VISOKA ŠKOLA
ELEKTROTEHNIKE I RAČUNARSTVA
STRUKOVNIH STUDIJA

Zoran Ćirović

UVOD U INTERNET TEHNOLOGIJE

Naslov:	Uvod u Internet tehnologije drugo izdanje
Autor:	dr Zoran Ćirović
Recenzenti:	dr Boško Nikolić
	dr Zoran Banjac
Izdavač:	Visoka škola elektrotehnike i računarstva strukovnih studija u Beogradu
Tehnička obrada:	Zoran Ćirović
Korice:	Ana Miletić
Štampa:	Razvojno-istraživački centar grafičkog inženjerstva TMF, Beograd
Tiraž:	20
ISBN:	978-86-7982-290-1

CIP - Каталогизација у публикацији - Народна библиотека Србије, Београд

004.42:004.738.5(075.8)

ЋИРОВИЋ, Зоран, 1970-
Uvod u Internet tehnologije / Zoran Ćirović. - 2. [dopunjeno] izd. -
Beograd : Visoka škola elektrotehnike i računarstva strukovnih studija,
2018 (Beograd : Razvojno-istraživački centar grafičkog inženjerstva TMF). -
311 str. : ilustr. ; 24 cm

Tiraž 20. - Bibliografija: str. 300. - Registri.

ISBN 978-86-7982-290-1

a) Интернет - Програмирање
COBISS.SR-ID 270259468

Sadržaj

Predgovor

Knjiga „Uvod u Internet tehnologije" najvećim delom pokriva gradivo istoimenog predmeta koji se sluša na drugoj godini osnovnih studija Visoke škole za elektrotehniku i računarstvo strukovnih studija u Beogradu.
Knjiga je podeljena u osam poglavlja koja postepeno uvode nove pojmove počev od osnova HTML-a, preko CSS-a pa sve do praktičnih dizajnerskih tehnika i JavaScript jezika.
Prvo poglavlje je posvećeno osnovama jezika HTML. Prvo se uvode osnovni koncepti jezika kao što su pravila pisanja, struktura dokumenta, tagovi, atributi, praznine i komentari. Zatim se detaljno definišu elementi HTML jezika sa pratećim primerima.
Drugo poglavlje je posvećeno XML standardu, odnosno XHTML-u. Najpre se daju osnove standarda, definišu se pravila dobro formiranog dokumenta. Zatim se uvodi pojam validacije dokumenta i definiše DTD. Uz pomoć XML-a i DTD-a definiše se specifikacija XHTML sa posebnim isticanjem razlika u odnosu HTML.
Treće poglavlje detaljno se bavi dizajnom web stranica pomoću kaskadnih stilova. Prvo se definiše struktura CSS zapisa, načini povezivanja sa HTML dokumentom, zatim se opisuju načini formiranja selektora i primena sitlova. Izložena svojstva i atributi su detaljno opisani sa pratećim primerima. Na kraju poglavlja se uvode plutajući elementi.

U četvrtom poglavlju je prezentovano nekoliko praktičnih primera i različitih načina dizajna web stranica. Naglasak je na dizajnu web stranica sa više kolona i u nekoliko varijanti.
U petom poglavlju je objašnjeno nekoliko praktičnih tehnika koje se primenjuju pri dizajnu web stranica. Obrađene su teme: reset tj. poništavanje već postojećeg CSS-a, upotreba slika umesto teksta i dizajn formi.
U šestom poglavlju se rade tehnike za prilagodljivi dizajn. Kreće se od pojma prozora za prikaz, a zatim se radi mrežasta organizacija sadržaja. Nakon toga uvode se medija upiti, tačke preloma kao i tehnika dizajna zasnovanog na fleksibilnom okviru.
Sedmo poglavlje je posvećeno odabranim svojstvima CSS3 standarda: zaobljavanje ivica, senčenje, gradijenti boja, transformacije, animacije i tranzicije.
U poslednjem poglavlju rade se osnove JavaScript jezika.
Nakon svakog poglavlja data su pitanja namenjena proveri i boljem razumevanju izložene materije.
Cilj knjige je da posluži kao udžbenik za sticanje osnovnih znanja i veština u radu sa HTML dokumentima . Zato knjiga obiluje primerima praćenih sa slikama, odgovarajućim kodom i dodatnim objašnjenjem.
Beograd, 2018. Autor

Deo 1: HTML

- HTML dokument
- Zaglavlje HTML stranice
- Telo HTML stranice
- Formatiranje teksta
- Liste
- Slike
- Hiperveza
- Tabele
- Font
- Frejmovi
- Forme

Prvo poglavlje je posvećeno osnovama jezika HTML. Na početku se daju osnovni koncepti jezika kao što su pravila pisanja, struktura dokumenta, tagovi, atributi, praznine i komentari. Zatim se detaljno definišu elementi HTML jezika sa primerima. Naglašava se razlika u sintaksi i primeni verzije HTML5 odnosno HTML 4.01. Posebno se vodi računa o postepenom uvođenju elemenata i pratećim primerima koji ilustruju njihovu praktičnu primenu.

HTML je skraćenica od engleskih reči **H**yper**T**ext **M**arkup **L**anguage. „Hypertext“ označava deo teksta koji ima ulogu veze (eng. link), a „Markup Language“ označava način definisanja informacija u dokumentu. HTML je jezik za prikaz sadržaja **web stranice**.

HTML dokument

HTML dokument je opšti naziv za web stranicu. HTML dokument je dokument **tekstualnog formata**, najčešće u obliku fajla. HTML fajlovi imaju ekstenziju (sufiks u svom imenu) .**html** ili **.htm**.

Kreiranje jedne web stranice tj. pisanje HTML koda se može vršiti korišćenjem bilo kog editora teksta. Primeri takvih editora u Windows i Mac OS X operativnom sistemu su Notepad i TextEdit respektivno. Postoje specijalizovani programi za editovanje baš HTML koda, besplatni ili komercijalni, koji mogu olakšati dizajn web stranice i unos koda sa validacijom.
HTML nije programski jezik nego markup jezik ili jezik označavanja. Sastoji se od elemenata koji su uokvireni znacima manje i veće (**<…>**). Ove oznake u HTML kodu nazivaju se **tagovi**.
Programi za prikaz sadržaja HTML dokumenta se nazivaju **web čitačima**. Najčešće korišćeni čitači su: Chrome, Opera, Mozilla, Egde.

Elementi

Elementi formiraju strukturu HTML dokumenta, definišu izgled i sadržaj web stranice koju web čitač prikazuje.
Oznake za elemente nazivaju se **tagovi**. Naziv elementa je smešten između znakova manje (<) odnosno veće (>). Na primer element **body** se zapisuje kao: **<body>**. HTML dokument se sastoji od više elemenata koji na odgovarajući način definišu izgled i sadržaj svake web stranice.
Element može biti **složen** tj. sastavljen iz dva dela, početnog i završnog. Završni tag tj. oznaka ima isti naziv, ali sa kosom crtom iza znaka manje (</), na primer: **</body>**. Između početnog i završnog taga je smešten sadržaj elementa.

`Element *body* u HTML dokumentu:`

```
<body>
   ... sadržaj elementa body ...
</body>
```

Neki elementi nemaju završni tag `</...>`. Na primer, takav je element kojim se označava novi red: `<br>` ili `<br/>`, koji po svojoj prirodi značenja nema sadržaj, pa je stoga i nepotrebno da postoji i početni i završni tag. Takvi elementi se nazivaju **prostim** elementima.

Atributi

HTML atributi se koriste kako bi se dodatno opisali HTML elementi. Atributi se navode u početnom tagu elementa. Svaki atribut ima svoj naziv i vrednost, a između njih je znak jednakosti. Vrednost atributa je uvek zapisana između znakova navoda. Na primer, atributom se može definisati pozadinska boja stranice:

```
<body bgcolor="yellow">
```

element `body` ima atribut `bgcolor` kome je dodeljena vrednost `yellow`.

Jedan element može imati više atributa. U tom slučaju atributi se međusobno razdvajaju razmakom. Mogu se navoditi proizvoljnim redosledom. Na primer:

```
<img src="slika1.jpg" alt="portret">
```

element `img` sadrži dva atributa. Nazivi atributa su `src`, odnosno `alt`, dok su vrednosti atributa `slika1.jpg` odnosno `portret` respektivno.

Vrednosti atributa pišu se obično između dvostrukih, ali se mogu koristiti i jednostruki znaci navoda. Nekada je upotreba obe vrste navoda neophodna, kao u slučaju kada vrednost atributa sadrži znake navoda. Na primer:

```
<...name='Jovan Jovanović - "Zmaj"'>.
```

Većina atributa su opcioni tj. ne moraju se eksplicitno pisati u kodu. Tada atribut dobija podrazumevanu vrednost. Nasuprot opcionih, kod nekih elemenata određeni atributi su obavezni jer sadrže osnovne informacije bez kojih element ne bi imao smisla. Na primer, element `img` definiše sliku na web stranici, a sadrži obavezni atribut `src` za definisanje putanje do slike koja se prikazuje.

Struktura dokumenta

Svaki HTML dokument ima unapred definisanu strukturu osnovnih elemenata. U primeru koji sledi, prikazana je osnovna struktura svake

web stranice. Nju čine elementi `head` i `body`, koji su u elementu `html`.

`Struktura HTML dokumenta:`

```
<html>
    <head>
        opis osnovnih podataka o dokumentu
    </head>
    <body>
        sadržaj dokumenta
    </body>
</html>
```

U elementu `head` se obično navodi naziv dokumenta, tip dokumenta, kodni raspored, jezik, ključne reči, ... Ovaj element sadrži elemente koji dodatno opisuju sam html dokument ili ga povezuju s drugim fajlovima. Sledeći elementi mogu biti dodati u `head` deo: `title, base, link, meta, script, style`.
U elementu `body` su svi elementi koji formiraju prikaz sadržaja na web stranici. Tim elementima se posvećuje najveći deo ove sekciji u knjizi.

Velika, mala slova i praznine

Programi za prikaz sadržaja web stranica tj. web čitači, ignorišu način pisanja elemenata i atributa tj. nije od značaja da li su oni pisani malim ili velikim slovima.
Dakle, elementi se mogu pisati malim ili velikim slovima ili kombinovano. Na primer:
`<BOdY>sadržaj dokumenta</bOdy>`
je potpuno ispravno napisan kod.
Takođe, važno je zapamtiti da za prikaz HTML sadržaja nije od značaja eventualno postojanje praznih redova, tabulatora ili praznina u kodu. Da bi se prikazao novi red, tabulator ili više praznina koriste se odgovarajući HTML elementi.

Kreiranje prve HTML stranice

Koristeći neki editor teksta, na primer Notepad ili jEdit, unesite sledeći HTML kod:

Praznine u HTML kodu:

```
<html>
  <head>
    <title>   Prvi HtmL Primer   </title>
  </head>
  <body>
    <h1> Naslov br1: Uvod           </h1>
      <p> Prvi paragraph:             Pre ove recenice
              je ubaceno vise praznina
      <p>Drugi paragraf.
             Pre ove recenice postoji novi red u dokumentu.
  </body>
</html>
```

Napomena: Nazubljivanje/uvlačenje teksta koda nije bitno za web čitač, ali je veoma bitno za programera/dizajnera, pošto omogućava lakše razumevanje i ističe strukturu koda.

Pošto završite unos, sadržaj fajla sačuvati pod nazivom `primer1.html`. Obratite pažnju na obaveznu promenu ekstenzije fajla u `html`. Zatim otvorite isti fajl pomoću nekog web čitača, na primer dvostrukim klikom na fajl. Dobija se prikaz kao na slici 1.1.

Zapazite da višestruke beline, kao i novi red koji su u tekstu HTML koda, nisu vidljivi u prikazu.

Slika 1.1. Prva web stranica

Napomena: U verziji HTML5 dokument obavezno počinje deklaracijom:
`<!DOCTYPE html>`
Ova deklaracije je obavezno prvi zapis u HTML5 dokumentu, a zahtevaju je svi čitači. U verziji 4.01 ovom deklaracijom se referiše DTD dokument. Više detalja o referisanju DTD dokumenata biće reči u poglavlju koje se bavi XHTML standardom. HTML5 ne zahteva referisanje DTD specifikacije.

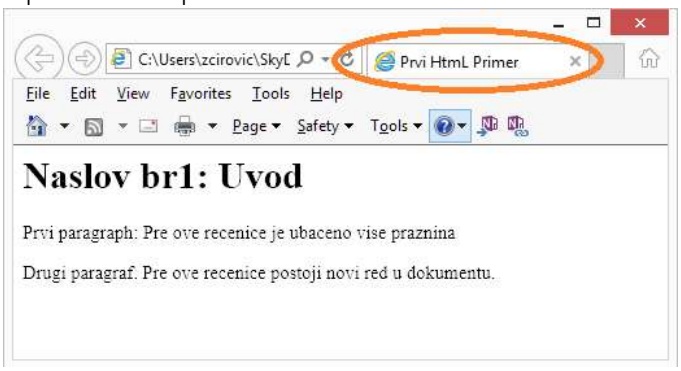
Zaglavlje dokumenta

Zaglavlje dokumenta se označava elementom **head**. Sadrži elemente koji su namenjeni web čitaču ili web pretraživačima, koji se obično nazivaju meta informacijama HTML stranice.
Svi sadržaji elemenata u zaglavlju, izuzev elementa **title**, ne mogu se videti od strane posetioca koji posmatra stranicu u web čitaču.

Naslov

Naslov web stranice se označava elementom **title** koji je podelement elementa **head** tj. smešten je između tagova **<head>** i **</head>**. Naslov se prikazuje u naslovnoj liniji prozora za prikaz web

stranice. Pozicija naslovne linije se može razlikovati od čitača do čitača. Na primer, element
`<title>  Prvi HtmL Primer  </title>`
u prethodnom primeru formira naslov dokumenta kao na slici 1.2.



Slika 1.2. Naslov HTML dokumenta

Napomena: Naslov HTML dokumenta nije isto što i ime datoteke u kojoj se nalazi dokument. Ako se naslov ne navede, dokument dobija naziv datoteke.

Meta tagovi

U zaglavlju stranice, pored naslova, nalaze se meta informacije označene tzv. meta tagovima. Najčešći korišćeni meta tagovi su:
META Description Tag je tag u kom se nalazi opis sadržaja web strane. Ovaj tag podržavaju svi pretraživači, ali ga ne koriste uvek, uglavnom sami kreiraju svoj opis. Ovaj tag je poželjno imati, samo treba voditi računa da ne sadrži više od 200 karaktera. Sa previše sadržaja u ovom tagu, neki pretraživači mogu dovesti u sumnju verodostojnost tih informacija pa bi pozicija sajta znatno opala ili ga mogu čak skroz izbrisati iz svog index-a.

`Primer meta taga name="description":`

```
<meta name="description" content="Upis studenata sa ostvarenim uslovom.">
```

META Keywords Tag sadrži niz ključnih reči i fraza koje se nalaze na vašem sajtu. Definišu se atributom `name` kome se pridružuje vrednost

"keywords". Ključne reči se navode kao sadržaj atributa content. Ovde se mogu staviti sve ključne reči koje predstavljaju sajt, ali bi bilo dobro da se biraju one ključne reči koje bi neko uneo u pretraživač kada želi da pronađe informacije koje nudi taj sajt. Bitno je da se ne stavljaju reči koje se ne nalaze na tekućoj stranici. Preporučljivo je da u ovom tagu ne bude više od 200 karaktera. Ovaj tag ne koriste svi pretraživači. Treba naglasiti da ovi tagovi iz dana u dan sve više gube na vrednosti i značaju u optimizaciji sajtova i da ih danas i ne morate implementirati zarad pozicija u glavnim pretraživačima, pošto su uveliko počeli ignorisati ovaj tag zbog njegove zloupotrebe i preterivanja u broju fraza.
Primer meta taga name="keywords":

```
<meta name="keywords" content="sport, kosarka, pravila">
```

META Language Tag definiše jezik sajta. Ovaj tag je potreban nekim pretraživačima prilikom pretrage po jeziku i treba ga koristiti.
Primer meta taga name="language":

```
<meta name="language" content="Srpski, Serbian, Serbisch">
```

META Revisit Tag definiše za koliko dana pretraživač treba ponovo da „poseti" web stranicu. Preporučljivo je za stranice koje često menjaju sadržaj.
Primer meta taga name="revisit-after "

```
<meta name="revisit-after" content="7 Days">
```

Srpska slova

Jedna od važnih primena meta tagova je definisanje kodnog rasporeda teksta u HTML dokumentu. Kada web čitač interpretira HTML dokument bez eksplicitno navedenog kodnog rasporeda, obično se koriste podaci podešavanja korisnika, bilo za ceo pretraživač ili za dati dokument, a može se izabrati podrazumevano kodiranje u zavisnosti od jezika korisnika. Za zapadnoevropske jezike, obično se koristi Windows-1252, sa ISO-8859-1. Usled pogrešnog kodnog rasporeda moguće je da se karakteri van ASCII opsega (32-

126) prikazuju nepravilno. U jezicima van engleskog govornog područja uvek se koriste znakovi van tog opsega. Pretraživači obično dozvoljavaju korisnicima da podese kodni raspored karaktera i ručno. Sve se češće, naročito za višejezičke sajtove koristi UTF-8 kodiranje, koje dozvoljava korišćenje istog kodnog rasporeda za sve jezike. Kodni rasporedi UTF-16 ili UTF-32, takođe se mogu koristiti za sve jezike. Međutim, ovi kodni rasporedi su manje efikasni u tekstovima sa velikom učestalosti ASCII karaktera. U programskim jezicima kao i u HTML dokumentima, obično se pretpostavlja bajt-orijentisano ASCII kodiranje, pa se primenjuje uglavnom UTF-8. U sledećem primeru se pokazuje način primene standarda UNICODE UTF-8 kodnog rasporeda koji podržava i srpsku latinicu i ćirilicu.

Primena UTF-8 kodnog rasporeda:

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

Napomena*:* Po novoj specifikaciji HTML5, zadavanje kodne stranice se još pojednostavljuje.

Primena UTF-8 kodnog rasporeda za HTML5:

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

Pri korišćenju određenog kodnog rasporeda treba znati kodove koji se koriste za određene karaktere. Na primer, ako se primenjuje kodni raspored 8859-2, prvo se dodaje meta tag u zaglavlje HTML dokumenta, a zatim se koriste kodovi za odgovarajuće karaktere srpskih slova iz tabele 1.1.

