას განსაკუთრებით აქვს აზრი ისეთი სიმბოლოებისათვის, რომლებიც კლავიატურაზე არ არის ასახული. მაგალითად, ეკრანის პარამეტრების საჩვენებლად იყენებენ კონსტრუქციას: 640**×**480.

ხაზგასმით აღვნიშნოთ, რადგანაც “, <, > და & სიმბოლოები **HTML** კოდისათვის სამსახურებრივი სიმბოლოებია, **Web**-ფურცლის ჩვეულებრივ ტექსტში მათი გამოყენება მხოლოდ ზემოთ მოყვანილი კოდების მეშვეობით უნდა მოხდეს.

ფრეიმები

ფრეიმების შემოღებით კიდევ ერთი ნაბიჯი გადაიდგა წინ – ეკრანი დაიყო უბნებად, თითოეულში **Web**-ფურცლის ჩათვალიერების შესაძლებლობით.

პირველი, რასაც ვაკეთებთ, ეკრანის სტრუქტურის განსაზღვრა გახლავთ. ხდება ვერტიკალზე ან ჰორიზონტალზე მისი უბნებად (ფრეიმებად) დაყოფა, რისთვისაც ვიყენებთ **cols** და **rows** ატრიბუტებს. მათი მნიშვნელობები გამოისახება პიქსელებსა ან პროცენტებში, ეკრანის დარჩენილ ნაწილს კი აღვნიშნავთ * სიმბოლოთი. მაგალითად:

- **cols**=50%, 50% – ეკრანი იყოფა ორ თანაბარ ვერტიკალურ უბნად;
- **cols**=25%, 75% – ეს პროპორცია მრავალ **Web**-ფურცელზე გამოიყენება;
- **rows**=150, 30%, * – ეკრანი იყოფა 3 ჰორიზონტალურ უბნად: ზედას სიმაღლე 150 პიქსელია, შუათანის კი – დარჩენილის 30%.

ეკრანის დაყოფის შესახებ ინფორმაცია თავსდება ფანჯრის სტრუქტურის ამსახველი დოკუმენტის **FRAMESET** ელემენტში, თითოეულ უბანს კი ვავსებთ შინაარსობრივი დოკუმენტებით. ამრიგად, ვიყენებთ ორი: **layout** და **content** ტიპის დოკუმენტებს (ფაილებს).

layout ტიპის დოკუმენტებში **BODY** სექციის ნაცვლად გამოიყენება **FRAMESET** სექცია (ელემენტი), რომლის შიგნითაც უნდა ფიგურირებდეს დაყრდნობა შინაარსობრივ დოკუმენტზე. სასარგებლოა **name** ატრიბუტის მეშვეობით ფრეიმის სახელის ჩვენებაც, რომელიც გამოყენებული იქნება ჰიპერდაყრდნობებში. შედეგად, **FRAME** ელემენტი ასე შეიძლება გამოიყურებოდეს:

<**FRAME src** = **“**ფაილის სახელი.**html” name = “**ფრეიმის სახელი”>

მას შემდეგ, რაც ეკრანზე ყველა ფურცელი ჩაიტვირთება, მომხმარებელს თაგვის მეშვეობით შეუძლია შეცვალოს ფრეიმების საზღვრები. **noresize** ატრიბუტით კი საერთოდ იკრძალება ამ შესაძლებლობის გამოყენება.

scrolling ატრიბუტისთვის **“YES”**, **“NO”** ან **“AUTO”** მნიშვნელობების მინიჭებით შეგვიძლია ნება დავრთოთ/ავკრძალოთ ეკრანზე დოკუმენტის გადახვევა, ან მხოლოდ მაშინ დავუშვათ ეკრანზე ამ ზოლების გამოყვანა, როცა ტექსტი მასზე ვერ ეტევა.

შევნიშნოთ, რომ გადახვევის აკრძალვით ეკრანზე იქმნება ე.წ. **ბანერი**.

frameborder ატრიბუტისთვის “1” ან “0” მნიშვნელობის მინიჭებით ფრეიმი გამოგვყავს ჩარჩოში ან მის გარეშე.

ფრეიმის საზღვრებიდან ტექსტის დასაცილებლად განკუთვნილია ატრიბუტები:

marginheigh = “75”

marginwidth = “10”

(ზომა პიქსელებში მოიცემა)

წარმოვადგინოთ ფრეიმებიანი ფურცლის შექმნის მაგალითი:

```
<HTML>
<HEAD>
<TITLE>FREIMEBI</title>
</head>
<FRAMESET rows="20%, 60%, 20%">
<FRAME src="fr1.html" noresize>
<FRAMESET cols="22%, 78%">
<FRAME src="fr2.html">
<FRAME src="fr3.html" scrolling="yes" marginwidth="10" marginheight="75">
</frameset>
<FRAME src="fr4.html">
</frameset>
</html>
```

ეკრანი დაყოფილია ოთხ ნაწილად. ზედა ნაწილისათვის აკრძალულია ზომების შეცვლა *(მაშასადამე, ეკრანიდან ამოგდება),* ხოლო შუა ნაწილში მარჯვენა ფრეიმისთვის ყველა ვარიანტში ჩანს გადახვევის ზოლები *(ანუ მიუხედავად იმისა, იტევს თუ არა იგი მასში გამოსაყვან ინფორმაციას).*