Tabela 1.1. Kodovi za srpska slova kodnog rasporedu 8859-2

Slovo	kod 8859-2	Slovo	kod 8859-2
Ć	262	ć	263
Č	268	č	269
Đ	272	đ	273
Š	352	š	353
Ž	381	ž	382

Primer primene kodnog rasporeda 8859-2:

```
<html>
  <head>
    <title>Pesnici</title>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
                      charset=iso-8859-2">
  </head>
  <body>
    Aleksa &#352;anti&#263; <BR>
    Milan Raki&#263;<BR>
    Jovan Du&#269;i&#263;
  </body>
</html>
```

Savet*:* Ukoliko HTML fajl sačuvate kao fajl tipa UTF-8 ili Unicode, mogu se koristiti srpski karakteri bez eksplicitnog navođenja svakog posebno!

URL

Upotreba uniformnih lokatora resursa, skraćeno URL-a (eng. Uniform Resource Locator) omogućava da se u jednom HTML dokumentu referencira: tekst, drugi HTML dokument, slika, itd. URL je string precizno definisanog formata:

`protokol://host.domen:port/stablo_dir/fajl`

U nekim slučajevima ovakva struktura URL-a može da bude proširena dodavanjem upita i fragmenta oblika: `?upit#fragment_id`, koje ćemo obraditi kasnije. Navedeni delovi stringa imaju sledeće značenje:

- `protokol` - označava protokol odnosno uslugu koja se koristi pri referenciranju fajla: `file, ftp, http gopher, news`…,
- `host` - ime računara (ili IP adresa) na kome se nalazi fajl,
- `domen` - Internet domen računara,
- `port` - u većini slučajeva 80, tako da se može izostaviti,
- `stablo_dir` – putanja do fajla kroz stablo dir,
- `upit` - sadrži podatke koji se prosleđuju programu koji se izvršava na serveru, a sadrži parove name/value

odvojene karakterom &, na primer: `?username=John&usergroup=admin`,

- `fragment_id` - označava deo ili poziciju unutar jednog resursa. Kada se koristi za HTML obično specificira sekciju ili lokaciju koja se koristi zajedno sa `anchor` tagovima koji obezbeđuju skrolovanje unutar stranice do označenog mesta. Više o URL adresiranju biće objašnjeno kasnije u posebnoj sekciji.

Telo dokumenta

Telo HTML dokumenta je definisano elementom `body`. U ovom elementu su smešteni svi HTML elementi koji definišu sadržaj i izgled stranice. Element `body` sadrži atribute koji bliže opisuju web stranicu. Inače, nijedan od tih atributa nije podržan novom specifikacijom HTML5. To znači da se umesto atributa predviđa korišćenje isključivo kaskadnih stilova.

- Pozadina stranice se definiše na dva načina. Kao:
  - slika, pomoću atributa **`background`** čija vrednost predstavlja sliku koja se prikazuje u pozadini stranice. Na primer: `<body background="URL slike">`, ili
  - boja, pomoću atributa `bgcolor` čija vrednost predstavlja pozadinsku boju stranice. Na primer: `<body bgcolor="#rrggbb">`, gde `rrggbb` predstavlja heksadekadni triplet boje: cifre rr (r = 0, 1, ... 9, a, ... f) određuju količinu crvene boje, cifre gg (g = 0, 1, … 9, a, ... f) određuju količinu zelene boje, cifre bb (b = 0, 1, … 9, a, ... f) određuju količinu plave boje). Više o bojama biće rečeno kasnije.
- Boja teksta se definiše atributom **`text`**, čija je vrednost zapisana na isti način kao i u slučaju pozadinske boje, na primer: `<body text="#rrggbb">`.

- Boja linkova se postavlja atributima **`link, alink, vlink`**:
  `<body link="#rrggbb" alink="#rrggbb" vlink="#rrggbb">`, gde je
  - `alink` - boja aktivnih linkova
  - `vlink` - boja posećenih linkova
  - `link` - boja ostalih linkova (neposećenih i neaktivnih)

`Definisanje boje teksta i pozadine:`

```
<html>
  <head>
    <title>Atributi elementa body</title>
  </head>
  <body bgcolor="#dedede" text="#323130">
    narodne poslovice <br>
    ko se dima ne nadimi....<br>
    ko rano rani....<br>
  </body>
</html>
```

Odgovarajući izgled je dat na narednoj slici.

Slika 1.3. Podešavanje atributa elementa ***body***

HTML5 napomena*:* Po novoj specifikaciji HTML5, svi atributi za definisanje prikaza u ovom elementu su izbačeni. Ova r su proizašla iz dobre prakse koja se primenjivala i u prethodnoj verziji HTML 4.01, a znači odvajanje strukture dokumenta od stilova za prikaz. HTML dokument treba da definiše strukturu podataka, a prikaz CSS

stilovima. BODY i dalje sadrži sve globalne atribute koji su primenljivi na svim elementima kao što su: id, style.

Formatiranje teksta

Jedna od osnovnih informacija koja se prikazuje na web stranici je sadržaj nekog teksta. Dužina linije teksta, veličina i oblik fonta utiču na njegovu čitljivost. Beline oko teksta mogu da istaknu strukturu ili naglase određeni deo. Zato je formatiranje teksta od velikog značaja za dobar web dizajn, pa su elementi za formatiranje teksta deo HTML standarda od prvih verzija. Ovo poglavlje je posvećeno HTML elementima i atributima koji se najčešće koriste za formatiranje teksta.

Pasusi

HTML element za označavanje pasusa (često se koristi termin paragraf) je `<p>`. Pasus sadrži: jednu ili više linija teksta, slike i druge linijske (eng. inline) elemente.
Vizuelno, tekst jednog pasusa je odvojen novim redom od ostatka teksta. Ukoliko u tekstu ima više pasusa, nije neophodno naglašavati kraj pasusa završnim tagom `</p>`, već je dovoljno navesti tag za početak novog pasusa. Na primer:

```
<p> . . . neki tekst prvog pasusa . . .
<p> . . . neki tekst drugog pasusa. . .
```

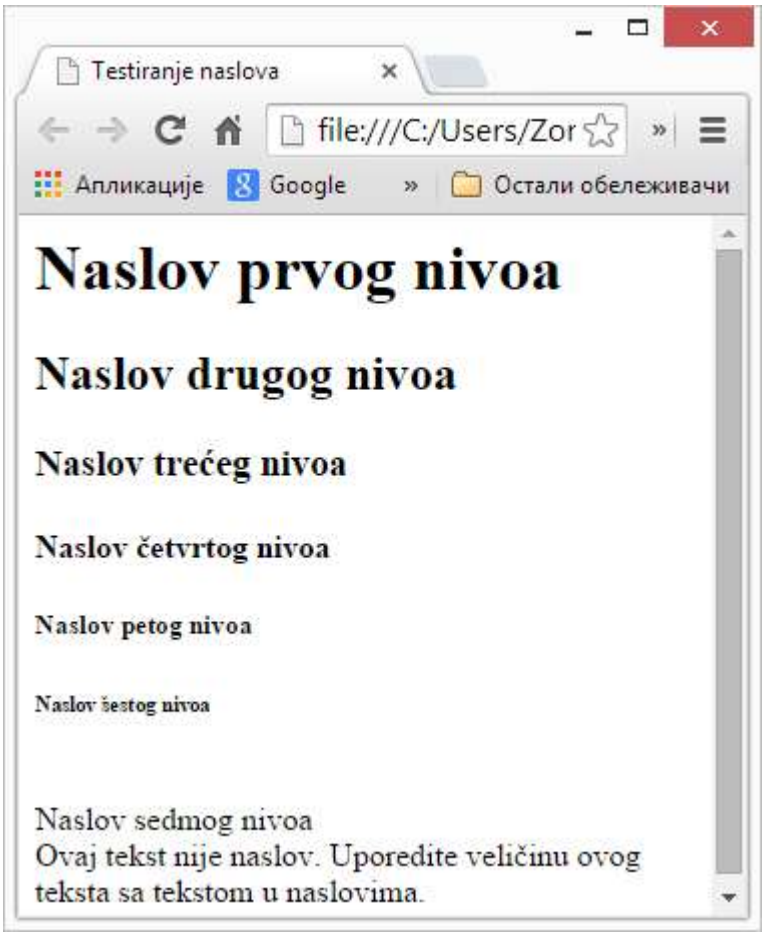
Tekst u pasusu se može poravnati na četiri načina: na levo, na desno, centrirano ili blokovski (uz obe ivice). Tekst je poravnat na levo za sve evropske jezike, ako se drugačije ne navede. Tačnije, ova opcija je podrazumevana za sve jezike koji se pišu s leva na desno.
Poravnavanje se obavlja pomoću atributa:
`align = left|center|right|justify`.
U novim sajtovima se koriste isključivo stilovi umesto ovog atributa, pa je izostala i njegova podrška u specifikaciji HTML5.

Naslovi

Postoji šest elemenata koji se koriste za označavanje naslova na stranicama. U odnosu na normalni tekst u dokumentu naslovi se obično prikazuju naglašenim fontom, uz dodavanje prazne linije pre i posle naslova.
Tag za prikaz najvećeg naslova je `<h1>...</h1>`.
Ostali idu redom:
`<h2>...</h2> <h3>...</h3> <h4>...</h4> <h5>...</h5> <h6>...</h6>`.
Veći broj je manji naslov. Naredni primer prikazuje primenu i uporedni prikaz naslova.

Uporedni prikaz naslova različitih nivoa na stranici:

```
<html>
  <head><title>Testiranje naslova</title></head>
  <body>
    <h1>Naslov prvog nivoa</h1>      <!-- najkrupniji -->
    <h2>Naslov drugog nivoa</h2>
    <h3>Naslov trećeg nivoa</h3>
    <h4>Naslov četvrtog nivoa</h4>
    <h5>Naslov petog nivoa</h5>
    <h6>Naslov šestog nivoa</h6><br> <!-- najsitniji -->
    <h7>Naslov sedmog nivoa</h7><br> <!-- ??? -->
    Ovaj tekst nije naslov. Uporedite veličinu ovog teksta sa
tekstom u naslovima.

  </body>
</html>
```



Slika 1.4. Različiti nivoi naslova na stranici

Novi red

Web čitač prikazuje jednu po jednu liniju teksta iz HTML koda. Pri tome višestruke beline se svode na jednu, a novi redovi ili tabulatori se ignorišu.

Za označavanje novog reda koristi se element **
**, eventualno **
**. Kosa crta u drugom slučaju, nije obavezna osim ako se koristi XHTML, o čemu će biti reči u posebnom poglavlju.

Kada je potrebno onemogućiti da se određeni tekst prelama u više redova tada se taj tekst označi tagovima **<nobr>** odnosno </nobr>. Na primer:

```
<nobr> Tekst koji se svakako nece prelomiti </nobr>
```

Napomena: Ukoliko je tekst prevelik da bi stao u jednu liniju prozora web čitača, na dnu prozora će se pojaviti horizontalni skrol-bar. HTML5 nudi još jedan tag. Ako treba da se tekst prelomi baš na određenom mestu, ali samo kada je širina prozora nedovoljna za prikaz, ubacuje se tag **<wbr>**. Za razliku od taga <br> koji će obavezno prelomiti red, ovaj tag će prelomiti red samo ako je to neophodno. Na primer omogućava da se prelomi neka dugačka reč na kraju reda.

Komentari

Komentarisani HTML kod se ne koristi za formiranje prikaza, ali je vidljiv u izvornom HTML kodu. Primenom komentara se može postaviti upozorenje ili dodatna informacija za kasnije ažuriranje dokumenta.
Komentar predstavlja bilo koji tekst zapisan između oznaka <!-- odnosno -->. Na primer:

```
<p><!-- uvodni pasus -->
<p><!-- napomena -->
```

Grupisanje

Grupisanje elemenata ili teksta olakšava primenu stilova i čini sajtove konzistentnim, a održavanje i razvoj sajtova bržim. U ovu grupu elemenata spadaju elementi **div** i **span** koji sami po sebi ne formatiraju tekst i zbog toga su veoma pogodni za naknadno formatiranje pomoću stilova.
Tag označava odeljak na stranici. Koristi se za pozicioniranje drugih elemenata, odnosno njihovo grupisanje.
Tag je sličan tagu , s tim što se koristi za obeležavanje dela teksta. Sam po sebi takođe ne proizvodi nikakav vizuelni efekat, sve dok se ne upotrebi u kombinaciji sa nekim stilom. Na primer:

Grupisanje teksta:

```
<div class="listing">
  <p><span class="hitno">   UVOD U INTERNET TEHNOLOGIJE
</span>, www.viser.edu.rs</p>
  <p> Osnovna znanja i razumevanje Internet tehnologija
uključujući osnovni dizajn HTML stranice.</p>
</div>
```

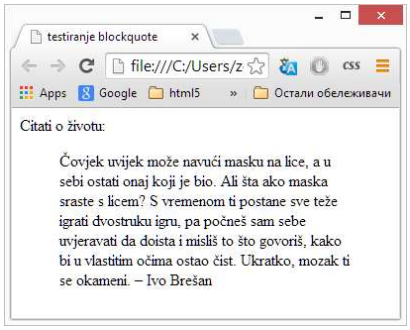
Element **div** može da sadrži atribut **align**, sa svim napomenama kao i u slučaju pasusa.

Tekst blok

Obično se koristi za predstavljanje citiranog teksta u pasusu. Tekst blok se označava elementom **blockquote**, a vizuelno je prikazan kao izdvojen blok teksta, tipično uvučen i poravnat ulevo. Na primer:

Tekst blok:

```
<p>Citati o životu:</p>
<blockquote> Čovjek uvijek može navući masku na lice, a u sebi
ostati onaj koji je bio. Ali šta ako maska sraste s licem? S
vremenom ti postane sve teže igrati dvostruku igru, pa. . .
</blockquote>
```



Slika 1.5. Tekst blok primenjen u citatu

Ovaj element u HTML5 dobija i značenje. Predstavlja sekciju koja je preuzeta (citirana) iz nekog izvora.

Kratki citat

Za označavanje kraćeg citiranog teksta koristi se HTML tag `<q>`. Element nije nov a postoji i u HTML5 verziji.

```
<p>Marko je rekao: <q>Ja ću obrisati tablu.</q></p>

<p><q>Ja ću otvoriti prozor</q>, rekla je Jelena.</p>
```

Marko je rekao: "Ja ću obrisati tablu."

"Ja ću otvoriti prozor", rekla je Jelena.

Slika 1.6. Primena kratkog citiranog teksta

Obično se Preformatiran tekst

Ako je potrebno na web stranici prikazati blok tekst kao u HTML kodu, tj. sa sačuvanim višestrukim belinama, tabulatorima, novim redovima i sa neproporcionalnim fontom, taj tekst se smešta između tagova `<pre>` i `</pre>`. Dakle, ovako označen tekst biće prikazan na ekranu bez izmena u pogledu praznina. Na primer:

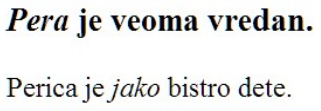
```
<pre>
    I
    B
    M
    ...
    VSER -
www.viser.edu.rs
</pre>
```

Slika 1.7. Primena preformatiranog teksta

Naglašavanje teksta

Element `<em>...</em>` koristi se da istakne deo nekog teksta (eng. *emphasized*). Vizuelni efekat ovako označenog teksta najčešće je ukošavanje, a ostale stilove tekst nasleđuje od elementa kome pripada. Različiti web čitači mogu imati različito toniranje naglašenog teksta.

```
<h3><em>Pera</em> je veoma vredan.</h3> <p>Perica je <em>jako</em> bistro dete.</p>
```



Slika 1.8. Naglašavanje teksta

Font

Element `font` se koristi za promenu vrste karaktera tj. fonta. Ovaj element nije podržan novom specifikacijom HTML5. Osnovni atributi su:

- `color = colorname|#rrggbb` – boja teksta,
- `face = familija_fonta` – naziv fonta,
- `size = number` – veličina fonta. Veći broj je veći font.

`Primena različitih fontova:`

```
<font size="4" color="red">Crveni tekst velicine 4!</font>
<font size="3" color="blue">Plavi tekst velicine 3!</font>
<font face="verdana" color="green" size="2">Verdana, zelen,
velicine 2!</font>
```

HTML5 napomena: Ovaj elemenat nema semantičku informaciju i izbačen je u celosti u HTML5 verziji.

Indeks i eksponent

HTML tekst se može formatirati tako da sadrži delove koji su u formatu indeksa ili eksponenta, koristeći elemente **sub**, odnosno **sup**. Formatirani tekst indeksa odnosno eksponenta je manji od ostatka teksta, a pozicija je pomerena dole, odnosno gore.

```
<h3>
log <sub>10</sub>10<sup>x</sup> = ? </h3>
```

$\log_{10} 10^{x} = ?$

Slika 1.9. Primena indeksa i eksponenta

Horizontalna linija

Odvajanje dela sadržaja na stranici ili naglašavanje tekstualne celine obično se vrši ubacivanjem novog reda. Ako je potrebno da se

razdvajanje uradi na još očigledniji način koristi se horizontalna linija, odnosno prazan element **hr**.
Dimenzije linije se zadaju sa dva atributa:

- **size** - debljina linije u pikselima,
- **width** - širina linije u pikselima ili %.

Na primer, HTML kodom `<hr size=3 width="55%">` dobija se linija debljine 3 piksela i širine 55% od širine prozora.
Ostali atributi su:

- `noshade` – koristi se da linija bude potpuno crna. Podrazumevano linija je zasenčena,
- `align` – horizontalno poravnavanje,
- `color` - boja linije.

HTML5 napomena: Ovaj element postoji u verziji HTML5, ali ima samo značenje tematskog prekida, dok u prethodnoj verziji znači prikaz linije. Svi atributi su izbačeni.

Praznine i specijalni karakteri

Praznine

Kako je ranije navedeno, višestruke praznine u HTML kodu se u prikazu ignorišu.
Kada su neophodne, dodatne praznine se ubacuju oznakom ** **.
Na primer, ako se u kodu navede `   ` u prikazu se javljaju tri praznine.

Specijalni karakteri

Pošto se HTML kod sastoji od tagova koji su oivičeni znacima manje i veće, ubacivanje teksta koji sadrže ove karaktere izazvalo bi grešku u interpretaciji HTML koda od strane web čitača. Zato ovi karakteri spadaju u grupu specijalnih karaktera koji se posebno predstavljaju u HTML kodu. U sledećoj tabeli 1.2. navedeni su specijalni karakteri koji se najčešće koriste.

Osim standardnih specijalnih karaktera, koji se nazivaju entitetima, postoje još i entiteti namenjeni matematičkim oznakama, grčkim slovima i slično. U nastavku navodimo tri tabele sa grupama specijalnih karaktera.