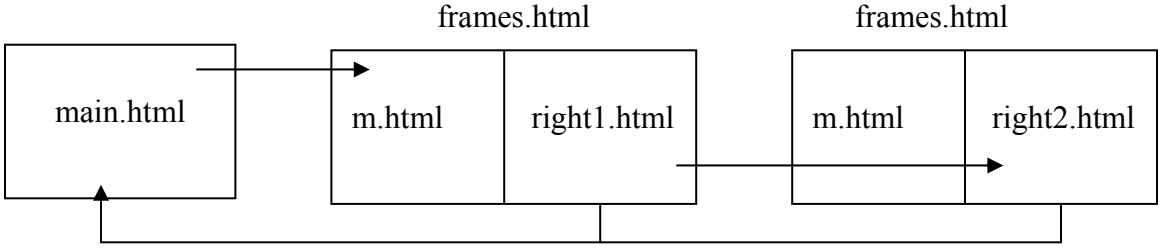
მივაქციოთ ყურადღება – ეკრანის ერთდროულად ვერტიკალურ და ჰორიზონტალურ უბნებად დაყოფა ხდება ერთი **FREIMESET** ელემენტის მეორეში ჩადგმით.

გადასვლების ორგანიზაცია ფრეიმული სტრუქტურის ფურცლებზე

აღვნიშნოთ, რომ საიტი (**site**) განიმარტება, როგორც **Web**-ფურცლების ერთობლიობა, რომელიც ერთ მფლობელს ეკუთვნის. როგორც წესი, დიდი ნაწილი ამ ფურცლებისა ერთ ლოკალურ უბანზე, ხშირად კი ერთ კომპიუტერზეა თავმოყრილი. ფურცლები უმეტესწილად ფრეიმული სტრუქტურის გახლავთ.

ურთიერთკავშირში მყოფი, ფრეიმული სტრუქტურის მქონე ფურცლებისაგან შედგენილი საიტის მაგალითზე ვაჩვენოთ, თუ როგორ უნდა იქნეს ორგანიზებული გადასვლები. საქმე გვაქვს ტიპურ შემთხვევასთან, როცა მთავარი ფურცელი (**Main.html**) ფრეიმებს არ შეიცავს, ხოლო დანარჩენები აგებულია სტანდარტული სქემის მიხედვით: მარცხნივ მენიუა, მარჯვნივ კი – ინფორმაცია.

ფრეიმების კონფიგურაციას განსაზღვრავს **frames.html** ფაილი. ქვემოთ მოყვანილია ფურცლებზე გადასვლების სქემა და ფაილების ლისტინგები:



main.html ფაილის ლისტინგი

```
<HTML>
<HEAD>
<TITLE>main</title>
<STYLE>
A,H2 {font-family: LitNusx}
</style>
</head>
<BODY bgcolor="aqua">
<H2>მთავარი ფურცელი</h2>
<HR>
<A href="frames.html"> შემდეგი ფურცელი</a>
<HR>
</body>
</html>
```

frames.html ფაილის ლისტინგი

```
<HTML>
<HEAD>
<TITLE>frames</title>
</head>
<FRAMESET frameborder=1 framespacing=5 cols=”175,*”>
<FRAME name=”menu” NORESIZE src=”m.html”>
<FRAME name=”info” src=”right1.html”>
</frameset>
</html>
```

m.html ფაილის ლისტინგი

```
<HTML>
<HEAD>
<TITLE>left-frame</title>
<STYLE>
A,H3 {font-family: LitNusx}
</style>
</head>
<BODY bgcolor=”gold” link=”green” vlink=”purple” alink=”red” >
<H3>მენიუ</h3>
<HR>
<A target=_parent href=”main.html”> ძირითადი ფურცელი</a><P>
<A target=”info” href=”right1.html”> მარჯვენა 1</a><P>
<A target=”info” href=”right2.html”> მარჯვენა 2</a>
<HR>
</body>
</html>
```

right1.html ფაილის ლისტინგი

```
<HTML>
<HEAD>
<TITLE>right-frame</title>
<STYLE>
A,H2 {font-family: LitNusx}
</style>
</head>
<BODY bgcolor=#9CB3BE link=”green” vlink=”purple” alink=”red” >
<H2>მარჯვენა ფრეიმი-1</h2>
<HR>
<A href=”main.html” target=_parent > წინა ფურცელი</a>
       
<A href=”right2.html”> შემდეგი ფურცელი</a>
<HR>
</body>
</html>
```

right2.html ფაილის ლისტინგი

```
<HTML>
<HEAD>
<TITLE>right-frame</title>
<STYLE>
A,H2 {font-family: LitNusx}
</style>
</head>
<BODY bgcolor=#9CB3BE link=”green” vlink=”purple” alink=”red” >
<H2>მარჯვენა ფრეიმი-2</h2>
<HR>
<A href=”right1.html” > წინა ფურცელი</a>
       
<A href=”main.html” target=_parent > ძირითადი ფურცელი</a>
<HR>
</body>
</html>
```

საწყის (*ძირითად*) ფურცელზე მოთავსებული ჰიპერკავშირით მივდივართ **frames.html** ფაილზე, რომელშიც არსებულ ინფორმაციაზე დაყრდნობით ბროუზერი განსაზღვრავს ეკრანზე უბნების კონფიგურაციას. ამავე ფაილში მითითებულია, რომელი შინაარსობრივი ფაილი სად მდებარეობს. ეს ინფორმაცია მენიუს სახით გამოდის **m.html** ფრეიმში, ხოლო მარჯვენა ფრეიმში აისახება მენიუში არჩეული რომელიმე ფაილის შემცველობა.

ვხედავთ, რომ ჰიპერკავშირებში გამოყენებული **target** ატრიბუტი მიგვითითებს იმ ფრეიმის სახელზე *(ბრჭყალებშია ჩასმული)*, რომელშიც უნდა მოხდეს საჭირო ფაილის გამოყვანა. მაგრამ, თუ ეს გახლავთ ძირითადი ფურცლის შემცველი ფაილი, მაშინ ვწერთ:

target =_parent.

როცა ინფორმაციის განახლება მიმდინარე ფრეიმში ხდება *(სწორედ მასში, სადაც გაკეთდა ახალი არჩევანი)*, **target** ატრიბუტის გამოყენება საჭირო აღარ არის – მოქმედებს დუმილის წესი.

თუ გვსურს ფაილის შემცველობის ახალ, დამატებით ფანჯარაში გამოყვანა, ამავე ატრიბუტს ვანიჭებთ **_blank** მნიშვნელობას.

<u>ობიექტები</u>

დანიშნულება, საერთო ატრიბუტები

HTML-ში ობიექტად მოიაზრება მომხმარებლის მიერ შექმნილი **Web**-ფურცლის არასტანდარტული ნაწილი. ეს შეიძლება იყოს პროგრამა, ნახატი და ა.შ. ფურცლის შემადგენლობაში ობიექტის ჩართვისათვის გამოიყენება **OBJECT** ელემენტი.

ხშირად ობიექტს ჩარჩოში ათავსებენ, რომლის სისქე განისაზღვრება **border** ატრიბუტით პიქსელებში.

თუ ობიექტი ჰიპერკავშირადაც გამოიყენება, მისი ჩარჩოს ფერის ცვლილებით შეგვიძლია დავადგინოთ, გამოყენებულ იქნა თუ არა ეს კავშირი.

height და **width** ატრიბუტებით შეიძლება განისაზღვროს ობიექტის ზომები, ხოლო **align**-ით – ჩარჩოს მიმართ მისი განლაგება *(ღებულობს მნიშვნელობებს: **"bottom", "top", "left", "right", "middle"**)*.

hspace და **vspace** ატრიბუტებით შეირჩევა ჩარჩოდან დაცილებები *(პიქსელებში)*.

ნახატები და რუკები

ნახატები

<**IMG**> ელემენტის მეშვეობით შეიძლება ეკრანზე განვათავსოთ ობიექტი-ნახატი *(უშუალოდ ან სხვა ელემენტებში, მაგალითად, ცხრილში, ჩასმით)*. ნახატი შეიძლება გამოვიყენოთ ჰიპერკავშირად, მარკერად, დიზაინის ამოცანების გადასაწყვეტად და ა.შ. ამ ელემენტის აუცილებელი კომპონენტი გახლავთ **src** ატრიბუტი, რომელიც გრაფიკული ფაილის **URL**-ს წარმოადგენს.

*შენიშვნა: აღსანიშნავია, რომ ინტერნეტიდან **HTML**-ფაილების კოპირება ჯერ კიდევ არ გვაძლევს საშუალებას, ვისარგებლოთ მათში არსებული ნახატებით. ეს ნახატები კეშ-საქაღალდეში უნდა მოვიძიოთ და მოვახდინოთ ჩვენთვის სასურველ საქაღალდეში მათი კოპირება, შემდეგ კი **HTML** ფაილში **src** ატრიბუტებს უნდა მივანიჭოთ ნახატებამდე ახალი გზების ამსახველი მნიშვნელობები.*

სასარგებლო როლს ასრულებს **alt** ატრიბუტი. ჩარჩოში, რომელშიც ნახატი იტვირთება ინტერნეტიდან *(ეს პროცესი არცთუ ისე იშვიათად ჭიანურდება),* გამოდის, ვთქვათ, ასეთი წარწერები:

ფოტოსათვის – **alt=”My photo”,** ნახატიანი ღილაკისათვის – **alt=**”**send**”, ჰიპერკავშირისათვის – **alt=**“**URL**”.

ნახატისათვის, როგორც წესი, უჩვენებენ ერთ ზომას (**height** ან **width**), რათა დაცულ იქნეს პროპორცია.

რუკები

IMG ელემენტები შეიძლება წარმოვიდგინოთ, როგორც <**MAP**> (რუკა) ელემენტის შემადგენელი ნაწილები, რომელთაგან თითოეული დაკავშირებულია ჰიპერდაყრდნობასთან და თაგვის დაწკაპუნებისას სხვაგან გადასვლის შესაძლებლობას უზრუნველყოფს. თუკი ამ მოქმედებაზე სერვერმა უნდა მოახდინოს რეაგირება, ელემენტში ჩავრთავთ **ismap** ატრიბუტს. საინტერესოა, რომ აქ ატრიბუტისათვის მნიშვნელობის მინიჭება სავალდებულო არცაა, თუმცა ხშირად მას მაინც განუსაზღვრავენ მნიშვნელობას:

ismap=”**ismap**”

თუ ეს ჩვენი ქმედება *(თაგვით დაწკაპუნება)* ბროუზერის დასამუშავებელია, ვიყენებთ სხვა ატრიბუტს, რომელშიც მიეთითება რუკის სახელი:

usemap=”# რუკის სახელი”

რუკის გამოყენებისას, პირველი, რაც უნდა გავაკეთოთ, ეს გახლავთ <**MAP**> ელემენტის მეშვეობით მისთვის სახელის განსაზღვრა და არეებად წარმოდგენა შემდეგი **AREA** ელემენტებით:

- წრე (**circle**);
- მართკუთხედი (**rect**);
- მრავალკუთხედი (**poly**).

(ელემენტებისთვის უნდა ვუჩვენოთ შესაბამისი პარამეტრების მნიშვნელობები. იხ. ქვემოთ).