Tabela 1.2. Specijalni karakteri

HTML oznaka	karakter	HTML oznaka	karakter
<	<	"	"
>	>	'	'
&	&	€	€

HTML oznaka	karakter	HTML oznaka	karakter
∀	∀	Α	Α
∂	∂	Β	Β
∃	∃	Γ	Γ
∅	∅	Δ	Δ
∇	∇	Ε	Ε
∈	∈	Ζ	Ζ
∉	∉	©	©
∋	∋	®	®
∏	∏	€	€
∑	∑	™	™

Liste

Liste se koriste da bi se na web stranici prikazalo nabrajanje pojmova, tako da se pojmovi prikazuju jedan ispod drugog. Pojmovi koji se nabrajaju nazivaju se stavkama liste. Uz svaku stavku liste može da stoji broj, slovo ili neki grafički znak. HTML podržava tri tipa listi:

- nenumerisane liste,
- numerisane liste,
- opisne (definicione) liste.

Nenumerisane liste

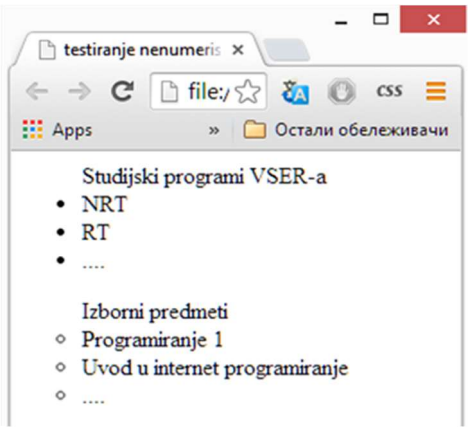
Koriste se za predstavljanje nenumerisane (eng. **U**norderd **L**ist) liste stavki koje su odvojene novim redom i označene specijalnim vodećim znakom (eng. bullet), kao što su: kružić, disk, kvadrat.
Nenumerisana lista se navodi između tagova **`<ul>`** i `</ul>`.
Stavke liste (eng. **L**ist **I**tem) su podelementi elementa liste - `ul`. Stavka se označava tagom **`<li>`**. Slično kao i u slučaju pasusa, stavka ne mora biti završena završnim tagom `</li>`, već se može završiti završnim tagom liste ili početnim tagom nove stavke.
Specijalni karakter ispred stavke liste predstavlja tip nenumerisane liste. Tip liste se definiše pomoću atributa `type` koji može uzimati vrednosti:

- `circle` - okruglo dugme,
- `disc` - ispunjeno okruglo dugme,
- `square` - kvadratno dugme.

Ovaj atribut nije podržan novom specifikacijom HTML5 pa se tip liste u tom slučaju zadaje isključivo pomoću stilova.

`Formiranje nenumerisane liste`

```
<ul> Studijski programi VSER-a
  <li>NRT<li>RT<li>....
</ul>

<ul type="circle">Izborni predmeti
  <li>Programiranje 1</li>
  <li>Uvod u internet programiranje</li>
  <li>....
</ul>
```



Slika 1.10. Nenumerisane liste

Numerisane liste

Koriste se za prikaz listi čije su stavke numerisane, (eng. **O**rdered **L**ist). Numerisana lista se označava tagovima `<ol>` i `</ol>`.
Svaka stavka u listi počinje tagom `<li>` i ne mora da ima završni tag `</li>`.
Za numerisanje stavi liste se podrazumevano koriste arapski brojevi.
Za drugačije numerisanje koristi se atribut `type`, sa mogućim vrednostima:

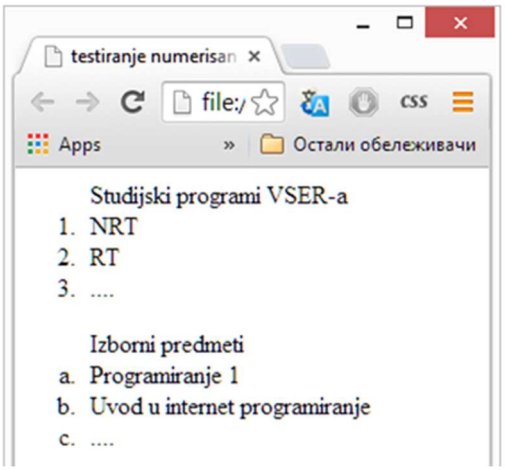
- A - velika slova,
- a - mala slova,
- I - rimski brojevi,
- i - mali rimski brojevi.

Za razliku od nenumerisane liste, tip numerisane liste je podržan novom specifikacijom HTML5.

`Formiranje numerisane liste:`

```
<ol>Studijski programi VSER-a
  <li>NRT<li>RT<li>....
</ol>
<ol type="a">Izborni predmeti
  <li>Programiranje 1</li>
  <li>Uvod u internet programiranje</li>
```

```
  <li>....
</ol>
```



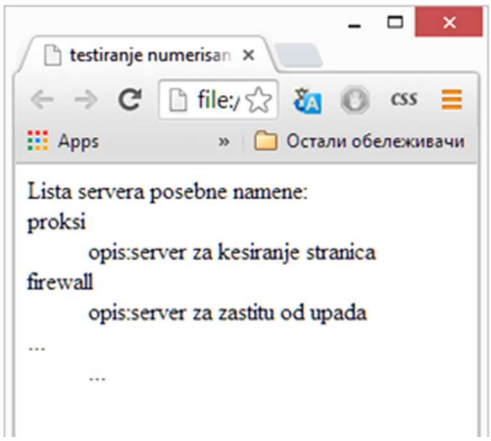
Slika 1.11. Primer numerisanih listi

Opisne liste

Koriste se za prikaz listi kada stavke ne treba označiti brojevima, slovima ili buletima, već kada iza stavki sledi odgovarajući opis. Opisna ili definiciona lista (eng. **D**efinition **L**ist) se označava HTML elementima `<dl>...</dl>`, između kojih se navode stavke. Svaka stavka sadrži **termin** koji se definiše i odgovarajući **opis**. Termin počinje obeležjem `<dt>`, a opis počinje obeležjem `<dd>`.

`Formiranje opisne liste:`

```
<dl> Lista servera posebne namene:
  <dt>proksi
    <dd>opis:server za kesiranje stranica
  <dt>firewall
    <dd>opis:server za zastitu od upada
  <dt>...
    <dd>...
</dl>
```



Slika 1.12. Primer definicione tj. opisne liste

HTML5 napomena: Polazeći od činjenice da HTML5 čuva strukturu dokumenta bez namere da se bavi formatiranjem prikaza očekivano je da atribut `compact` bude izbačen iz dalje upotrebe u označenoj i neoznačenoj listi. Istovremeno atribut `type` je nestao u neoznačenoj listi, a u označenoj ostao sa nešto umanjim skupom opcija.

Slike

Slika se u web stranicu uvodi HTML elementom:
`<img src="URLslike">`,
pri tome se navodi **obavezni** atribut `src` čija je vrednost URL putanja do slike. Slike se mogu ubaciti bilo gde u element **body**.
Treba imati u vidu da se slika ne umeće u HTML dokument, već se vrši odvajanje prostora u HTML dokumentu za sliku i povezivanje HTML dokumenta sa slikom.
Web čitači podržavaju upotrebu brojnih standardnih formata za slike. Na primer, format **GIF** se preporučuje za slike koje su crteži odnosno imaju više kontinuiranih delova iste boje, dok se **JPEG** preporučuje za slike kod kojih je prisutna veća paleta boja. Jedan od najčešće

korišćenih formata je **PNG** format koji može da reguliše nivo transparentnosti.
Dimenzije slike se zadaju atributima **`width`**, za širinu slike, odnosno **`height`** za visinu slike. Osim što se može promeniti veličina slike, **ovim podešavanjem se utiče na brzinu učitavanja slike**.
Prikaz slike u web čitaču se ne može uvek ostvariti, bez obzira na podešavanja u elementu **`img`**. Na primer, slika se ne može prikazati ako čitač nema mogućnost grafičkog prikaza, ako je prikaz slika namerno isključen ili ako je fajl naveden kao vrednost atributa `src` nedostupan. U ovakvim situacijama veoma je važno da web čitač umesto slike ostavi prazninu i odgovarajuću poruku posetiocu sajta. Za to se koristi atribut **`alt`**.

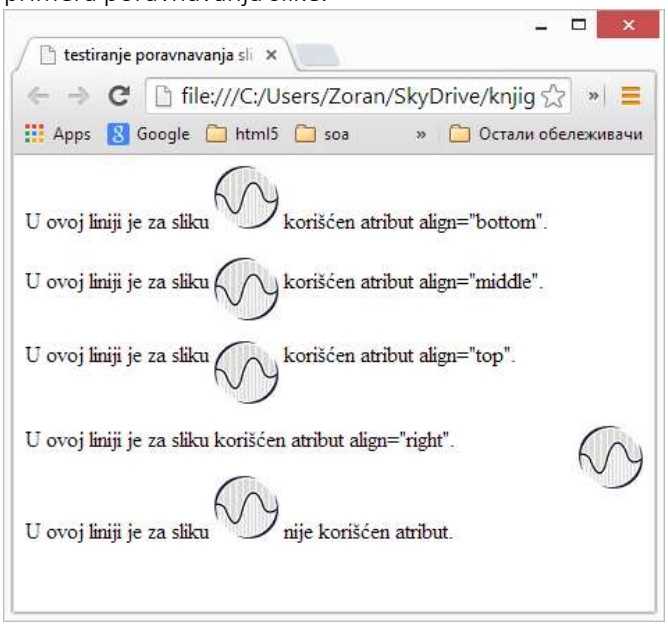
HTML5 napomena: Ostali atributi koje navodimo u nastavku nisu podržani specifikacijom HTML5, a to su: `border`, `align`, `vspace`, `hspace`.

Slika može biti uokvirena. Veličina okvira oko slike se definiše atributom **`border`**, a zadaje se u pikselima. Ukoliko se ova vrednost postavi na 0 okvir se isključuje.
Primer: U primeru koji sledi prikazana je lista koja sadrži tri slike. Slike u listi su dimenzije 48 x 48 piksela. Prva slika u originalu je u formatu JPG dimenzije 219 x 230 piksela. Druga slika je formata PNG dimenzija 293 x 300 sa transparentnom pozadinom i okvirom od 1 piksel. Treća slika ne postoji da bi se mogao prikazati izgled stranice i u tom slučaju. Sve slike se nalaze u istom folderu gde je i HTML dokument.

```
<li>
<img src="jabuka.jpg" alt="jabuka" width="48"
height="48"></img>

<li>
<img src="kruska.png" alt="kruška" width="48"
height="48" border="1"></img>

<li>
<img src="sljiva.gif" alt="šljiva" width="48"
height="48"></img>
```

Slika 1.13. Podešavanja atributa slike

Poravnanje slike u odnosu elemente koji su u okruženju, a najčešće je to tekst, određuje se atributom `align` sa mogućim vrednostima `top, middle, bottom, left, right`. Ako je poravnavanje `left` ili `right`, slika se prikazuje sa leve ili desne strane pasusa u kome je tekst, dok je tekst raspoređen oko nje. Ako se koriste vrednosti `middle`, `bottom` ili `top` onda se linija teksta ravna sa sredinom, dnom ili vrhom slike, respektivno. Na slici 1.11. prikazano je nekoliko primera poravnavanja slike.



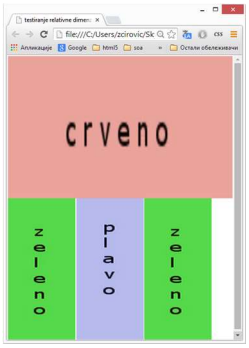
Slika 1.14. Primer primene atributa za poravnavanje u pasusu

Definisanje dodatnog prostora između slike i teksta definiše se atributima `vspace` odnosno `hspace`. Vrednosti se zadaju u pikselima. Atribut `vspace` se odnosi na marginu ispod i iznad, a `hspace` se odnosi na marginu levo i desno od slike.

Definisanje dimenzije slike na web stranici može biti urađeno relativno u odnosu na veličinu prozora. U narednom primeru primenjeno je ovo svojstvo i celokupan prozor web čitača podeljen u tri celine sa tri boje.

`Relativno podešavanje dimenzija:`

```
<img src="crveniPravougaonik.jpg"
width="100%" height="50%">

<img src="zeleniPravougaonik.jpg"
width="30%" height="50%">

<img src="plaviPravougaonik.jpg"
width="30%" height="50%">

<img src="zeleniPravougaonik.jpg"
width="30%" height="50%">
```



Slika 1.15. Definisanje dimenzija slike u procentima

Hiperveza

Hiperveza (veza ili link) je jedna od osnovnih karakteristika web sajtova. Predstavlja deo teksta ili slike koja obezbeđuje vezu sa drugom web stranicom. HTML tag za umetanje veze je `<a>`, na primer:
`<a> tekst koji je ujedno link </a>`.
Osnovni atributi hiperveze su:

- `href` – URL adresa stranice na koju se može preći,
- `id` – jedinstveni naziv tj. identifikator koji se obično koristi pri označavanju pozicije u dokumentu tzv. sidra, a koji se koristi pri referenciranju,
- `name` – određuje naziv sidra, nije podržan HTML5 specifikacijom,
- `target` – određuje gde će se otvoriti referencirani dokument.

Relativne i apsolutne putanje

Deo URL-a je putanja do fajla koji sadrži resurs koji se referencira. U opštem slučaju, putanje mogu biti apsolutne i relativne.

Apsolutna putanja je putanja do fajla koja podrazumeva putanju koja počinje sa korenom fajl sistema, na primer:
`c:/sajtovi/viser/index.html`.
Apsolutna putanja je ista bez obzira iz kog fajla se referisanje vrši.
Relativna putanja je putanja koja se navodi od foldera u kom se nalazi dokument u kom se vrši referisanje na drugi. Na primer, ukoliko je struktura web sajta:

```
/root
  index.html
  detalji.html
    img
    pozadina.jpg
    strelica.png
      photo
      foto1.jpg
      foto2.jpg
      listaprofila.html
    map
    prikaz.html
```

ako se referisanje obavlja u fajlu `index.html`, a prikazujemo sliku `pozadina.jpg`, apsolutna putanja bi bila `/img/pozadina.jpg`. Ova putanja je ista i kada bi istu sliku koristili u bilo kom drugom fajlu. Međutim, ako koristimo relativne putanje, moramo da vodimo računa gde se nalaze fajlovi jedan u odnosu na drugi. Na primer:

Tekući fajl	Referisani fajl	Relativna putanja
index.html	pozadina.jpg	`img/pozadina.jpg`
index.html	detalji.html	`detalji.html`
index.html	strelica.png	`img/strelica.png`
prikaz.html	foto1.jpg	`../img/photo/foto1.jpg`
prikaz.html	detalji.html	`../detalji.html`
prikaz.html	strelica.html	`../img/strelica.png`

listaprofila.html	prikaz.html	`../../map/prikaz.html`

Napomena: Dvotačka označava nadfolder. Primena dvotačke znači da putanja mora prvo da "izađe" iz foldera u kome se nalazi tekući fajl, a tek onda da "uđe" u foldere referisanog fajla.

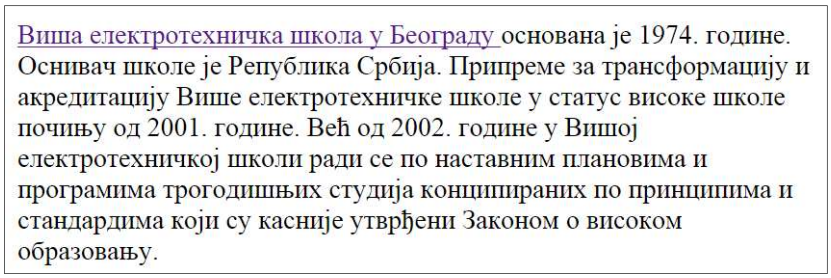
Važna napomena: Web sajt se sastoji od više dokumenata, slika i drugih fajlova koji se nalaze u jednom direktorijumu (sa poddirektorijumima). Obično su u folderima grupisani HTML dokumenti ili resursi sličnog značenja, poput slika koji se koriste. Web serveri vrše usluge za više sajtova i pri tome sajtove grupišu u direktorijume. S druge strane, sajtovi se po pravilu razvijaju u okruženju koje nije isto kao i ono na kom se postavlja konačno rešenje. Zato je važno da se omogući lako prebacivanje fajlova jednog sajta sa jedne mašine na drugu, odnosno iz jednog foldera u drugi, a da pri tome korišćeni URL stringovi budu funkcionalni. Imajući u vidu da se referisanje može obaviti koristeći apsolutno i relativno adresiranje, treba zapamtiti da treba **koristiti isključivo relativno adresiranje**.

Atribut `href`

Atribut `href` sadrži URL nekog HTML dokumenta, slike, CSS fajla i sl. sa kojim se povezuje stranica ili tekući dokument. Na primer HTML kod:

```
<a href="http://www.viser.edu.rs"> Виша електротехничка школа у Београду </a> основана је 1974. године. . .
```

formira link do početne stranice škole. Podrazumevano, linkovani tekst se prikazuje podvučeno, kao na slici 1.13.



Slika 1.16. Podrazumevani izgled hiperveze u tekstu

Napomena: Kada se navodi adresa web stranice, pre naziva domena mora se navesti **`http://`**.

Sidro

Hiperveza povezuje tekući dokument sa nekim drugim HTML dokumentom. Povezivanje se može odnositi na ceo dokument ili na tačno određenu poziciju ciljanog dokumenta. Pozicija ciljanog dokumenta naziva se sidro.
Sidro mora biti jedinstveno označeno u dokumentu. Zbog toga se označavanje sidra obavlja atributom koji čuva jedinstvenu vrednost - `id`. Označavanje sidra nije vidljivo za korisnika.

`Primeri dodeljivanja sidra:`

```
<h1>Priručnik za osnove HTML-a<a id="top"></a></h1>
...
<h1>Sadržaj priručnika<a id="sadrzaj"></a></h1>
...
<h1>Kontakt e-mail<a id="email"></a></h1>
```

Sa druge strane, u dokumentu koji sadrži linkove, svojstvu `href` se dodaje vrednost naziva sidra tj. vrednosti svojstva `id`, ispred koga navodimo znak `#`. Ukoliko je to isti dokument onda `href` sadrži samo sidro kao u primerima:

`Primeri povezivanja u kojima se navodi sidro:`

```
<a href="#top">Veza do naslova</a>
...
```

```
<a href="#sadrzaj">Prikaz sadržaja</a>
...
<a href="#email">Kontakt e-adresa</a>
```

Kada se povezivanje obavlja sa nekom pozicijom u drugom dokumentu, sidro se navodi na kraju URL-a. Na primer, ako dokument `index.html`, iz prethodnih primera, sadrži link do sidra `xyz` na stranici `detalji.html`, kod je:

```
<a href= "../detalji.html#xyz">veza do </a>
```

Atribut `target`

Atribut `target` specificira kako će se otvoriti referencirani dokument u hipervezi. Može imati vrednosti:
`_blank` – referencirani dokument se otvara u novom prozoru ili tabu,
`_parent` – u roditeljskom okviru tj. frejmu,
`_self` – u istom frejmu, ovo je podrazumevana vrednost,
`_top` – u istom prozoru ali u kompletnom, ne u frejmu,
`framename` – u imenovanom prozoru.
Značenje frejma biće kasnije razjašnjeno.

Slika kao hiperveza

Da bi slika u HTML dokumentu imala funkciju hiperveze potrebno je sliku postaviti između tagova `<a>` i `</a>`. U tom slučaju element veze dobija oblik:

```
<a href="URLcilja"> <img src="URLslike"> tekst </a>
```

Ako slika predstavlja hipervezu, okvir oko slike će biti u bojama veze koje su postavljene kao vrednosti atributa `link`, `alink`, `vlink`. Okvir može biti uklonjen ako se vrednost atributa `border` postavi na 0.

Primeri upotrebe slika kao hiperveze:

```
<a href="http://www.viser.edu.rs">
  <img alt="VSER" src="logo.jpg" width="25" height="25"
       border="1" align="middle"/> </a>
naša škola
```

```
<a href="https://www.viser.edu.rs/sservis/"
   target="_blank">studentski servis</a>
```

-  naša škola
- studentski servis

Slika 1.17. Upotrebe slike kao hiperveze

Mapiranje slika

Slika se može podeliti i svakom delu slike može se pridružiti adresa nekog odredišta. Kada posetilac sajta „klikne“ na određeni deo slike vrši se učitavanje odredišne web stranice. Ovakvo deljenje slike nazivamo mapiranjem.
Mapiranje se izvodi primenom atributa `usemap` na elementu slike i definisanjem mape koristeći element `map`. Vrednost atributa `usemap` je URL do mape koja se koristi. Na primer:

```
<img src="URLslike" usemap="URLmape#sidro">
```

URL mape se definiše standardno, na isti način kao i vrednost `href` atributa elementa za hipervezu.

Primer mapiranja

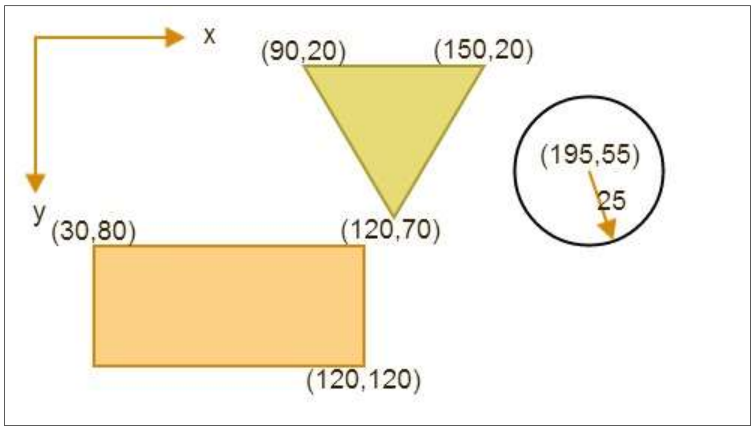
Neka je data slika `geomFigure.gif`, kao na slici 1.15 koja treba da se mapira tako da se otvaraju različite web stranice u zavisnosti na koju se geometrijsku figuru „klikne“.
Prvo je potrebno pridružiti jednu mapu slici. Neka je naziv mape `MapaGF`:

```
<img src=" geomFigure.gif" usemap="#MapaGF">.
```

Zatim treba izvršiti opis delova slike koji su od značaja. To se izvodi „mapom“ tj. posebnim elementom `map`. Naziv mape se definiše atributom `name`. Naziv odgovara vrednosti `usemap` koja se koristi pri referisanju mape u elementu `img`:

```
<map name="MapaGF">.
```

Koordinatni početak mape je gornji levi ugao slike.



Slika 1.18. Slika `geomFigure.gif` koja se mapira u `MapaGF`

Mapa se sastoji od oblasti (eng. area). Zato u elementu `map` postoje podelementi `area` čiji su atributi:

- `shape`,
- `href`,
- `coords`.

Vrednost atributa **`shape`** definiše oblik oblasti na slici, a njegova vrednost uslovljava i značenje vrednosti koje ima atribut `coords`, i može imati vrednosti:

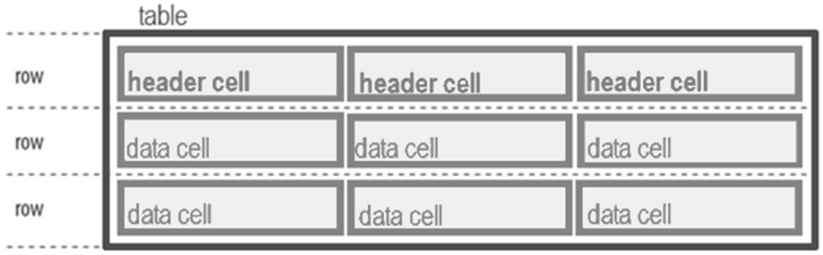
- `rect` - pravougaona oblast,
  `coords` - vrednosti su gornji levi i donji desni ugao pravougaonika u pikselima,
- `circle` - kružna oblast,
  `coords` - vrednosti su koordinate centra kruga i poluprečnik kruga u pikselima,
- `poly` – oblast je mnogougaonik,
  `coords` - vrednosti su koordinate tačaka zatvorene izlomljene linije u pikselima,
- `default` - nema koordinate i koristi se samo jednom za oblast koja obuhvata koordinate koje ne pripadaju nijednoj od već definisanih oblasti.

Napomena: Ako se dve oblasti preklapaju, prednost ima ona koja je prvo definisana.
Ako se jednoj oblasti, umesto `href` atributa, dodeli `nohref` atribut, to označava da izbor te oblasti ne prouzrokuje akciju. Ovaj atribut nije podržan specifikacijom HTML5.
Za sliku 1.15. mapiranje bi bilo sledeće:

```
<img src="geomFigure.gif" usemap="#MapaGF">

<map name="MapaGF">
  <area shape="CIRC" coords="195,55,25" href="krug.html">
  <area shape="POLY" coords="90,20,150,20,120,70"
href="trougao.html">
  <area shape="RECT" coords="30,80,120,120"
href="pravougaonik.html">
</map>
```

Tabele

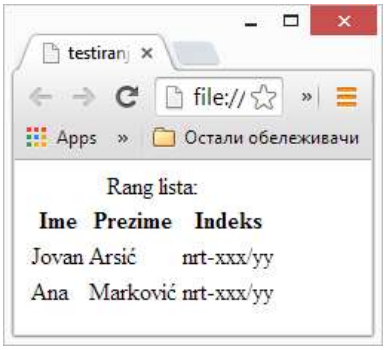
Tabele se koriste za prikazivanje podataka po redovima odnosno kolonama, kao što je prikazano na slici 1.16. Nekada su se koristile i za izradu strukture web stranice, dok se danas više koriste kaskadni stilovi.
HTML tabela se navodi između tagova `<table>` i `</table>`. Tabela ima definisan broj redova. Svaki red je zapisan između početnog i završnog taga `<tr>...</tr>`. Svaki red tabele ima isti broj ćelija koje su označene između tagova `<td>...</td>`.



Slika 1.19. Tabela sa označenim redovima i ćelijama

Tabela može imati prvi red koji čine zaglavlja (naslovi) po kolonama. Zaglavlja se vizuelno prikazuju boldovanim tekstom, a označavaju se elementom `th`, na sličan način kao i ćelije tabele koje su označene elementom `td`.
Naslov cele tabele navodi se između tagova `<caption>` i `</caption>`. Primer jedne tabele sa odgovarajućim prikazom dat je u primeru koji sledi.

```
<table>
<caption>Rang lista:</caption>
  <tr>
    <th>Ime</th>
    <th>Prezime</th>
    <th>Indeks</th>
  </tr>
  <tr>
    <td>Jovan</td>
    <td>Arsić</td>
    <td>nrt-xxx/yy</td>
  </tr>
  <tr>
    <td>Ana</td>
    <td>Marković</td>
    <td>nrt-xxx/yy</td>
  </tr>
</table>
```



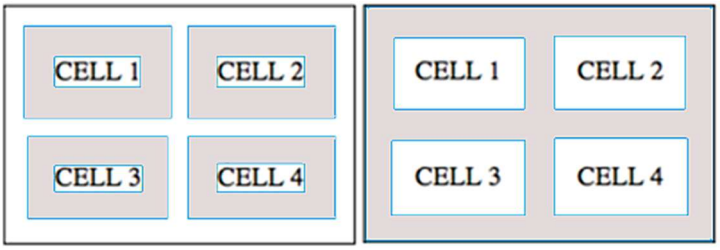
Slika 1.20. Primer jedne tabele

Obratite pažnju na detalje u prikazu. Šta bismo mogli da uradimo da bi izgled bio bolji?
Element `table` je bliže opisan atributima: `align, bgcolor, border, cellpadding, cellspacing, width,` ...

HTML5 napomena: Svi atributi koji se tiču prikaza nisu podržani novom specifikacijom HTML5. To su: `align, bgcolor, border, cellpadding, cellspacing, frame, rules, summary` i `width.`

Navodimo listu najčešće korišćenih atributa:

- `align = left|center|right` - definiše poravnavanje tabele,
- `border = broj` - definiše širinu ivice u pikselima (podrazumevano je 0),
- `cellspacing = broj` - definiše prostor u pikselima između ćelija, podrazumevano 3, uključuje `border`,
- `cellpadding = broj` - definiše prostor u pikselima između ivice ćelije i elementa tabele, podrazumevano 1,



Slika 1.21. Označen `cellpadding` odnosno `cellspacing`

- `width = broj[%]` - definiše širinu u pikselima ili procentima stranice ili frejma,
- `bgcolor = boja` - pozadinska boja tabele, takođe važi elemente reda, zaglavlja i ćelije tj. za `<tr>, <th>,` odnosno `<td>.`

Atributi za red

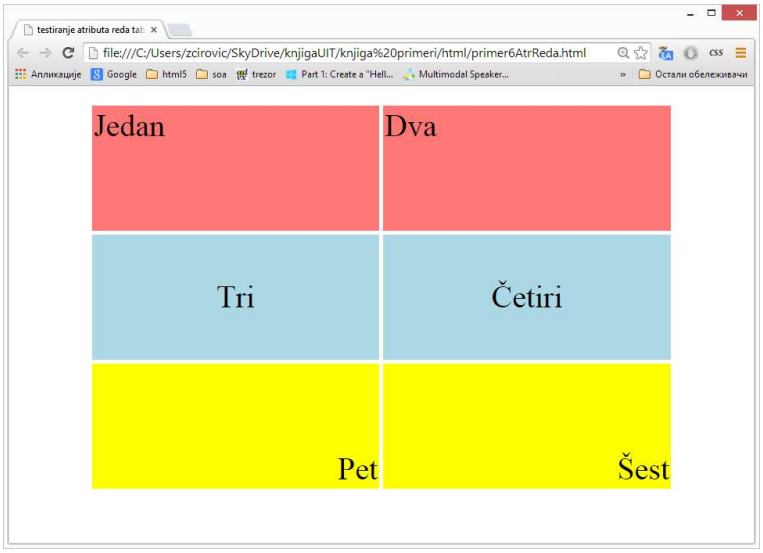
Atributi za detaljniji opis jednog reda tabele su:

- `align = left|center|right` - horizontalno poravnavanje sadržaja u redu,
- `valign = top|middle|bottom` - vertikalno poravnavanje,
- `bgcolor = color` - boja reda.

Sledi primer sa prikazom jedne tabele sa primenom navedenih atributa.

`Upotreba atributa za red tabele:`

```
<table align="center" width="300" height="200">
  <tr align="left" valign="top" bgcolor="red">
    <td>Jedan</td>
    <td>Dva</td>
  <tr align="center" valign="middle" bgcolor="lightblue">
    <td>Tri</td>
    <td>Četiri</td>
  <tr align="right" valign="bottom" bgcolor="yellow">
    <td>Pet</td>
    <td>Šest</td>
</table>
```



Slika 1.22. Primer upotrebe atributa za red tabele

Povezivanje ćelija

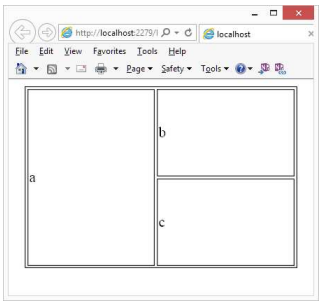
Susedne ćelije tabele mogu se spajati i na taj način formirati veće ćelije, dajući željenu strukturu tabele. Spajanje ćelija se izvodi atributima:

- `colspan = broj` - označava koliko kolona obuhvata ćelija,
- `rowspan = broj` - označava broj redova obuhvaćenih tom ćelijom.

Sledi primer povezivanja ćelija jedne tabele i odgovarajući prikaz.

`Povezivanje ćelija:`

```
<table align="center" width="300"
height="200" border="1">
 <tr>
  <td colspan="1" rowspan="2">a</td>
  <td colspan="1" rowspan="1">b</td>
 </tr>
 <tr>
  <td colspan="1" rowspan="1">c</td>
 </tr>
</table>
```



Slika 1.23. Upotreba atributa `colspan`

Frejmovi

HTML frejmovi omogućavaju predstavljanje više nezavisnih dokumenata u okviru istog prozora web čitača. Višestruki prikaz omogućava dizajnerima da zadrže neke informacije vidljivim, dok se druge skroluju ili menjaju.
Na primer, u istom prozoru jedan frejm može prikazivati statički baner, drugi navigacioni meni, a treći glavni dokument koji se može skrolovati ili menjati tokom navigacije. Inače, HTML elementi `frame` odnosno `frameset` nisu podržani novom specifikacijom HTML5.
Web čitač, ukoliko može da radi sa frejmovima, interpretira frejmove tako što deli prozor prikaza na više nezavisnih celina, od kojih svaki sadrži adresirani osnovni tag `<frameset>`. Ovaj element **zamenjuje body element** u HTML-dokumentu.
Osnovni atributi su `cols` i `rows` koji određuju broj kolona, odnosno redova u tabelarnoj podeli glavnog prozora. Dakle:

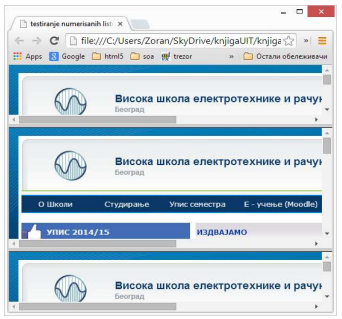
- `cols = [broj][%][*]` – širina u pikselima ili % frejmova,
- `rows = [broj][%][*]` – visina u pikselima ili % frejmova.

Poddokumenta se navode u okviru elementa `frame` čiji su atributi:

- `src` - adresa poddokumenta koji će biti prikazan,
- `marginwidth` - leva i desna margina u pikselima,
- `marginheight` - gornja i donja margina u pikselima.

`Primeri kreiranja frejmova:`

```
<frameset rows="25%,*,25%">
  <frame src="http://....">
  <frame src="http://....">
  <frame src="http://....">
        </frameset>
```



Slika 1.24. Primena frejmova na jednoj web stranici

```
<frameset rows="25%,*,25%"
     cols="*,*,50%">
    <frame src="A.html">
    <frame src="B.html">
    <frame src="C.html">
    <frame src="D.html">
    <frame src="E.html">
    <frame src="F.html">
    <frame src="G.html">
    <frame src="H.html">
    <frame src="I.html">
       </frameset>
```

A	B	C
D	E	F
G	H	I

Slika 1.25. Frejmovi u tabelarnom prikazu

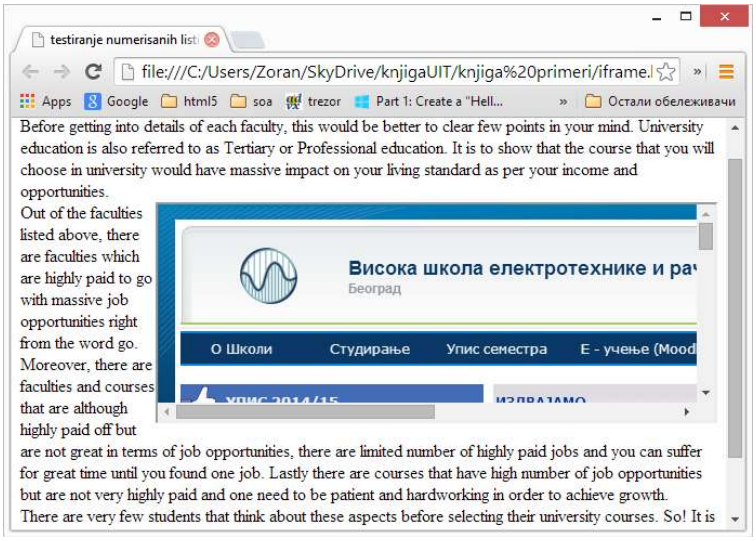
Ako se u jednu web stranicu umeće prozor za nezavistan prikaz druge web stranice, koristi se HTML element **`iframe`**, na primer:
`<iframe src="url"></iframe>` .
Atributi su:

- `src` - url stranice koja se prikazuje u umetnutom prozoru,

- `height, width` - definišu visinu i širinu prozora u pikselima ili u procentima,
- `frameborder` - specificira debljinu ivice oko prozora.

`Ubacivanje frejma:`

```
<iframe src="primer6.html" width="80%" height="200"
align="right"></iframe>
```



Slika 1.26. Umetanja frejma elementom `iframe`

Forme

HTML forme se koriste da bi se podaci od web čitača prosledili do servera. HTML forme sadrže elemente za unos podataka, kao na slici:

Slika 1.27. Elementi forme

Osnovni elementi formi su:

tekstualno polje

Primer: `<input type="text" name="username">`
`<input type="password" name="password">`
Predstavlja jednolinijsko polje za unos teksta. Polje tipa password je identično tekstualnom, ali se tekst ne prikazuje korisniku, tačnije maskiran je.

tekstualna oblast

Primer:
```
<textarea rows="3" cols="30">Poruka ..
</textarea>
```
Predstavlja kontrolu za unos teksta koja sadrži više linija.

radio dugme

Primer:
```
<input type="radio" name="pol" value="Muški">
Muški
<input type="radio" name="pol" value="Ženski">
Ženski
```
Omogućava označavanje jednog radio dugmeta iz iste grupe tj. onih koji imaju istu vrednost atributa `name`.

polje za potvrdu

Primer:

```
<input type="checkbox" name="job"
value="Posao"> Zaposlen
<input type="checkbox" name="student"
value="Studije"> Student
```

Omogućava označavanje više navedenih izbora.

lista za izbor

Primer:

```
<select name="dozvola">
<option value="a">A kategorija</option>
<option value="b">B kategorija</option>
<option value="c" selected>C
kategorija</option>
<option value="ne">Ne</option>
</select>
```

Omogućava izbor jedne stavke iz liste navedenih.

dugme za slanje

Primer:

```
<input type="submit" value="Pošalji">
```

Pritiskom na dugme vrši se slanje podataka serveru. Vrednost označava naziv dugmeta.

dugme za reset

Primer:

```
<input type="reset">
```

Pritiskom na dugme vrednosti kontrola u formi se resetuju. U ovom primeru nije naveden naziv dugmeta pa je web čitač postavio podrazumevani.