თითოეული არისათვის აუცილებელია **href** ატრიბუტის მეშვეობით გადასვლის მისამართის განსაზღვრა. დასაშვებია მასში გამოსაყვანი ნახატის შემცველი წარწერის შექმნაც.

ამის შემდეგ, ვქმნით რუკასთან დაკავშირებულ **IMG** ობიექტებს *(საერთოდ, მათი მოთავსება **Web**-ფურცლის ნებისმიერ ადგილზე შეიძლება).*

თითოეულ ობიექტს განსაზღვრის თავისი წესი აქვს. მაგალითად, წრისათვის უნდა ვუჩვენოთ ცენტრის **x, y** კოორდინატები და **z** რადიუსი, მრავალკუთხედისთვის - თითოეული წვეროს, ხოლო მართკუთხედისათვის – ზედა მარცხენა და ქვედა მარჯვენა კუთხეების კოორდინატები.

ქვემოთ მოყვანილია **Map**.**html** ფურცლის მაგალითი, რომელზეც განლაგებულია რუკა. იგი რამდენიმე მრავალკუთხედს შეიცავს. დანართში მოცემულია ამ რუკასთან დაკავშირებული ნახატი, ქვემოთ კოდში map.jpg ფაილის მეშვეობით წარმოდგენილი.

```
<HTML>
<HEAD>
 <TITLE>რუკა</title>
 <META http-equiv="Content-Type" content="text/html; charset=windows-1252">
</head>
<BODY>
 <map name="გადასვლები">
    <area href=mainpage.html shape="polygon" coords="81,108, 125,45, 212,28, 212,158">
    <area href=test.html shape="polygon" coords="216,25, 290,49, 337,109, 213,159">
    <area href=letters.html shape="polygon" coords="215,170, 340,118, 353,205, 304,274">
    <area href=about.html shape="polygon" coords="208,177, 297,282, 217,309, 121,279">
    <area onclick=parent.close(); shape="polygon" coords="199,169, 113,274, 66,204, 78,121" >
 </map>
 <H1 align="center" ><FONT face="AcadNusx">ბმულების რუკა</font></H1>
  <HR WIDTH="100%" SIZE="1">
   <p align="center"><img border="0" src="map.jpg" ismap usemap="#გადასვლები"></p>
  <HR WIDTH="100%" SIZE="1">
</body>
</html>
```

რუკისადმი მიმართვა **OBJECT** ელემენტის მეშვეობითაც შეიძლება:

<OBJECT data=”ფაილის სახელი.**gif“ type= ”image/gif” usemap=”#karta”>**
</object>

gif-ის გარდა, შეიძლება **jpg** და **png** ფორმატების გამოყენებაც. თუ ნახატი სხვა ფორმატში მომზადდა, უმრავლეს შემთხვევაში ხერხდება რომელიმე ზემოაღნიშნულ ფორმატთაგანში მისი გადაყვანა.

უფრო დაწვრილებით განვიხილოთ ზოგიერთი **OBJECT** ელემენტი.

<APPLET> </applet>

ამ ელემენტის გამოსაყენებლად ბროუზერში ჩაშენებული უნდა იყოს **JAVA**-ს ინტერპრეტატორი.

HTML კოდში აპლეტის გამოძახება ასე ხდება:

<APPLET CODE=”ფაილის-სახელი.**class**” **width**=nnn **height**=mmm> *კომენტარის ჰიპერტექსტი* **</applet>**

Java-პროგრამის კომპილირებული ფაილი აქ მხოლოდ იმ საქაღალდეში უნდა მოთავსდეს, რომელშიც განლაგდება **Web**-ფურცელი *(სხვა ბაზური მისამართის ჩვენებისათვის აუცილებელია* ***codebase*** *ატრიბუტის გამოყენება. უმეტეს*

*შემთხვევაში ბაზური მისამართის ნაცვლად აქ აპლეტის სრულ **URL** მისამართს უჩვენებენ. ხოლო, თუ ეს ატრიბუტი ნაჩვენები არაა, გამოიყენება დუმილით გათვალისწინებული **URL**).*

შემდეგ ნაჩვენებია ფანჯრის ზომები (*პიქსელებში*).

კომენტარში მოთავსებული ჰიპერტექსტი მხოლოდ იმ ბროუზერებს გამოჰყავთ, რომლებიც **Java**-სთან "მწყრალად" გახლავთ.

თუ ფურცელზე რამდენიმე ერთმანეთთან "მოსაუბრე" აპლეტი გამოიყენება, საჭირო ხდება მათი სახელების გათვალისწინებაც **name** ატრიბუტის მეშვეობით:

name="აპლეტის სახელი"

დასასრულ, აქაც შეიძლება მივმართოთ სხვა, ჩვენთვის უკვე ნაცნობ ატრიბუტებს: **alt**, **align**, **vspace**, **hspace** და ა.შ.

<OBJECT> </object>

უმარტივეს შემთხვევაში ეს ელემენტი ფურცელზე ნახატს განათავსებს:

<**OBJECT date**="ფაილის სახელი.**png**" **type**="**image/png**">

ამჯობინებენ **APPLET** ელემენტის **OBJECT** ელემენტით შეცვლას:

<**OBJECT codetype**="**application/java**" **classid**="**java**:იდენტიფიკატორი" **width**="nnn" **height**="mmm">
ობიექტის ტექსტური აღწერა
</object>

ჩამოვთვალოთ **OBJECT** ელემენტის ძირითადი ატრიბუტები:

- **classid**="ობიექტის სახელი" - ობიექტის უნიკალური იდენტიფიკატორი ან მისამართი;
- **codebase**="**URL**" - სრული ან ბაზური **URL**.

  თუ გამოიყენება ეს უკანასკნელი, მაშინ იყენებენ **data** და, შესაძლოა, **archive** ატრიბუტებსაც. **data** მიუთითებს ობიექტის ფარდობით მისამართზე *(თუმცა დასაშვებია სრული **URL**-ის ჩვენებაც),* ხოლო **archive** უჩვენებს ერთ ან რამდენიმე დამატებითს მისამართს **data**-ს სტილში.
- **codetype**="ტიპი" - ობიექტის ტიპი. გამოიყენება **classid** ატრიბუტთან ერთად;
- **type**="ტიპი" - ობიექტის ტიპი (**MIME**);
- **stadby**="შეტყობინების ტექსტი" - მოქმედებს, როგორც ადრე შესწავლილი **alt** ატრიბუტი.

ზოგჯერ ხდება, რომ ობიექტის გააქტიურება ვერ ხერხდება. ასეთი შემთხვევისათვის შეგვიძლია ტექსტური შეტყობინების გათვალისწინება:

<**OBJECT classid**="იდენტიფიკატორი" **data**="მისამართი/ფაილის სახელი.ტიპი">
ობიექტის გამოყვანა ვერ ხერხდება </**object**>

შეიძლება "ავარიულის" ნაცვლად ობიექტში ჩაშენებული სხვა ობიექტის *(ვთქვათ, ნახატის)* გამოყვანა, მაგალითად:

<**OBJECT title**=”ტექსტი” **classid**=”მისამართი/ფაილის სახელი.ტიპი” >
<**OBJECT title**=”სხვა ტექსტი” **data**=”ნახატის ფაილის სახელი.**gif**” **type=”image/gif”** >
</object>
</object>

შევნიშნოთ, რომ ობიექტს იგივე ატრიბუტები თავიანთი მნიშვნელობებით შეიძლება გადაეცეს მასში ჩადგმული <**PARAM**> ელემენტების სახითაც.

ფორმები

დანიშნულება, საერთო ატრიბუტები

ფორმა წარმოადგენს ე.წ. **მართვის ელემენტების** კონტეინერს. ასეთი ელემენტების მაგალითებია: ტექსტის შეტანის ველები, გადამრთველები, ღილაკები, ალმები და სხვ.

მართვის თითოეული ელემენტი იქმნება **HTML** ელემენტების მეშვეობით.

ფორმის გააქტიურების მომენტში ელემენტებს შეიძლება მინიჭებული ჰქონდეთ რაიმე მნიშვნელობა. მაგალითად, შეტანის ველისთვის ეს შეიძლება იყოს დუმილით არჩეული ტექსტი.

ელემენტს მივმართავთ **name** ატრიბუტის მეშვეობით, მნიშვნელობა განისაზღვრება **value** ატრიბუტით. გადამრთველებში იმ ელემენტის ნომერი, რომლის არჩევანსაც აკეთებს უმეტესწილად მომხმარებელი, განისაზღვრება ,,1“-ით და ეს მნიშვნელობა მიენიჭება **tabindex** ატრიბუტს. სწორედ მასზე დგას კურსორი ფორმის გახსნისას. **tab** ღილაკზე ხელის დაჭერით კურსორი გადადის მომდევნო (2,3,...) ველებზე.

disabled ატრიბუტის ჩართვით ფორმის ელემენტი შეუღწევადი ხდება. მისი ტექსტი ფერმკრთალდება. ასეთი ელემენტის არჩევა და მისთვის მნიშვნელობის შეცვლა ვერ ხერხდება.

მართვის ელემენტებთან შეიძლება დაკავშირებული იყოს **ხდომილობები**. ხდომილობები ის სიტუაციებია, რომლებსაც იწვევს ჩვენ მიერ კომპიუტერზე შესრულებული მოქმედებები. მოვახდინოთ მათი კლასიფიცირება:

- *კლავიატურასთან დაკავშირებული ხდომილობები*

– **onkeydown** - კლავიში დაჭერილია;
– **onkeyup** - კლავიში აშვებულია;
– **onkeypress** - კლავიში დაჭერილ და აშვებულ იქნა.

- თაგვთან დაკავშირებული ხდომილობები

– **onclick** - ელემენტზე დაწკაპუნება;
– **ondbclick** - ორმაგი დაწკაპუნება;
– **onmousedown** - დაჭერილია თაგვის ღილაკი;
– **onmouseup** - თაგვის ღილაკი აშვებულია;
– **onmousemove** - თაგვის მაჩვენებელი ელემენტის უბანში გადაადგილდა;
– **onmouseover** - თაგვის მაჩვენებელი იმყოფება ელემენტზე;

– **onmouseout** – თაგვის მაჩვენებელი გასცდა ელემენტის უბნის ფარგლებს.

▪ *ელემენტების არჩევასა და ფორმების რედაქტირებასთან დაკავშირებული ხდომილობები*

– **onfocus** - ელემენტი არჩეულია *(მიიღო ფოკუსი);*
– **onselect** - ელემენტის შიგნით მოინიშნა ტექსტის ნაწილი;
– **onchange** - მონაცემები ელემენტში შეცვლილ იქნა;
– **onblur** - ელემენტმა დაკარგა ფოკუსი.