Forma može da sadrži i druge elemente osim formi, na primer blok elemente, kao što su `<p>`, `<h1>`, liste, ...
Element za označavanje forme je `form` i unutar njega se nalaze elementi za unos podataka:

```
<form>
    input elements
</form>
```

Podaci sa forme šalju se serveru na obradu. Tačnije, šalju se određenoj skripti koja je na serveru, a naziv te skripte se navodi u atribut forme `action` čiji sadržaj predstavlja URL skripte koja obrađuje podatke koji se šalju od forme. Na primer, ako se podaci šalju skripti za obradu mail-liste koja se naziva `mailinglist.php`, vrednost atributa bi bila:

```
<form action="/mailinglist.php" method="post">...</form>
```

Osim naziva skripte dodaje se i atribut `method` koji definiše način slanja parametara. Postoje samo dve metode za slanje podataka serveru.

- `post` - koristi se za manju količinu podataka i pri tome su podaci koji se šalju dostupni tj. vidljivi,
- `get` - podaci se šalju praćeni posebnim zaštitom i bez ograničenja u veličini podataka koji se šalju.

Osim osnovnih elemenata u formi se mogu naći i posebni elementi koji služe za dodatno sređivanje prikaza i poboljšavanje dizajna, kao što su `fieldset` ili `label` sa atributom `for`. O ovim elementima i atributima biće više reči u poglavlju koje se bavi stilizacijom formi.

HTML5

U prethodnom tekstu često smo naglašavali da neki element ili svojstvo ne postoji u novoj HTML5 verziji. Standard HTML5 predstavlja naslednika standarda HTML 4.01 odnosno XHTML 1.1. Sadrži nove tagove, karakteristike i API. Novi standard ima za cilj da omogući tačnije ponašanje različitih web čitača na različitim platformama, definisanje obrade grešaka i dalji razvoj jezika kako bi se olakšao razvoj web aplikacija. Specifične karakteristike HTML5 su:

- Novi strukturni elementi
- Forms 2.0 i „client-side“ validacija
- Ugrađena podrška za video i audio (<video>, <audio>)
- Canvas API i SVG
- Web skladišta

- Offline aplikacije
- Geolokacija
- Drag & Drop
- Novi komunikacioni API

HTML5 donosi 28 novih elemenata: `<section>`,`<article>`, `<aside>`,`<hgroup>`,`<header>`,`<footer>`,`<nav>`,`<figure>`, `<figcaption>`,`<video>`,`<audio>`,`<source>`,`<embed>`,`<mark>`, `<progress>`,`<meter>`,`<time>`,`<ruby>`,`<rt>`,`<rp>`,`<wbr>`, `<canvas>`,`<command>`,`<details>`,`<summary>`,`<datalist>`,`<keygen>`,`<output>`.

Struktura dokumenta

HTML5 stranica startuje obavezno sa deklaracijom DOCTYPE. Osnovna struktura je:

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

    <meta charset="utf-8" />

    <title></title>

</head>

<body>

    Ovo je ceo sadržaj moje prve HTML5 stranice

</body>

</html>
```

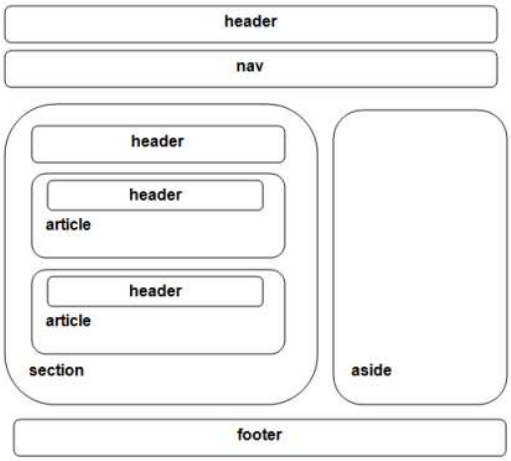
Elementi za formatiranje postoje u ovoj verziji, ali osnovni značaj je u informaciji o značenju, a ne o formatiranju izgleda. Takvi elementi su: `<b>`, `<em>`, `<i>`, `<small>`, `<strong>`, `<sub>`, `<sup>`, `<ins>`, `<del>`. Osim ovih elemenata treba dodati i `<mark>`.

.

Novi elementi

Novi elementi za osnovno strukturiranje

U prethodnoj verziji HTML 4.01 osnovni element koji se koristi za osnovno grupisanje elemenata odnosno strukturiranje stranice bio je `<div>`. Ovaj element nije nosio nikakvo značenje. Značenje je dobijao preko stilova odnosno određenih klasa. Uviđajući ovaj nedostatak nova verzija dodaje više elemenata koji nose značenje ali, kao i `<div>` element bez unapred definisanog formatiranja. Na slici 1.28. prikazana je blok šema jedne tipične strukture.



Slika 1.28. Primer osnovne strukture dokumenta

Osnovni novi elementi za definisanje strukture stranice su:

- `<nav>`

  Predstavlja glavni blok za navigaciju. U ovom elementu se grupišu linkovi ka drugim stranicama ili delovima tekuće stranice. Na primer, futeri oblično sadrže linkove, pa je očekivano koristiti ovaj element. Ipak ne treba zaboraviti da element `<nav>` ne mora i ne treba da bude korišćen na svakom mestu gde se koriste linkovi.

```
<nav>
    <ul>
        <li> <a href="/"> pocetna </li>
        <li> <a href="/dogadjaji"> dogadjaji </li>
```

```
        <li> <a href="/kontakti"> adrese </li>
    </ul>
</nav>
```

- `<header>`

Definiše zaglavlje dokumenta. Obično grupiše osnovnu navigaciju, logo, adresu… Ne mora biti korišćen samo na stranicama gde stoji na vrhu u vidu naslova. Može se koristiti i kao deo sa naslovom i datumom na blogu.

```
<body>
    <header>
        <h1>Naslov dokumenta </h1>
        <nav>
            <ul>
                <li> <a href="/"> pocetna </li>
                <li> <a href="/dogadjaji"> dogadjaji </li>
                <li> <a href="/kontakti"> adrese </li>
            </ul>
        </nav>
        Ovo je ceo sadržaj moje prve HTML5 stranice
    </header>
    <p> sadrzaj van zaglavlja </p>
</body>
```

- `<article>`

Definiše članak na stranici. Obično je to deo teksta koji predstavlja logičku celinu nezavisnu od ostatka teksta. Na primer: novinski članak, komentar, odgovor i slično.

```
<article>
    <header>
        <h1>Naslov...</h1>
        <time datetime="2015-04-05" pubdate>
            5.4.2015.
        </time>
    </header>
    <p>
```

```
        Neki članak...
    </p>
</article>
```

- `<footer>`

Slično kao i element "header", obično se referiše kao podnožje stranice. Ovih elemenata može biti više u jednom dokumentu. Može postojati futer u svakoj sekciji ili svakom članku.

```
    <article>
        <header>
            <h1>Naslov teme...</h1>
            <p>Opis teme ... </p>
        </header>
        <footer>Objavljeno datuma:
          <time datetime="2015-04-05"
pubdate>5.4.2015.</time>
        </footer>
    </article>
    <footer>
        <nav>
            <ul>
                <li> <a href="/"> pocetna </li>
                <li> <a href="/dogadjaji"> dogadjaji </li>
                <li> <a href="/kontakti"> adrese </li>
            </ul>
        </nav>
    </footer>
```

- `<aside>`

Ovaj element se koristi za odeljak koji je obično nekako povezan sa glavnim sadržajem, ali koji može biti odvojen od njega.

```
<article>
    <p>
        Виша електротехничка школа у Београду
        основана је 1974....
    </p>
```

```
    <aside>
        Студентима је пружена могућност да кроз
        међународну размену студентата...
    </aside>
</article>
```

- `<section>`

Za razliku od elementa <article> za ovaj element se obično kaže da nije samoodrživ na stranici ili dokumentu. Obično je sredstvo za deljenje stranice na tematske celine ili članka na sekcije.

```
<article>
    <h1>Predlog jelovnika</h1>
    <p>Zdrava hrana u svakom trenutku...</p>
    <section>
        <h2>Predjelo</h2>
        <p>Voće ili salata. Obavezno sa
           limunovim sokom...
        </p>
    </section>
    <section>
        <h2>Glavno jelo</h2>
        <p>Riba na roštilju u marinadi od...</p>
    </section>
</article>
```

Elementi `<progress>,<meter>`

Ovo su elementi koji se koriste za vizuelni prikaz stanja nekog procesa (napretka), odnosno veličine. Mada vizuelno slično izgledaju, treba nastojati da se primeni element odgovarajućeg značenja.

```
Progres1... <progress value="60" max="100">60%</progress> <br />
Meter1... <meter value="4" min="0" max="15">4 od 15</meter><br />
Meter2... <meter value="0.6">60%</meter><br />
```

Slika 1.29. Vizuelni izgled `progress` i `meter` elemenata

Element `<mark>`

Ovim elementom se označava deo teksta, obično sa namerom da se istakne tj. značenjem izdvoji od ostatka.

```
Виша електротехничка школа у Београду <mark>основана је
1974. godine.</mark>..
```

Виша електротехничка школа у Београду основана је 1974. godine...

Slika 1.30. Vizuelni izgled markiranog teksta

Elementi `<figure>,<figcaption>`

Ovaj element grupiše sadržaje koji mu pripadaju, kao što su: ilustracije, dijagrami, fotografije, listinzi koda,...
Dodavanjem ovog elementa slika i njen potpis čine zajedno jednu celinu. Primer jedne upotrebe slike u okviru elementa `<figure>` sa naslovom `<figcaption>`:

```
<figure>
    <img src="jabuka.jpg" alt="Jabuka">
    <figcaption> Sl.1. Jabuka</figcaption>
</figure>
```

Drugo, mada je sadržaj ovog elementa povezan sa glavnim tokom pozicioniranja elemenata na stranici, pozicija je ipak nezavisna, tj. izbacivanje ovog elementa ne menja tok u dokumentu.

```
<p>Tekst u paragrafu1.</p>
    Tekst ispred slike:
        <img height="100" width="100" src="jabuka.jpg"
             alt="Jabuka">
    Tekst u nastavku
    <p>Tekst u paragrafu2. Šta se događa ako izbrišete figure
element?</p>
```

Tekst u paragrafu1.

Tekst ispred slike:  Tekst u nastavku

Tekst u paragrafu2. Šta se događa ako izbrišete figure element?

```
<p>Tekst u paragrafu1.</p>
    Tekst ispred slike:
    <figure>
        <img height="100" width="100" src="jabuka.jpg"
             alt="Jabuka">
    </figure>
    Tekst u nastavku
    <p>Tekst u paragrafu2. Šta se događa ako izbrišete figure
element?</p>
```

Tekst u paragrafu1.

Tekst ispred slike:

Tekst u nastavku

Tekst u paragrafu2. Šta se događa ako izbrišete figure element?

Slika 1.31. Primena `figure` elementa

`Element <embed>`

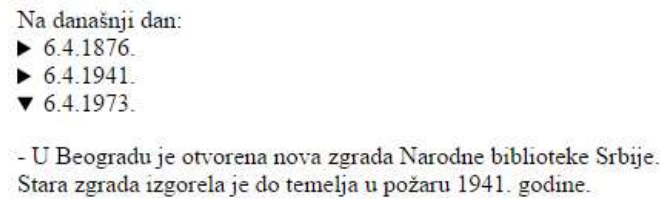
Ovaj element definiše kontejner za prikaz sadržaja podržanih od nekih plug-in komponenti (recimo za flush).

```
<embed src="animacija1.swf">
```

`Elementi <details>,<summary>`

Element `<summary>` omogućava da se sadržaj elementa `<details>` klikom na `<summary>` prikaže ili sakrije.

```
Na današnji dan:
<details>
    <summary>6.4.1876.</summary>
    <p> - Srpska vojska je ušla u ...
</details>
<details>
    <summary>6.4.1941.</summary>
    <p> - Nacistička Nemačka je u ...
</details>
<details>
    <summary>6.4.1973.</summary>
    <p> - U Beogradu je otvorena nova zgrada ...
</details>
```



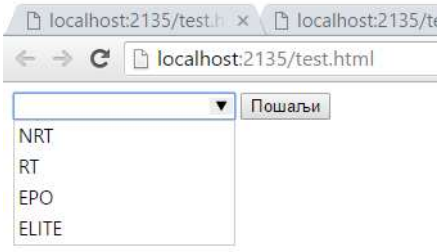
Slika 1.32. Kod i vizuelni izgled `details`/`summary` elemenata

Element `<datalist>`

Definiše listu predefinisanih opcija za input element. Ovaj tag se koristi da omogući svojstvo "autocomplete". Korisnik vidi padajuću listu preporučenih opcija. Objedinjuje padajuću listu i jednoredno tekstualno polje za unos, omogućavajući korisnicima da unesu tekst ili da iskoriste neku ponuđenu predefinisanu vrednost iz liste.

```
<!DOCTYPE html>
<html>
<body>
<form action="demo_form.asp" method="get">
    <input list="StudijskiP" name="sp">
```

```
    <datalist id="StudijskiP">
        <option value="NRT">
        <option value="RT">
        <option value="EPO">
        <option value="ELITE">
    </datalist>
    <input type="submit">
</form>
</body>
</html>
```



Slika 1.33. Kod i vizuelni izgled `datalist` elemenata

Element `ping` – deklariše putanju, kako bi url adrese registrovale kada korisnik pritisne (klikne) na link. Može se navesti više adresa. Obično je to url neke skripte.

Element `download` – pokazuje da čitač treba da preuzme datoteku.

```
<a href=http://www.jsbooks.com/content/myfile.pdf
      ping="http://www.jsbiblioteka.com/control.php" download>
      klikni ovde
</a>
```

HTML5 forme

HTML5 donosi nove elemente formi i ulazne tipove. Neke slične karakteristike postojale su i ranije na web stranicama, ali osnovna razlika je što je za to bilo neophodno uključivanje JavaScript-a. Sada

su to samostalni elementi a njihova ponašanja su ugrađena u same elemente.
HTML5 omogućava lakše formiranje formi. Nove forme imaju bolje karakteristike. Na strani klijenta je obezbeđeno prirodno rukovanje validacijama od strane web čitača, brže učitavanje stranica bez JavaScripta. Novi atributi formi su:

- `autocomplete` – Definiše se za `<form>` ili `<input>` element. Specificira da li se na formi ili ulaznom polju vrši automatizovano dopunjavanje ili ne. Kada je postoji ovaj atribut tj. uključen je, web čitač automatski dopunjuje vrednost polja na osnovu ranije već unetog od strane korisnika.

```
<form action="test.asp" method="get" autocomplete="on">
    ime:<input type="text" name="ime"><br>
    E-mail: <input type="email" name="email"
                   autocomplete="off"><br>
    <input type="submit">
</form>
```

- `novalidate` - Definiše se za `<form>` element. Ima vrednost boolean tipa (tačno/netačno). Ako je prisutan forma nema validacije.

```
<form action="nebitno.asp" novalidate>
    E-mail: <input type="email" name="email">
    <input type="submit">
</form>
```

- `autofocus` - Definiše se za `<form>` element. Slično atributu `novalidate` i ovaj atribut je tipa boolean. Kada postoji označava da jedan `<input>` element treba da primi automatski fokus kada se stranica učita. Samo jedan element može imati autofokus na jednoj stranici.
- `formaction` - Definiše se za `<input>` element. Ovaj atribut označava URL koji će obraditi ulazne kontrole kada se podaci forme pošalju. Ovaj atribut preklapa atribut `action` elementa `<form>`. Koristi se u tipovima kontrola `type="submit"` odnosno `type="image"`.

- `formmethod` - Definiše se za `<input>` element. Ovaj atribut preklapa atribut `method` elementa `<form>`. Ovaj atribut se koristi u tipovima kontrola `type="submit"` odnosno `type="image"`.
- `formnovalidate` - Definiše se za `<input>` element. Ovaj atribut je tipa boolean. Ako postoji, označava da se `<input>` elementi ne validiraju kada se šalju (submit). Preklapa ranije `novalidate` atribut `<form>` elementa. Može se koristiti samo sa `type="submit"`, `type="image"` ili kao atribut elementa `<button>`. Koristi se kada je potrebno sačuvati ali ne i poslati podatke forme.
- `formtarget` - Definiše se za `<input>` element. Označava ime ili ključnu reč koja definiše gde će se prikazati odgovor koji je primljen nakon slanja forme. Preklapa raniji atribut target. Može se koristiti sa tipovima `type="submit"`, `type="image"`.
- `required` - Definiše se za `<input>` element. Označava obavezu da Web čitač može poslati podatke samo ako su označena polja korektno popunjena. Ako je polje prazno ili nevalidno, forma neće poslati podatke, a fokus će se prebaciti na prvu nevalidnu kontrolu.

```
<form action="action1.php">
  Godine: <input type="text" name="usrname" required>
  <input type="submit">
</form>
```

- `placeholder` - Definiše se za `<input>` element. Ovim atributom postavlja se kratak tekst u kontrolu kojom se pomaže korisniku u vidu upustva šta se unosi u to polje. Obično je taj tekst prikazan nekom sivom nijansom čime se ukazuje da polje nije popunjeno.

```
<form action="action1.php">
  <input type="text" name="firstN"
         placeholder="Ime"><br>
  <input type="text" name="lastN"
         placeholder="Prezime"><br>
  <input type="submit"
```

```
value="Submit">
</form>
```

- `readonly` - Definiše se za `<input>` element. Ovo svojstvo je slično atributu `disabled`: korisnik ne može da unese vrednost tj. vrednost se samo čita. Kontrola može primiti fokus i ta vrednost se šalje sa ostalim podacima forme.

```
<input type="text" name="viser" id="rez224" readonly>
<label for="rez224"> Naslov </label>
```

- `multiple` - Definiše se za `<input>` element. Kada je prisutan označava da se višestruke vrednosti mogu uneti u kontrole forme. Ranije je postojao samo u elementu `select`. U html5 postoji i u tipovima `email` i `file`.
<form action="action_page.php">

```
Odabrane slike: <input type="file" name="img" multiple>
<input type="submit">
```

- To znači da korisnik može selektovati više od jednog fajla ili uključiti *comma-separated* više email adresa.