ფორმათა ელემენტები

<**ISINDEX**>

ეს მარტივი ელემენტი მომხმარებელს აწვდის კომპიუტერთან დიალოგის წარმართვის საშუალებას. კომპიუტერის მხრიდან, კარნახის მიზნით, დასაშვებია **promt** ატრიბუტის გამოყენებაც. მოვიყვანოთ ეკრანზე შეტანის ველის შექმნის მაგალითი:

<**ISINDEX promt**="*შეიტანეთ ძებნის კრიტერიუმი*">

დავუშვათ, რომ მიმდინარე **Web**-ფურცლისთვის ბაზურ მისამართად წინასწარ განსაზღვრულია ინტერნეტში რომელიმე საძიებო საშუალების **URL.** თუ მომხმარებელი **ISINDEX** ელემენტის ველში შეიტანს რამდენიმე საკვანძო სიტყვას, ბროუზერი სერვერზე გადააგზავნის მოთხოვნას:

http://**www**.დასახელება.დომენი/?სიტყვა1+სიტყვა2+სიტყვა3

შენიშვნა: ზოგიერთი საძიებო სერვერი "?" და "+" სიმბოლოებს ძებნის პროცესში ვერ იყენებს.

<**FORM**> </**form**>

მომხმარებლის მიერ ინფორმაცია ფორმაზე, როგორც წესი, რამდენიმე ველში შეიტანება და მხოლოდ შემდეგ გადაიგზავნება იგი სერვერზე ამ ფორმასთან დაკავშირებული, ვთქვათ, **CGI**-პროგრამით დასამუშავებლად.

action ატრიბუტი ამ პროგრამის სახელის ჩვენებისათვის არის გათვალისწინებული:

<**FORM action**="**http://www**.დასახელება.დომენი/პროგრამის სახელი"
method="**post**">
ფორმის ელემენტები
</**form**>

ფორმის დამუშავების ერთ-ერთი ვარიანტია ელექტრონული ფოსტის მეშვეობით მონაცემების გაგზავნა:

action="**mailto**:*მისამართი@სერვერი.დომენი*"

ფორმას სხვადასხვა მომხმარებლები ავსებენ, რომლებიც შესაძლოა, სხვადასხვა კოდირებას იყენებდნენ. მათი ჩამონათვალის ჩვენებისათვის გათვალისწინებულია ატრიბუტი

accept-charset = *"კოდირებების სია"*

ფორმების უმეტესობაზე ათავსებენ **reset** და **submit** ღილაკებს, რომლებთანაც დაკავშირებულია ფორმის შევსების სისწორის კონტროლისა და გასუფთავების პროგრამები. ამ პროგრამების გამოსაძახებლად გათვალისწინებულია ხდომილობები:

onsubmit და **onreset.**

<INPUT>

ამ ელემენტით ვქმნით ფორმის ისეთ ნაწილებს, როგორებიცაა:

ალამი, გადამრთველი, შეტანის ველი.

ელემენტი საბოლოო ტეგს არ საჭიროებს, რადგანაც ყველა პარამეტრი ატრიბუტების მეშვეობით იქმნება.

ელემენტის სახეს განსაზღვრავს **type** ატრიბუტი. იგი ღებულობს მნიშვნელობებს:

„**text**“, „**password**“, „**checkbox**“ *(ალამი),* „**radio**“ *(ერთი გადამრთველი).*

გადამრთველთა ჯგუფის შესაქმნელად გამოვიყენოთ შემდეგი კონსტრუქცია:

<**H3**> გადამრთველები </**h3**>

<**INPUT type**=”**radio**” **name**=”**SOO**1” **value**=”პირველი”>

<**INPUT type**=”**radio**” **name**=”**SOO**1” **value=**”მეორე”>

<**INPUT type**=”**radio**” **name**=”**SOO**1” **value=**”მესამე” **checked**>

checked ატრიბუტი განსაზღვრავს, თუ რომელი გადამრთველი უნდა აირჩეს დუმილით. ეკრანზე ეს გადამრთველები ასეთი სახით გამოვლენ:

ვაგრძელებთ **type** ატრიბუტის ნაირსახეობების ჩამოთვლას:

– “**button**” - ქმნის რაიმე დანიშნულების ღილაკს;

– “**submit**” - ამ ღილაკზე დაწკაპუნება ადასტურებს ინფორმაციის შეტანას ფორმაში;

– “**reset**” - ფორმას ასუფთავებს მონაცემებისგან;

– “**image**” - ქმნის ნახატიან ღილაკს. გრაფიკული ნახატის მითითება ხდება **src** ატრიბუტით;

– “**file**” - მომხმარებელს შეუძლია ფაილის სახელი თვითონვე აკრიბოს შეტანის ველში ან მის მოსაძებნად მიმართოს ბროუზერის მიერ იქვე გამოტანილ **browse** ღილაკს. არჩეული ფაილი ფორმას მიეერთდება.

– “**hidden**” - მომხმარებლისგან დამალული ელემენტის შექმნა, ფაქტობრივად, ფორმაში ცვლადისა და მისი მნიშვნელობის განსაზღვრისათვის არის გათვალისწინებული.

ყოველ **INPUT** ელემენტში ჩართული უნდა იყოს **name** ატრიბუტი. **value** ატრიბუტი შეტანის ველისთვის ქმნის დუმილით ნაგულისხმებ მნიშვნელობას, ხოლო ღილაკისთვის - წარწერას.

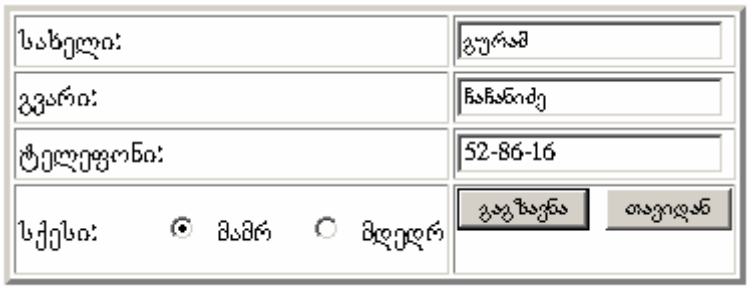
გადამრთველებში **checked** ატრიბუტი შეიძლება გამოვიყენოთ მხოლოდ ერთადერთი ელემენტისთვის, ალმების შემთხვევაში კი – მათი ნებისმიერი რაოდენობისთვის.

size ატრიბუტი განსაზღვრავს შეტანის ველის სიგრძეს *(სიმბოლოებში).* სასურველია ამ სიგრძის რამდენადმე გადიდება.

FORM ელემენტის ანალოგიურად, **INPUT**-შიც დასაშვებია **accept** ატრიბუტის ჩვენება. გათვალისწინებულია აგრეთვე სხვა, შედარებით ნაკლებმნიშვნელოვანი ატრიბუტებიც.

მოვიყვანოთ ფორმის ნიმუში ანკეტური მონაცემების შესატანად:

```
<HTML>
<HEAD>
<STYLE>
table, input {font-family: LitNusx}
</style>
<META http-equiv="Content-Type" content="text/html; charset=windows-1252">
<TITLE>TABLE</title>
</head>
<BODY>
<TABLE border=3>
<FORM action="mailto:სახელი@სერვერი.დომენი" method="post">
<TR>
<TD>სახელი:
<TD><INPUT type="text" name="saxeli">
<TR>
<TD>გვარი:
<TD><INPUT type="text" name="gvari">
<TR>
<TD>ტელეფონი:
<TD><INPUT type="text" name="telefoni">
<TR>
<TD>sqesi:   
<INPUT type="radio" name="sqesi" value="მამრ"> მამრ 
<INPUT type="radio" name="sqesi" value="მდედრ"> მდედრ
<TD align="bottom">
<INPUT align="bottom" type="submit" value="გაგზავნა">
<INPUT type="reset" value="გასუფთავება">
</form>
</table>
</body>
</html>
```



ამ მაგალითში ფორმის ელემენტების ეკრანზე მწყობრად გამოსატანად გამოყენებულ იქნა ცხრილი. შევიტანოთ კოდში ჩვენი საფოსტო მისამართი. "გაგზავნა" ღილაკზე დაწკაპუნებით წერილი გადაინაცვლებს საქაღალდეში "გასაგზავნი წერილები". შევიდეთ ამ საქაღალდეში და წერილი გავუგზავნოთ საკუთარ თავს.

<SELECT> <OPTION> </select>

SELECT ელემენტი განკუთვნილია **Web**-ფურცელზე სიის ან მენიუს გამოსაყვანად, ხოლო **OPTION** ელემენტი – სიის პუნქტის შესაქმნელად. დავწეროთ შემდეგი კოდი:

```
<SELECT>
  <OPTION value=a> პირველი
  <OPTION value=b> მეორე
  <OPTION value=c> მესამე
  <OPTION value=d> მეოთხე
 </select>
```

შედეგად მივიღებთ შემდეგ სიას:

SELECT ელემენტში გამოყენებული ძირითადი ატრიბუტებია:

- **name** – ახდენს მენიუს სახელდებას;
- **multiply** *(მნიშვნელობა არ ენიჭება)* – მომხმარებელს ეძლევა ერთდროულად სიის რამდენიმე პუნქტის ამორჩევის საშუალება;
- **size** ატრიბუტი იმ პუნქტების რიცხვს განსაზღვრავს, რომლებიც ეკრანზე ჩანს. შევნიშნოთ, რომ დუმილით მხოლოდ ერთი სტრიქონი აისახება, მის მარჯვნივ ისარზე დაწკაპუნებით კი ჩამოიშლება მთელი სია.

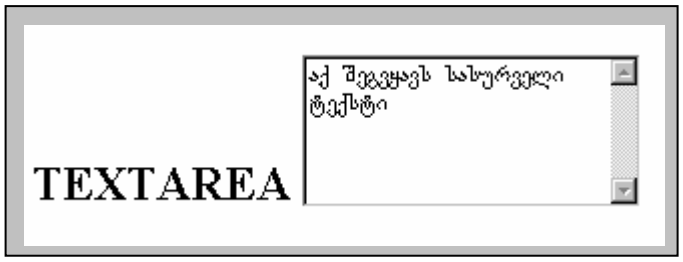
OPTION ელემენტში **selected** ატრიბუტი *(მნიშვნელობებს არ ღებულობს)* განსაზღვრავს დუმილით არჩეული სიის პუნქტს, ხოლო **value** ატრიბუტი სერვერის მხარეზე მონაცემთა დამუშავებისთვის არის განკუთვნილი.

<TEXTAREA> </textarea>

ეს ელემენტი, **SELECT**-ის მსგავსად, ეკრანზე შეიძლება ავტონომიურადაც გამოვიყვანოთ ანუ ფორმის ელემენტში მოთავსების გარეშე. იგი **rows** და **cols** ატრიბუტებით ქმნის ტექსტის შეტანის და ჩათვალიერების უბანს. **readonly** ატრიბუტით იკრძალება ტექსტის რედაქტირება.

აღვნიშნოთ, რომ **SELECT** და **TEXTAREA** ელემენტები იძლევა ეკრანზე ინფორმაციის კომპაქტურად განლაგების საშუალებას.

```
<H2> TEXTAREA ელემენტი</h2>
<TEXTAREA  name = "text_25" rows=5 cols=25 >
აქ შეგვყავს სასურველი ტექსტი
</textarea>
```



<BUTTON> </button>

INPUT ელემენტისაგან განსხვავებით, ასეთი ღილაკი კონტეინერი გახლავთ, მაშასადამე, აქვს ბოლო ტეგიც. ეს ნიშნავს, რომ ელემენტზე შეგვიძლია ტექსტისა და ნახატის დატანაც. მაგალითად:

```
<BUTTON  name = "სახელი"   value ="submit"  type ="submit"  >
ტექსტი <IMG  src = "ნახატის ფაილი">
</button>
```

type ატრიბუტი ღებულობს **button, submit** ან **reset** მნიშვნელობას.