```
<input tymule="text" name="viser" id="rez224" readonly>
<label for="rez224"> Naslov </label>
```

Ulazni tipovi

U tabeli 1.3. prikazani su ulazni tipovi formi koji su postojali i koji su dodati u HTML5 verziji. Zapažamo da postoji veći broj novih ulaznih kontrola, tj. tipova. Radi se o potrebi za što jednostavnijom validacijom i primenom strogih pravila unosa.

Tabela 1.3. HTML5 ulazni tipovi

Postojeći	button, checkbox, file, hidden, image, password, radio, reset, submit, text
Novi	`search`, `email`, `url`, `tel`, `datetime`, `date`, `month`, `week`, `time`, `datetime-local`, `number`, `range`, `color`

- `search` – Ulazni tip koji se koristi za pretragu. Ne uključuje automatski neku pretragu.
- `email` – Koristi se za podatke tipa elektronske pošte. Može uključiti više email adresa. Web čitači rade osnovnu validaciju a smartphone uredjaji često nude dodavanje .com domena kao podrazumevanog.

```
<form action="action1.php">
  E-mail: <input type="email" name="E-pošta">
  <input type="submit">
</form>
```

- `url` – Koristi se za ulazna polja koja sadrže neki URL. Uključuje se automatska validacija.

```
<form action="action1.php">
  Adresa: <input type="url" name="adresa">
  <input type="submit">
</form>
```

- `tel` – Koristi se za ulazna polja koja sadrže telefonski broj. Za validaciju se koristi šablon telefonskih brojeva.

```
<form action="action1.php">
  Mobilni: <input type="tel" name="mobtel">
  <input type="submit">
</form>
```

- `range` – Prikazuje klizač *slider* kontrolu. Prateći atributi su: `min`, `max`, odnosno `step`.

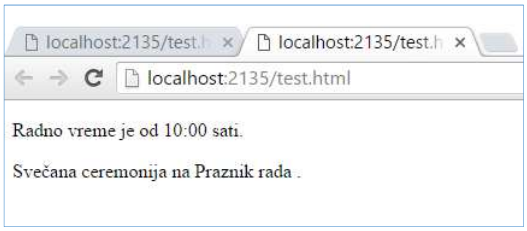
```
<form action="action1.php" method="get">
   Jačina:
   <input type="range" name="jacina" min="0" max="10">
   <input type="submit">
</form>
```

- `number` – Omogućava proveru unosa numeričkog podatka. Dobro funkcioniše sa atributima: `min`, `max`, odnosno `step`.

```
<form action="action1.php" method="get">
   <input type="number" name="ocena" min="5" max="10">
   <input type="submit">
</form>
```

- `datetime, date, time, month, week` – Omogućava unos podataka koji je datum sa vremenom, datum, vreme, mesec i nedelja u godini.

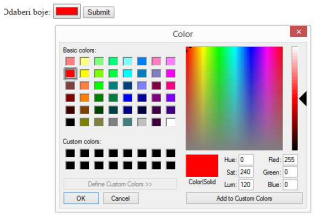
```
<p>Radno vreme je od <time>10:00</time> sati.</p>
<p>Svečana ceremonija na <time datetime="2016-05-01
20:00">Praznik rada </time>.</p>
```



Slika 1.34. Kod i vizuelni izgled `time` elemenata

- `color` –Izbor boje koristeći standardni prozor za izbor.

```
<form action="action1.php" method="get">
  Odaberi boje:
  <input type="color" name="boja" value="#ff0000">
  <input type="submit">
</form>
```



- `output` – Koristi se da bi se u formi računale neke vrednosti. Atributi su:

- `for` - Lista ID vrednosti razdvojenih razmakom čije vrednosti učestvuju u izračunavanju.
- `form` - Sadrži naziv forme koja je vlasnik ovog (output) elementa. Vrednost mora biti ID forme u istom dokumentu. Ovaj atribut nam omogućava da output element bude izvan forme na koju se odnosi.
- `name` - Naziv elementa.
  Napomena: Nije podržan od IE i Edge čitača

```
<form onsubmit="return false" oninput="o.value =
parseInt(a.value) + parseInt(b.value)">
  <input name="a" type="number" step="any"> +
  <input name="b" type="number" step="any"> =
  <output name="o"></output>
</form>
```

Audio i video

HTML5 uvodi ugrađenu podršku za multimedijalne sadržaje preko elemenata `<audio>` i `<video>` omogućavajući laku ugradnju ovih sadržaja u HTML dokumente.

- `audio` – Fajl koji sadrži audio zapis se definiše posebnim elementom `<source>`. Atributi su:
  - `controls` kojim se definiše da li se kontrole (*play*, *pause*, *volume*) prikazuju ili ne.
  - `autoplay` – vrednost `autoplay`. Označava automatsko startovanje.

  Primer:

```
<!DOCTYPE<html>
<html>
<body>
    <audio controls>
        <source src="pesma1.mp3" type="audio/mpeg">
         Vas web čitač ne podržava audio element
    </audio>
</body>
</html>
```

Slika 1.35. Kod i vizuelni izgled audio elemenata

- `video` – Fajl koji sadrži audio zapis se definiše elementom `<source>`. Atributom `controls` definiše se prikaz kontrola, a `autoplay` automatsko pokretanje, identično kao i kod `<audio>` elementa. Novi atributi su:
  - `heght`, `width` – vrednost u pikselima. Definiše visinu i širinu kontrole.

```
<!DOCTYPE html>
<html>
<body>
    <video width="320" height="240" autoplay>
        <source src="movie.mp4" type="video/mp4">
    </video>
</body>
</html>
```

Pitanja za proveru znanja

1. Šta predstavlja skraćenica W3C?
2. Šta znači HTML?
3. Napišite osnovnu strukturu HTML dokumenta.
4. Kakva je razlika između HTML elementa i taga?
5. Kako se pišu komentari u HTML kodu?
6. Navedeni primeri HTML kod-a su netačni. Objasnite zbog čega.
   a. <img "logo.jpg">
   b. <a href="strana2.html">neki tekst</a href="strana2.html">
   c. <p>Ovo je novi paragraf<\p>

7. Kakva je razlika odnosno sličnost u primeni elementa za označavanje pasusa odnosno novog reda?
8. Kojim elementom se praznine napisane u kodu na isti način prikazuju na web stranici?
9. Napisati po jedan primer za sve vrste HTML lista.
10. Kako se ubacuju dodatne praznine u prikazu, a kako specijalni karakteri „<" i „>"?
11. Kojim atributom se označava jedinstven element u dokumentu?
12. Koji atributi moraju biti uključeni za svaki **`img`** element?
13. Zašto je neophodno uključiti alternativni tekst za slike?
14. Koji su osnovni elementi za HTML tabelu?
15. Koje su specifičnosti HTML5 standarda?
16. Zašto su uvedeni novi strukturni elementi?
17. Navesti primer jedne organizacije dokumenta zasnovan na novim strukturnim HTML5 elementima.
18. Navesti primer nekog elementa odnosno atributa koji su isključeni iz HTML5 verzije i objasniti svojim rečima zašto.

Rezime

HTML osnovni dokument:

```
<html>
  <head>
  <title>
    Naziv dokumenta
  </title>
  </head>
```

```
  <body>
    Vidljivi tekst
  </body>
</html>
```

Elementi naslova:

```
<h1>Najveci naslov</h1>
<h2> . . . </h2>
<h3> . . . </h3>
<h4> . . . </h4>
<h5> . . . </h5>
<h6>Najmanji naslov</h6>
```

Tekstualni elementi:

```
<p>Jedan pasus</p>
<br> (novi red)
<hr> (horizontalna linija)
<pre>Preformatiran tekst</pre>
<em>Naglašen tekst</em>
<b>Podebljan tekst</b>
<i>Iskošen tekst</i>
<!—Komentar -->
<blockquote> Citiran tekst</blockquote>
```

Veze:

```
<a href="http://www.viser.edu.rs/">Naša škola</a>
```

```
<a href="http://www.viser.edu.rs /"><img src="logo.jpg"
alt="viser"></a>
```

Imenovana veza:

```
<a name="sadrzaj">Sadržaj</a>

<a href="#sadrzaj">Skoci na sadržaj</a>
```

Liste:

Nenumerisana lista:

```
<ul>
  <li>Stavka 1</li>
  <li>Stavka 2</li>
</ul>
```

Numerisana lista:

```
<ol>
  <li>Prva stavka</li>
  <li>Druga stavka</li>
</ol>
```

Definiciona lista:

```
<dl>
<dt>Prvi pojam</dt>
<dd>Definicija prvog pojma</dd>
<dt>Drugi pojam</dt>
<dd>Definicija drugog pojma</dd>
</dl>
```

Tabele:

```
<table border="1">
```

```
  <tr>
    <th>naslov kol. 1</th>
    <th>naslov kol. 2</th>
  </tr>
  <tr>
    <td>sadrzaj 1,1</td>
    <td>sadrzaj 1,2</td>
  </tr>
</table>
```

Frejmovi:

```
<frameset cols="25%,75%">
  <frame src="stranica1.html">
  <frame src="stranica2.html">
</frameset>
```

Forme:

```
<form action="http://www.viser.edu.rs/akcija.jsp"
method="post/get">
  <input type="text" name="prezime">
  <input type="password">
  <input type="checkbox" checked="checked">
  <input type="radio">
  <input type="submit">
  <input type="hidden">
  <select>
    <option>Jabuke
```

```
    <option selected>Banane
    <option>Kruške
  </select>
  <textarea name="Komentar" rows="10" cols="10"></textarea>
</form>
```

Deo 2: XML, XHTML

- Uvod u XML
- Dobro formiran dokument
- Validacija dokumenata
- DTD
- XHTML
- Specifikacija XHTML-a

Drugo poglavlje je posvećeno sticanju osnovnih znanja u vezi XML standarda, odnosno XHTML-a specifikacije. Najpre se daju osnove standarda, definišu se pravila dobro formiranog dokumenta. Zatim se uvodi pojam validacije dokumenta i definišu pravila dokumenata za definicije tipova – DTD. Na kraju, uz pomoć XML-a i DTD-a definiše se specifikacija XHTML sa posebnim isticanjem razlika u odnosu HTML. XML je standardizovan jezik, a za njegovu standardizaciju brine W3C konzorcijum. Danas je to *de facto* standard za opis sadržaja i strukture dokumenata. Takođe, to je univerzalni standard pri razmeni dokumenata na web-u, a ugrađen je i u nove programske jezike. Detaljnije izučavanje XML specifikacije prevazilazi predviđeno gradivo ovog udžbenika, zato će u ovom poglavlju biti proučene osnove XML specifikacije odnosno XHTML kao jedna njegova primena.

Uvod u XML

Naziv XML je skraćenica od engleskih reči: *E**X**tensible **M**arkup **L**anguage.* XML je jezik za označavanje (eng. markup) koji se zasniva na primeni tagova, kao i HTML. Za razliku od HTML jezika, XML je

proširiv (eng. extensible) tj. omogućava definisanje novih tagova, pa i celih specifikacija tj. drugih jezika za označavanje.

Elementi

Osnovna struktura XML dokumenta je veoma slična HTML dokumentu. XML dokument je organizovan pomoću tagova. Elementi mogu biti:

- složeni tj. kontejner elementi, koje čini par tag-ova (početni i krajnji tag) sa sadržajem i
- prosti tj. prazni elementi.

Završni tag počinje kosom crtom, kao kod HTML tagova. Ukoliko je element prost, za razliku od HTML-a, tag mora da se završava sekvencom />, na primer <br/>.
Takođe, za razliku od HTML jezika, elementi u XML dokumentu nisu unapred ograničeni na određeni skup. XML dokument je bilo koji dokument koji zadovoljava XML pravila, a korišćeni elementi tj. tagovi se definišu. Značenje korišćenih elemenata u dokumentu je definisano aplikacijama koje koriste XML dokumente i nije predmet XML specifikacije. Na primer, ako je XML dokument istovremeno XHTML dokument, onda je podrazumevana aplikacija za obradu dokumenta web čitač.
Kao i elementi HTML jezika, tako i XML elementi mogu da sadrže atribute. Vrednosti atributa moraju uvek biti unutar znaka navoda. Moguće je koristiti jednostruke ili dvostruke znake navoda. Na primer:
`<student ime="Aca" prezime="Petrović" smer='NRT' >` .

Dupli znaci navoda su češći, mada je nekada neophodno koristiti obe vrste navoda, na primer:
`<ime='Bora "Čorba" Đorđević'>` .

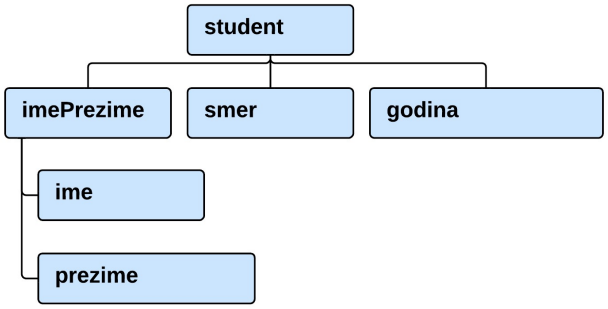
Hijerarhijska struktura

XML dokument sadrži tačno jedan element koji nema nadelement tj. roditeljski element. To je uvek prvi element u dokumentu koji sadrži sve druge elemente. Ovaj element se naziva korenskim (eng. *root*). Svi ostali elementi imaju samo jedan roditeljski element i mogu imati više podelemenata (dece elemenata). Zato se za XML dokument, tj.

sadržaj u dokumentu, kaže da je hijerarhijski organizovan. Na primer, ako je XML dokument:

```
<student>
  <imePrezime>
    <ime>Jovan</ime>
    <prezime>Petrović</prezime>
  </imePrezime>
  <smer>NRT</smer>
  <godina>2</godina>
</student>
```

njegova hijerarhijska struktura se može prikazati kao na slici 2.1.



Slika 2.1. Hijerarhijska organizacija u XML dokumentu

Sadržaj elementa

XML element može da sadrži:

- tekst,
- druge elemente ili
- zajedno tj. tekst sa drugim elementima.

Ako element sadrži tekst onda treba voditi računa da se u tekstu ne mogu pojavljivati proizvoljni karakteri, slično kao i u HTML jeziku. Na primer, ne bi bilo moguće razdvojiti oznake od teksta ako u tekstu postoje karakteri kao što su znakovi manje (<) odnosno veće (>). Za označavanje ovakvih karaktera koriste se zamene tj. **reference entiteta**. Reference se ubacuju u tekst dokumenta, a XML aplikacije

pri parsiranju dokumenta „znaju“ da se na tim mestima zamenjuju odgovarajući entiteti. U tabeli 2.1 dat je spisak nekih često korišćenih referenci.

Tabela 2.1. Često korišćene reference

HTML oznaka		Opis
<	<	Manje od
>	>	Veće od
&	&	Ampersend
'	'	Jednostruki navod
"	"	Dvostruki navod

Za razliku od HTML-a, u XML dokumentu praznine tj. **prazan prostor** su od značaja. XML aplikacije mogu obrađivati na različite načine prazan prostor, ali treba zapamtiti da XML specifikacija čuva praznine ravnopravno kao i ostale karaktere.
Sadržaj elemenata čine i atributi sa pripadajućim vrednostima.
Osim podataka, u XML dokumentu se mogu naći i dodatne informacije koje pomažu razumevanju i održavanju dokumenta. To su XML **komentari**. Oni se pišu između znakova `<!--` odnosno `-->`. Komentar jedino ne može sadržati sekvencu `--`, sve ostale markup karaktere može.

Odeljak CDATA

Ukoliko XML dokument sadrži običan tekst, na primer duži novinski članak, koji sadrži specijalne karaktere, dokument postaje teško čitljiv i još teži za održavanje. U takvim situacijama može se koristiti odeljak CDATA koji obezbedjuje da se proizvoljni sadržaj u odeljku tretira kao sirovi tekst, tj. aplikacije koje obrađuju dokument neće parsirati sadržaj ovog elementa, već ga mogu koristiti samo kao originalan. Na primer:

```
<p>Primer wpf koda:</p>
<![CDATA[
<Grid>
  <Label Content="{Binding ElementName=textBox1,Path=Text}"
Name="label1" VerticalAlignment="Top" Width="159" />
```

```
    <TextBox   Height="23"   HorizontalAlignment="Left"
VerticalAlignment="Top" Width="120" />
</Grid>
]]>
```

Instrukcije obrade

Instrukcije obrade se koriste da bi pružila određena informacija aplikacijama za obradu XML dokumenata. Informacija može biti način obrade dokumenta ili način prikaza. Instrukcija obrade sastoji se iz dva dela, to su (i) naziv instrukcije (target) i (ii) podatak za instrukciju (data):
<? target data ?>.

Target mora da zadovolji ista pravila za naziv kao i XML elementi, odnosno atributi. Unutar konteksta instrukcije ignorišu se HTML tagovi. Zato, instrukcije obrade mogu imati potpuno drugačiju sintaksu i semantiku od samog XML-a. Mogu imati efektivno neograničenu količinu teksta. Na primer, PHP smešta velike programe u instrukcije za obradu:

```
<?php
mysql_connect("baza2.viser.edu.rs", "administrator",
"lozinka"); $result = mysql("HR", "SELECT ime, prezime FROM
Studenti Order By prezime ");
      $i = 0;
      while ($i < mysql_numrows ($result)){
        $fields = mysql_fetch_row($result);
        echo "<ime>$fields[0] $fields[1]    </ime>\r\n"; $i++;
      }
      mysql_close( );
?>
```

Instrukcije obrade spadaju u označavanje, tj. nisu elementi. Mogu se pisati na bilo kom mestu u dokumentu osim unutar oznaka. Mogu se pisati pre i posle korenskog elementa.

Deklaracija XML dokumenta

Svaki XML dokument mora da sadrži XML **deklaraciju**, tj. instrukciju obrade, kojom se dokument identifikuje kao XML dokument. Ovo je prvi red u dokumentu. Osnovni oblik XML deklaracije je:
`<?xml version ="1.0"?>`.
Deklaracija može da sadrži atribute za definisanje znakovnog koda u kome je XML dokument napisan , na primer UTF-8 - kompresovana verzija Unicode-a ili UTF-16 - Unicode.