ლიტერატურა:

1. А. Гончаров. Самоучитель HTML, «Питер» , 2001

2. А. Матросов, А. Сергеев, М. Чаунин. HTML4, «Вильямс» , 2003

3. Учебный курс «Компьютерные сети», Microsoft Press. Санкт-Петербург, 1999.

ტერმინების ლექსიკონი:

HTML (HyperText Markup Language – *ჰიპერტექსტის მონიშვნის ენა)* **–** **Web-**დოკუმენტების მონიშვნის *(გაწყობის, დაფორმატების)* ენა.

Web-ფურცელი - **Web-**სერვერზე შენახული დოკუმენტი *(უმეტესწილად* ***HTML*** *ფორმატის მქონე),* რომელიც სერვერიდან ჩამოიტვირთება კლიენტის **Web-**ბროუზერში. შეიძლება განთავსდეს ლოკალურ კომპიუტერზეც.

Web-ბროუზერი – პროგრამა, რომლის მეშვეობითაც ხორციელდება **WWW-**ში ინფორმაციის მოძიება და ჩათვალიერება.

Internet Explorer – კორპორაცია **Microsoft-**ის მიერ შექმნილი **Web-**ბროუზერი.

Navigator – კორპორაცია **Netscape Communications-**ის მიერ შექმნილი **Web-**ბროუზერი. ხშირად მას **Netscape-**საც უწოდებენ.

Web-კვანძი - **WWW-**დოკუმენტების კრებული. აქვს საწყისი ფურცელი, რომლიდანაც გადავდივართ კრებულის სხვა ფურცლებზე.

WWW (World Wide Web) – ინტერნეტის ყველაზე პოპულარული სამსახური, რომელიც მეტად აადვილებს ინფორმაციის ძებნის პროცესს და საერთოდ, ინტერნეტის სამყაროში მოგზაურობას.

ჰიპერტექსტი - ჩვეულებრივ ტექსტზე უფრო მეტი ინფორმაციული და ფუნქციური მონაცემების შემცველი დოკუმენტი.

ელემენტი - **HTML**-ის კონსტრუქცია – კონტეინერი, რომელიც შეიცავს ამა თუ იმ წესით დასაფორმატებულ *(ან რაიმე სხვა გზით დასამუშავებელ)* მონაცემებს.

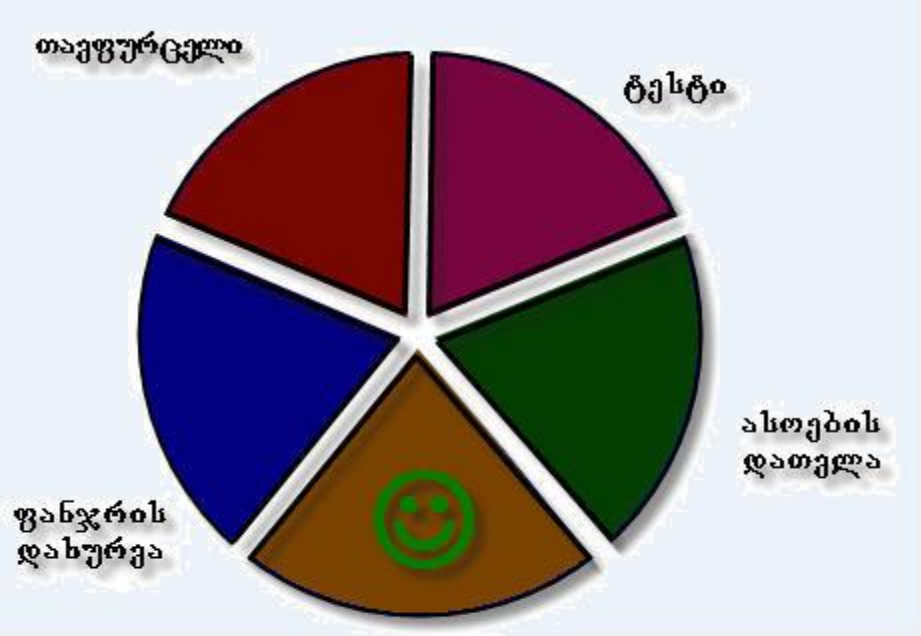
აპლეტი (**Applet**) - პროგრამა, რომელიც **Web**-ფურცლის ჩათვალიერებისას დინამიურად მიეერთდება **HTML-**კოდს ფაილის სახით.

სცენარი (Script) - პროგრამა, რომელიც უმეტეს შემთხვევაში დაწერილია **JavaScript**-ის ერთ-ერთ ვერსიაში და წარმოადგენს **Web**-ფურცლის მაფორმირებელ **HTML-**კოდში ჩართულ კომპონენტს.

ფრეიმი – ეკრანის უბანი **Web**-ფურცლის ჩათვალიერების შესაძლებლობით. ეკრანის სტრუქტურის განსაზღვრა *(ვერტიკალზე ან ჰორიზონტალზე მისი დაყოფა ფრეიმებად)* ხორციელდება **cols** და **rows** ატრიბუტების მეშვეობით.

GIF – ნახატი ფაილების ყველაზე უფრო პოპულარული ფორმატი **Web-**სივრცეში.

სარჩევი



დაწართი