```
<? xml version ="1.0" encoding= "UTF-8" ?>
```

Dobro formiran dokument

Sažimajući W3C preporuke, kažemo da je **dobro formiran** onaj XML dokument koji poštuje sledeća pravila:

- Postoji XML deklaracija.
- Dokument sadrži jedan i samo jedan korenski element u kom su ugnježdeni svi ostali elementi i njihovi sadržaji.
- Svi elementi i atributi u dokumentu moraju da budu sintaksno ispravni.
- Element mora imati završni tag (<...> ... </...>) ili u slučaju praznog elementa koji nema sadržaj, na kraju taga mora stajati oznaka da je to ujedno i završni tag <.../>.
- Elementi moraju biti ugnježdeni.
- Imena elemenata u XML-u su „case-sensitive“ tj. elementi se razlikuju ukoliko se razlikuje veličina slova u njihovom nazivu.
- Pri dodeljivanju imena moraju se poštovati određena pravila. Na primer, naziv elementa ne može početi sa „xml“, mora početi slovom, ne može imati beline.
- Sve vrednosti atributa moraju biti u okviru navodnika.

`Primer dobro formiranog dokumenta:`

```
<?xml version='1.0' encoding='UTF-8' ?>
<p:student xmlns:p='www.viser.edu.rs-uit2014:primeri'>
  <ime>Jovan Petrović</ime>
  <!—novi student-->
  <godina>1</godina>
  <indeks smer='nrt' >nrt-xx/yy</indeks>
</p:student>
```

Primer loše formiranog dokumenta:

```
<?xml version='1.0' encoding='UTF-8' ?>
<p:student>
  <ime>Jovan Petrović</ime>
  <!—novi student-->
  <godina>1 <b><i>(prva godina)<b><i></godina>
  <indeks smer='nrt' smer='rt' >nrt-xx/yy</indeks>
</p:student>
```

Validacija dokumenata

Postoje dva standarda za validaciju XML dokumenta:
• DTD validacija (eng. Document Type Definiton),
• validacija šemama ili XSD validacija (eng. XML Schema Definition).
XSD validacija je novijeg datuma, daje veće mogućnosti i u novim aplikacijama se uglavnom ona primenjuje. Ipak, postoje veoma značajne praktične primene DTD validacije, uključujući i XHTML koji ćemo u nastavku poglavlja obraditi.

DTD validacija

Pomoću DTD dokumenta mogu se uvesti ograničena za podatke nekog XML dokumenta, tj. može se definisati struktura podataka. DTD dokument je javan i dostupan za korisnika određenih XML

dokumenata. Pošto je standardizovan, DTD dokument je i potpuno "razumljiv" za obe strane tokom razmene podataka. DTD omogućava:

1. automatizaciju provere ispravnosti dokumenta,
2. višestruku upotrebu definisanih delova dokumenta.

DTD nije XML dokument. Može biti potpuno odvojen od dokumenta na koji se primenjuje, a može biti i deo tog dokumenta. Pomoću DTD-a definišu se:

- elementi koji se pojavljuju u dokumentu,
- način ugnježdavanja elemenata,
- skup dozvoljenih, zahtevanih i podrazumevanih atributa.

Pridruživanje DTD-a XML fajlu

DTD deklaracije se pridružuju jednom XML dokumentu na više načina:

- **DTD je poseban dokument** u odnosu na XML. Pri tome XML dokument sadrži URL u kome je DTD. Sintaksa je: `<!DOCTYPE element SYSTEM "URI">`. Na primer:

  `student.xml:`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE imePrezime SYSTEM "student.dtd">
<imePrezime>
  <ime>Jovan</ime>
  <prezime>Petrović</prezime>
</ imePrezime >
```

  `student.dtd`

```
<!ELEMENT imePrezime (ime,prezime)>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT prezime (#PCDATA)>
```

  Kada je neki DTD dokument javan onda se umesto ključne reči SYSTEM koristi PUBLIC, na primer:
  `<!DOCTYPE element PUBLIC identifikator "URI">`.
  Primer koji navodimo je jedan od XHTML standarda:

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

U ovom slučaju radi se o javnoj DTD specifikaciji koja, čak u slučaju da nema konekcije, a zahvaljujući identifikatoru, može biti prepoznata od strane aplikacija i primenjena.

- DTD deklaracije su deo XML dokumenta. Sintaska je: `<!DOCTYPE element [deklaracije]>`. Na primer: `student.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE imePrezime [
<!ELEMENT imePrezime (ime,prezime)>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT prezime (#PCDATA)>
]>
<imePrezime>
  <ime>Jovan</ime>
  <prezime>Petrović</prezime>
</imePrezime>
```

- Kombinacijom prethodna dva slučaja, tj. kada postoje DTD deklaracije u posebnom dokumentu i u XML dokumentu.

Definisanje elemenata

Sintaksa za deklaraciju elemenata je:
`<!ELEMENT nazivElementa (sadrzaj)>`. Sadržaj elementa može biti:

- **#PCDATA** – sadržaj je tekst, ali koji se parsira. Na primer, ako je DTD: `<!ELEMENT greeting (#PCDATA)>`, validni XML kod je:

```
<greeting>Hello World!</greeting> ili
<greeting> <![CDATA[G'day!]]> </greeting>
```

- **EMPTY** – prazan element. Ne može da sadrži podatke. Može da ima samo atribute. Na primer, za deklaraciju:

`<!ELEMENT br EMPTY>`, ispravan XML kod je: `<br></br>` ili `<br/>`.

- **ANY** - bilo koji element koji je deklarisan u DTD-u može biti sadržaj. Na primer za DTD kod:

```
<!ELEMENT nrt ANY>
<!ELEMENT student (#PCDATA)>
<!ELEMENT nastavnik (#PCDATA)>
```

validan XML bi bio:

```
<nrt>
  <student>Jovan J.</student>
  <nastavnik>Verica V.</nastavnik>
</nrt>
```

ili

```
<nrt></nrt>
```

ili

```
<nrt>pera, luka, mika</nrt>
```

- Tačno definisani podelementi sa strukturom. Sintaksa je `<!ELEMENT naziv (struktura)>`. Struktura podelemenata je definisana tabelom 2.2.

Tabela 2.2. Strukture podelemenata

Sekvenca	<!ELEMENT ime **(a,b)**>
Izbor	<!ELEMENT ime **(a|b)**>
Jedan	<!ELEMENT ime **(a)**>
Jedan ili više	<!ELEMENT ime **(a) +**>
Nula ili više	<!ELEMENT ime **(a) ***>
Nula ili jedan	<!ELEMENT ime **(a) ?**>

Na primer, za DTD:

```
<!ELEMENT imePrez ((ime,pre)|(pre,ime))>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT pre (#PCDATA)>
```

validni XML dokumenti bi bili:

```
<imePrez>
  <ime>Jovan</ime>
  <pre>Petrović</pre>
</imePre>
```

```
<imePrez>
  <pre>Petrović</pre>
  <ime>Jovan</ime>
</imePre>
```

- Mešavina elemenata i teksta opisuje elemente koji sadrže tekstualne podatke i/ili druge elemente. Na primer:

```
<!ELEMENT indeks (#PCDATA)>
<!ELEMENT napomena (#PCDATA|indeks)*>
```

Prva deklaracija definiše element `indeks` čiji je sadržaj sastavljen samo od karaktera. Drugi element `napomena` sastoji se od teksta ili od elementa `indeks`, koji se mogu javiti nula ili više puta. Na primer, ispravan XML kod bi bio:

```
<napomena>svi upisani studenti …</napomena>
<napomena>
  Studentu <indeks>abc-xx/yy</indeks> omogućava
  se izlazak na ispit…
</napomena>
```

Definisanje atributa

DTD omogućava da se za određeni XML element definiše lista atributa. Sintaksa za listu atributa je:

```
<!ATTLIST elemenat atribut tipAtr podrDek
                   atribut tipAtr podrDek >
```

pri čemu je `tipAtr` tip atributa, a `podrDek` podrazumevana deklaracija atributa. Najčešće korišćeni tipovi atributa su dati u tabeli 2.3.

Tabela 2.3. Tipovi atributa

CDATA	Standardni tekst
(en1|en2|..)	Jedna vrednost od navedenih u listi
ID	Jedinstvena vrednost
IDREF	Jedinstvena za drugi element
NMTOKEN	Vrednost je XML ime

ENTITY	Vrednost je entitet

Najčešće korišćene podrazumevane deklaracije su date u tabeli 2.4.

Tabela 2.4. Podrazumevane deklaracije atributa

#REQIRED	Atribut mora postojati.
#IMPLIED	Nije obavezan i nema podrazumevane vrednosti.
value	Predstavlja podrazumevanu vrednost ako vrednost ne postoji.
#FIXED value	Ako atribut postoji, njegova vrednost mora da odgovara parametru attribute-value.

U narednom primeru dat je DTD dokument sa jednim validnim i jednim nevalidnim XML dokumentom:

DTD dokument:

```
<!ELEMENT proj (#PCDATA)>
<!ATTLIST proj broj CDATA #REQUIRED>
<!ATTLIST proj napomena CDATA #IMPLIED>
<!ATTLIST proj ocena (loše|dobro|odlično) "dobro">
<!ATTLIST proj tipIzrade CDATA #FIXED "pojedinačno">
```

Primer validnog XML dokumenta:

```
<proj broj="1">Primer sajta</proj>
<proj broj="1" ocena="odlično"></proj>
<proj broj="1" tipIzrade="pojedinačno"></proj>
<proj broj="1 sleeve" collar="button-down"></proj>
```

Primer nevalidnog XML dokumenta:

```
<proj></proj>
<proj broj="1" ocena="dobro odlično"></proj>
<proj broj="1" tipIzrade ="samostalno"> </proj>
```

Definisanje entiteta

Kao i kod HTML-a specijalni karakteri kao što su <, odnosno >, moraju se u tekstu zameniti specijalnim sekvencama tzv. referencama entiteta: `<`, odnosno `>`.

Referenca entiteta ima tri dela: (i) ampersend, (ii) naziv entiteta i (iii) tačku-zarez.
Druga primena referenci je za definisanje skraćenica u nekom tekstu.
Primer definisanja entita:

```
<!ENTITY naziv "Visoka škola elektrotehnike i računarstva">
<!ENTITY adresa "Vojvode Stepe 283, Beograd">
<skola>&naziv; &adresa;</skola>
```

Entiteti mogu biti spoljašnji. U slučaju spoljašnjih entiteta najčešće se radi o poznatim javnim referencama koje mogu biti zajedničke za veći broj dokumenata. Promena na spoljašnjim entitetima automatski znači promenu na svim dokumentima koji referišu te entitete. Spoljašnji entiteti mogu biti privatni ili javni. Privatni se označavaju ključnom reči SYSTEM i obično se koriste od strane jednog ili grupe autora. Javni entiteti označavaju se sa PUBLIC i najčešće su široko primenljivi i korišćeni.

```
<!ENTITY naziv SYSTEM "URI">
<!ENTITY naziv PUBLIC "public_ID" "URI">
```

Spoljašnji entiteti imaju posebno mesto pri označavanju neparsiranih podataka, kao što su slike, video ili audio formati. Za neparsirane podatke entitet se dodatno deklariše nazivom ispred koga stoji NDATA:

```
<!ENTITY name SYSTEM "URI" NDATA name>
<!ENTITY name PUBLIC "public_ID" "URI" NDATA name>
```

Na primer:

```
<!ENTITY ts SYSTEM “http://www.ab12.rs/ts.jpg” NDATA jpeg>
```

Entiteti se koriste i kao parametri za druge deklaracije. U slučaju parametara iza ključne reči ENTITY mora da stoji procenat, `<!ENTITY % naziv " vrednost">`. Takođe, ova vrsta entiteta može biti spoljašnja. Na primer:

```
<!ENTITY % tekst "(#PCDATA)">
<!ENTITY % info "(indeks,prezime,ime)">
…
<!ELEMENT indeks %tekst;>
```

```
<!ELEMENT prezime %text;>
<!ELEMENT ime %text;>
<!ELEMENT student1 %info;>
<!ELEMENT student2 %info;>
```

XHTML

XHTML može da opiše koristeći standarde: XML, DTD i HTML. XHTML zadovoljava tri karakteristike:

- XHTML*,* kao i HTML, zadovoljava standard ”*International Standard ISO 8879”, tj.* SGML (eng. Standard Generalized Markup Language), pri tome XHTML i HTML koriste iste unapred definisane tagove.
- XHTML document je napisan po standardima koji važe za XML document tj. zadovoljava sve kriterijume koje mora imati dobro formiran XML document.
- XHTML mora biti validan po pravilima koja su navedena u javnoj specifikaciji DTD dokumenta.

Dakle, XHTML poseduje strožija pravila pisanja u poređenju sa HTML jezikom. XHTML omogućava bolju strukturu i konzistentnost dokumenta koja omogućava da web stranice budu lako parsirane i obrađene. Takođe, stroga sintaksa omogućava i lakše održavanje, editovanje i konvertovanje u toku rada.
Pošto je XHTML zvanično standard W3C konzorcijuma, web sajt postaje kompatibilan sa svim značajnijim web čitačima i na taj način je omogućen tačniji prikaz.
Koristeći XML relativno je lako uvoditi nove elemente ili dodatne atribute. XHTML standardi, kojih ima više, projektovani su da se prilagode proširenjima koristeći XHTML module i tehnike za razvoj novih modula. Moduli omogućavaju kombinovanje postojećih i novih karakteristika.

Primer jednog XHTML dokumenta:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>  <title></title>  </head>
<body>
  <p> Ovo je školski primer dokumenta koji je napisan po
pravilima XHTML. <!—Neki komentar -->
  </p>
</body>
</html>
```

Kao što se vidi iz primera, postoje razlike u odnosu na HTML. Razlike ćemo detaljno objasniti u narednom poglavlju. Treba napomenuti da XHTML dokument može da počne sa xml instrukcijom obrade, na primer `<?xml version="1.0" encoding="UTF-8"?>` mada to nije neophodno. Aplikacija za obradu dokumenata na osnovu definicije DOCTYPE „zna“ da je XML dokument i sa DTD pravilima koji su navedeni.

Primer dela specifikacije

Ako imate konekciju, možete da skinete sa mreže i sačuvate na disku DTD specifikaciju koju koristite, sa lokacije koja je navedena u DOCTYPE. Pogledajte deo specifikacije:

```
!ELEMENT html (head, body)>
<!ATTLIST html
  %i18n;
  id          ID             #IMPLIED
  xmlns       %URI;          #FIXED
'http://www.w3.org/1999/xhtml'
>
<!ENTITY % i18n
 "lang        %LanguageCode; #IMPLIED
  xml:lang    %LanguageCode; #IMPLIED
  dir         (ltr|rtl)      #IMPLIED"
  >
<!ELEMENT img EMPTY>
<!ATTLIST img
```

```
  %attrs;
  src         %URI;          #REQUIRED
  alt         %Text;         #REQUIRED
  longdesc    %URI;          #IMPLIED
  height      %Length;       #IMPLIED
  width       %Length;       #IMPLIED
  usemap      %URI;          #IMPLIED
  ismap       (ismap)        #IMPLIED
>
```

Razlike u odnosu na HTML

- XHTML dokument mora imati deklaraciju `DOCTYPE` sa javnim identifikatorom pre korenskog elementa.
- Korenski element mora imati deklaraciju imenskog prostora kome pripadaju svi XHTML elementi. Obično se zapisuje kao podrazumevani imenski prostor, na primer: `xmlns="http://www.w3.org/1999/xhtml"` tako da svi elementi implicitno pripadaju xhtml prostoru imena.
- Dokumenti moraju biti dobro formirani.
  `<p>Normalan vs. <em>istaknut.</em></p>`
  `<p>Normalan vs. <em>istaknut.</p></em>`
- Elementi i atributi moraju biti zapisani malim slovima.
- Ako element nije prazan mora imati završni tag.
  `<p>pasus 1.</p><p>pasus 2.</p>`
  `<p>pasus 1.<p>pasus 2.`
- Vrednosti atributa moraju uvek biti zapisani pod znacima navoda.
  `<td rowspan="3">`
  `<td rowspan=3>`
- Atributi moraju imati vrednost.
  `<fieldset disabled=“disabled“>`
  `<fieldset disabled>`

- Prazni elementi moraju imati jedan tag koji se završava sa /.
  `<br/><hr/>`
  `<br><hr>` 🚫
- Beli prostor se mapira u jedan razmak ako je unutrašnji ili bez praznina na početku ili kraju.
- Umesto atributa **`name`** koji se koristi u HTML-u, koristi se atribut **`id`**. Ovaj atribut je u DTD definisan kao ID, tj jedinstvene je vrednosti.
  Dobar HTML:
  `<input type="text" name="s" value="name">`.
  Dobar XHTML
  `<input type="text" id="s" value="name" />`.
- Atribut **`language`** u element **`script`** je zastareo.
  `<script language="JavaScript"` type="text/JavaScript">

Provera ispravnosti XHTML dokumenta

XHTML dokumenti su potpuno standardizovani po XML pravilima i validni po DTD specifikaciji koja se navede. Zahvaljujući tome svaki dokument može biti proveren sa aspekta validnosti i pre nego što se neki web čitač pokuša da interpretira njegov sadržaj.
Provera ispravnosti se obavlja posebnim aplikacijama koje se nazivaju **validatori**. Validatori postoje kao:

- web sajtovi koji nude servis validacije DTD dokumenata,
- samostalne aplikacije ili su
- delovi nekih razvojnih okruženja.

Primer jednog od validatora dat je na adresi http://validator.w3.org/.

SPECIFIKACIJE XHTML-a

Postoji tri različite specifikacije za XHTML 1.0 dokumenata. To su:

- striktna specifikacija (eng. strict),
- tranziciona specifikacija (eng. transitional),
- frejmovska specifikacija (eng. frameset).

Najčešće korišćena i najsličnija specifikaciji koju smo koristili je tranziciona specifikacija.
Razlike su u dostupnosti nekih HTML elemenata odnosno atributa u različitim specifikacijama. Zato pri pisanju XHTML dokumenta treba pažljivo birati elemente u zavisnosti od primenjene specifikacije.

- Striktna specifikacija

  Deklaracija je:

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

  Ovaj DTD sadrži sve HTML elemente i atribute osim zastarelih elemenata (na primer `font`). Takođe nije dozvoljen element `frameset`.

- Tranziciona specifikacija

  Deklaracija je:

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

  Ovaj DTD sadrži sve HTML elemente i atribute, uključujući zastarele elemente. Element `frameset` nije dozvoljen.

- Frejmovska specifikacija

  Deklaracija je:

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

  Ovaj DTD je isti kao tranzicioni XHTML, ali je dozvoljeno korišćenje elementa `fremeset`.

Pitanja za proveru znanja

1. Šta znači skraćenica XML?
2. Šta znači dobro formiran dokument?
3. Navesti jedan način validacije XML dokumenata.
4. Šta znači skraćenica DTD?
5. Navesti jedan primer DTD dokumenta i objasniti napisana pravila validacije.
6. Koja je razlika između HTML i XHTML koda?
7. Prepravite sledeći HTML kod tako da odgovara standardu XHTML:
   a. `<img src=logo.png>`
   b. `<h1>` … `</H1>`
   c. `<input type="radio" checked>`
   d. `<hr>`
   e.
```
<ul>
 <li>stavka 1
 <li>stavka 2
</ul>
```
8. Koje specifikacije XHTML-a postoje?

Rezime

Karakteristike XHTML-a:

1. XHTML, kao i HTML zadovoljava standard SGML, pri tome koriste iste unapred definisane tagove,
2. XHTML document je dobro formiran XML document,
3. XHTML mora biti validan po pravilima koja su navedena u specifikaciji DTD dokumenta.

Specifičnosti XHTML-a:

- XHTML ima deklaraciju `DOCTYPE` sa javnim identifikatorom pre korenskog elementa,
- korenski element mora imati deklaraciju imenskog prostora kome pripadaju svi XHTML elementi,
- dokumenti moraju biti dobro formirani,
- elementi i atributi moraju biti zapisani malim slovima,
- ako element nije prazan mora imati završti tag,
- vrednosti atributa moraju uvek biti zapisani pod znacima navoda,
- atributi moraju imati vrednost,
- prazni elementi moraju imati jedan tag koji se završava sa /,
- beli prostor se mapira u jedan razmak ako je unutrašnji ili bez praznina na početku ili kraju,
- umesto atributa `name` koji se koristi u HTML koristi se atribut `id`.

Deo 3: CSS

- Struktura zapisa
- Povezivanje stilova i dokumenata
- Kaskadni koncept stilova
- Box model
- Ograničavanje veličine elementa
- Formiranje selektora
- Formatiranje teksta
- Zadavanje boja
- Plutajući elementi

U ovom poglavlju se detaljno opisuje dizajn web stranica zasnovan na primeni kaskadnih stilova. Najpre se uvodi struktura CSS zapisa, načini povezivanja sa HTML dokumentom, zatim se definišu načini formiranja selektora i opisuje primena stilova. Na kraju poglavlja se uvode plutajući elementi.

HTML elementi definišu strukturu stranice i njen sadržaj, ali sadrže i brojne atribute koji određuju detalje u vezi izgleda. Takav koncept u razvoju web sajtova nosi puno problema u projektovanju i kasnijem održavanju obimnih i modernih sajtova, pa je vremenom bilo neophodno odvojiti stilove za prikaz od strukture HTML dokumenta i omogućiti jednoznačnu primenu stilova na grupe elemenata. Ovo se postiglo uvođenjem kaskadnih stilova – CSS-a. CSS je skraćenica od engleskog naziva ***C****ascading* ***S****tyle* ***S****heets.*
CSS omogućava da se isti stil primeni na više HTML elemenata ili grupa elemenata. Tako se na dosledan način definiše izgled i funkcionalnost stranica.

Struktura zapisa

Struktura zapisa CSS-a sastoji se od dva osnovna dela. To su:

- selektor koji definiše elemente na koje će pravilo primeniti i
- deklaracije koje se odnose na definisanje izgleda selektovanih elemenata.

Deklaracija opisuje prikaz tako što se u deklaraciji prvo navede naziv svojstva (npr. color), a zatim vrednost svojstva (npr. green).
Jedna ili više deklaracija smeštaju se između vitičastih zagrada. Međusobno su odvojene tačka-zapetom (;). Tačka-zapeta nije neophodna ako iza deklaracije sledi zatvorena vitičasta zagrada tj. kada nema funkciju razdvajanja deklaracija. Ako se izostavi tačka-zapeta tamo gde ona razdvaja deklaracije, deklaracija se ignoriše.

```
selector { svojstvo1: vrednost1 }
```

ili

```
selector {
           svojstvo1: vrednost1;
           svojstvo2: vrednost2;
           svojstvo3: vrednost3;
         }
```

Na primer CSS kod za definisanje stilova koji se primenjuju na naslove `h1,` odnosno pasuse sa pratećim deklaracijama je:

```
h1 { color: green; }
p  { font-size: small; font-family: sans-serif; }
```

gde su `h1` odnosno `p` elementi koji su korišćeni kao CSS selektori. Ovi selektori se nazivaju selektorima tipa elementa i jedan su od tipova koji postoje. Deklaracije koja su definisane za svako pravilo primenjuju se na sve `h1`, odnosno `p` elemente dokumenta.
Postoje i drugi selektori, a detaljnije o svim selektorima biće u nastavku ovog poglavlja.

Praznine i komentari u kodu

Praznine i novi redovi nisu od značaja u CSS kodu. Zato se deklaracije često pišu u zasebnim redovima i sa belinama. Na primer:

`Upotreba novog reda i belina u CSS kodu:`

```
p{
  font-size: small;
  font-family: sans-serif;
}
```

Komentari se pišu između znakova `/*` i `*/`, bez obzira da li su u jednom ili više redova (dakle ne koriste se dve kose crte - //). Pišu se da bi se omogućilo lakše razumevanje i kasnije lakše održavanje koda.

`Upotreba komentara:`

```
/*
Važno:
Ne preterujte sa komentarima!
Kod treba da bude razumljiv bez komentara!!!
*/
p{
  margin: 1em; /* margina */
  padding: 2em;  /* color: white; */
  background-color: blue;
}
```

Povezivanje stilova i dokumenata

Postoje tri načina za povezivanja stilova sa HTML dokumentom:
(i) unutrašnji, (ii) spoljašnji i (iii) linijski.

Unutrašnji

– Ovaj način povezivanja se često koristi u udžbenicima i pri testiranju, ali ne i u praksi. Unutrašnji stil je napisan neposredno u HTML dokumentu na koji se odnosi i deluje samo na taj dokument. Stil se dodaje u zaglavlje HTML dokumenta, između tagova `<head>` i `</head>` pomoću elementa **`style`**.
Na primer, da bismo promenili boju teksta na stranici u plavo, dodaje se sledeći CSS kod u zaglavlje:

```
<head>
  <title>Required document title here</title>
  <style type="text/css">
    body{color: blue;}
  </style>
</head>
```

Spoljašnji – Ovakvi stilovi su napisani u zasebnom fajlu, sa ekstenzijom .css. Fajl sa stilovima se povezuje ili importuje u jedan ili više HTML dokumenata. Na ovaj način ceo web sajt deli iste stilove. Ovo je najpraktičniji način upotrebe stilova. Spoljašnje povezivanje se postiže se primenom elemenata:

1. **`<link>`**
   Ovo je u praksi najčešći način povezivanje spoljnih stilova i HTML dokumenta. Element `link` se navodi u elementu `head`, može se pojaviti više puta, nema sadržaj, a atributi su:

   - `Rel = "stylesheet"` - definiše relaciju sa tekućim dokumentom. Kada je to CSS fajl onda je vrednost uvek `stylesheet`,
   - `Href = "url"` - definiše lokaciju .css fajla,
   - `Type = "text/css"` - definiše MIME tip linkovanog dokumenta (više o MIME tipovima pogledati na: http://www.iana.org/).

   `Spoljašnje povezivanje:`

```
<head>
  <link rel="stylesheet" type="text/css"
        href="glavni.css">
</head>
```

2. **`<@import>`**
   Ovaj element se može primeniti i u `.css` fajlu kao i u HTML dokumentu. Omogućava importovanje sadržaja fajla.
   `Importovanje stilova:`

```
 <head>
  <style>
    @import url("/path/stylesheet.css");
    p { font-face: Tahoma;}
  </style>
  <title>Naslov dokumenta</title>
</head>
```

Komanda `@import` omogućava čitanje pravila pre nego što budu primenjena. Zbog toga pravila koja su naknadno definisana preklapaju ova pravila. Mala je razlika u upotrebi između `@import` i `link` načina uključivanja spoljašnjih CSS fajlova. `Import` dozvoljava da se spoljašnji CSS fajl uključi u definiciju stila, na primer:

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

Stariji čitači ne prepoznaju `import` što takođe može biti iskorišćeno. Na primer:
`@import url(../style.css);`
neće biti primenjeno u slučaju IE3,4 ili Netscape 4....

Linijski (eng. inline) – Predstavlja način primene stilova kada se oni pišu u početnom tagu elementa. Tada se stil primenjuje samo na taj element, koristeći atribute tog elementa. Na primer:
`<h1 style="color: red"></h1>`.
Ako se koristi više deklaracija, one se razdvajaju tačka-zapetom, kao u prethodnim načinima primene. Na primer:
`<h1 style="color: red; margin-top: 2em">Uvod</h1>`.
Napomena: Linijske stilove treba izbegavati, osim ako je to apsolutno neophodno. Obično se koriste da bi se promenilo delovanje spoljašnjih stilova.

Kaskadni koncept stilova

Nasleđivanje

Web čitač prikazuje sadržaj HTML dokumenta na osnovu korišćenih HTML elemenata, njihove strukture i sadržaja. Elementi u dokumentu su u određenoj hijerarhijskoj strukturi, a celokupan sadržaj je u elementima tj. struktura elemenata definiše strukturu sadržaja. Stilovi u dokumentu takođe podržavaju hijerarhijsku strukturu, odnosno koriste osobinu nasleđivanja. Koristeći nasleđivanje elementi dobijaju svojstva prikaza od nadelemenata, odnosno prenose ih na podelemente.
U narednom primeru je dat HTML dokument gde se prikazuju dva naslova, h3 i h4, sa jednom naglašenom reči.

```
<h3>Podrazumevani naslov h3 sa <em> naglašenom </em>
reči.</h3>
<h4>Podrazumevani naslov h4 sa <em> naglašenom </em>
reči.</h4>
```

Za navedeni HTML kod prikaz je sledeći:

Podrazumevani naslov h3 sa *naglašenom* reči.

Podrazumevani naslov h4 sa *naglašenom* reči.

Slika 3.1. Podrazumevani prikaz naslova sa naglašenom reči

Ako se primeni stil za promenu fonta u naslovu, putem nasleđivanja tj. implicitno dolazi do promene fonta i u naglašenoj reči, iako to nije navedeno. Na primer:

CSS kod za promenu fonta u naslovima tipa h3,h4:

```
h3,h4 {
  font-size: small;
  font-family: sans-serif;
}
```

Rezultat primene za isti HTML kod je sledeći prikaz:

Podrazumevani naslov h3 sa *naglašenom* reči.

Podrazumevani naslov h4 sa *naglašenom* reči.

Slika 3.2. Nasleđivanje stila u naglašenom tekstu

Dakle, primenom stila na naslove h3, h4 definiše se novi font za ove elemente, ali još važnije, definisani font nasleđuje podelement u elementima h3, odnosno h4, a to je u našem slučaju element em. Tako se novi font javlja i na naglašenoj reči.
Isto pravilo važi za sve elemente jednog dokumenta. Na primer, definisani stil za font elementa body bio bi primenjen na sve elemente dokumenta tj. podelemente body elementa, pa samim tim i na sve naslove u dokumentu. Očigledno je da se ovakvim pravilima može definisati primena više različitih stilova nad istim elementom. Ovo nazivamo konfliktom stilova, a način rešavanja konflikata je jedna od najvažnijih osobina CSS-a.

Kaskadno rešavanje konflikata

Pogledati primer koji sledi.
CSS:

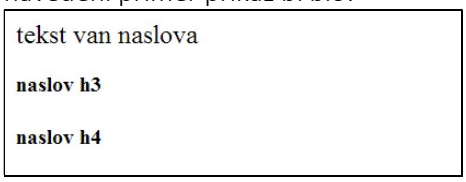
```
body {
  font-size: medium;
}
h3,h4 {
  font-size: small;
}
```

HTML:

```
<body>
  tekst van naslova
  <h3> naslov h3 </h3>
  <h4> naslov h4 </h4>
</body>
```

Prvi stil iz primera definiše font u body elementu, ali se taj stil primenjuje i na sve podelemente elementa body tj. na sve elemente HTML stranice. Drugi stil koji definiše font za neki specifičan element stranice, u primeru je to stil za elemente h3 i h4 istog dokumenta, u konfliktu je sa već definisanim stilom.

Razrešavanje konflikata toliko je važno da je način rešavanja ugrađen u sam naziv „kaskadnih" stilova. Pravilo se primenjuje „**odozgo na dole**“, sve dok ne bude preklopljeno stilom koji ima veću težinu. Za navedeni primer prikaz bi bio:



Slika 3.3. Slučaj preklapanja dva stila

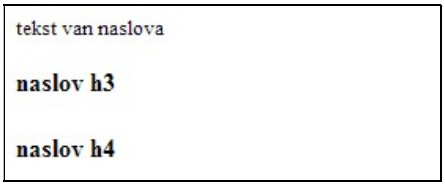
U skladu sa navedenim pravilom važi da linijski stil ima veću težinu od unutrašnjeg, dok unutrašnji stil ima veću težinu od spoljašnjeg stila. Ako se na neki element ne primenjuje nijedan eksplicitno naveden stil, važeći stil je interni stil čitača, koji nazivamo **podrazumevani stil** za renderovanje tj. formiranje prikaza. Posebno, korisnici mogu imati sopstvene stilove (eng. user style sheet) koji preklapaju podrazumevane stilove čitača.
Stilovi koji se definišu u web stranici, dakle, stilovi koje definiše autor web stranice i važe za istu, preklapaju i korisničke i podrazumevane stilove. Jedini izuzetak su stilovi koje korisnik identifikuje kao „important“.
Zaključak: CSS stilovi su hijerarhijski organizovani, a kao koncept za određivanje veće težine stila, koristi se sledeće pravilo - **stil koji je definisan bliže elementu ima veću težinu,** ili **stil koji je kasnije definisan ima veću težinu**.
Koji bi stil mogao da preklopi stil definisan za h3 i h4?
Jedini stil, od ranije navedenih, koji bi bio bliži elementima h3, h4 od onog koji je eksplicitno definisan za te elemente u zaglavlju dokumenta, je stil koji se definiše u samom elementu, tzv linijski. Kod i odgovarajući prikaz dati su u primeru koji sledi.

CSS:

```
body {
  font-size: small;
 }
h3,h4 {
  font-size: small;
 }
```

HTML:

```
<body>
  tekst van naslova
  <h3 style="font-size: medium"> naslov h3 </h3>
  <h4 style="font-size: medium"> naslov h4 </h4>
</body>
```



Slika 3.4. Preklapanje stilova primenom linijskih stilova

Kao što se vidi iz primera, linijski stil koji je definisan u elementima `h3` i `h4` preklopio je stilove koji su definisani za te elemente u zaglavlju stranice jer je hijerarhijski bliži i po pravilima primene kaskadnih stilova ima veću težinu u primeni.

!important

Stil može da nosi dodatnu oznaku `!important` kojom se postiže najveći prioritet u primeni u odnosu na sve ostale stilove. Primenjuje se samo kada je to neophodno.
Na primer, ako tekst u paragrafu treba da bude uvek plav, može se napisati pravilo:
`p { color: blue !important; }`.

U slučaju da se naknadno doda drugi stil koji takođe definiše boju teksta u paragrafu, pa čak to bio i linijski stil, tekst u paragrafu će ostati plav. Dakle, stil sa indikatorom `!important` ne može se preklopiti sa drugim stilovima.
Ipak, da li je moguće promeniti ovaj stil?
Jedini način da se neki `!important` stil preklopi jeste da drugi stil takođe koristi indikator `!important`, ali da bude sa većim prioritetom.

Redosled primene stilova

Podatak o stilu može da potiče od različitih izvora. U narednoj listi, navedeni su izvori stilova od opštijih ka specifičnijim, baš kako ide i prioritet. Stavke u listi na nižoj poziciji preklapaju stavke iznad.

1. Podrazumevani stilovi browser-a.
2. Korisnički definisani stilovi u browser-u.
3. Spoljni stilovi.
4. Importovani stilovi (dodati pomoću `@import` funkcije).
5. Ugrađeni stilovi (dodati pomoću elementa `style`).
6. Linijski (eng. inline) stilovi (dodaju se u početnom tagu).
7. Stilovi označeni sa `!important`.
8. Korisnički definisani stilovi sa `!important`.

Šta se događa ako imamo konflikt između stilova jednake težine? Na primer:

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

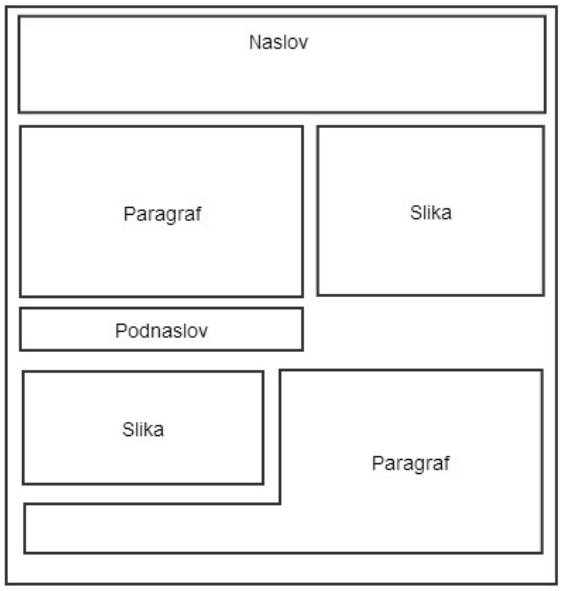
ili

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

U ovom scenariju, važeći stil će biti onaj koji je poslednji definisan!

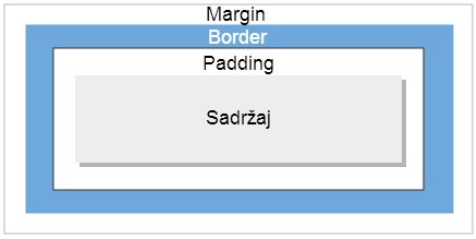
Box model

Svi HTML elementi, kao što su paragraf, slika, naslov itd. oivičeni su pravougaonikom kojim se odvaja njihov sadržaj od ostatka stranice, kao na slici 3.5.



Slika 3.5. Box model na jednoj stranici

Svaki pravougaonik se sastoji od nekoliko svojstava koja zajedno definišu oblast jednog elementa. Ta svojstva su: `padding`, `border`, `margin`. Na slici 3.6 je prikazan model sa ovim svojstvima za jedan element. Ovaj model nazivamo box modelom.

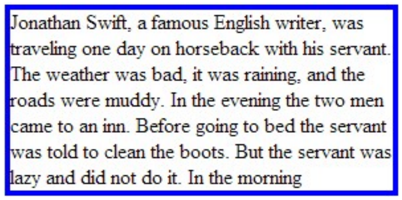


Slika 3.6. Box model

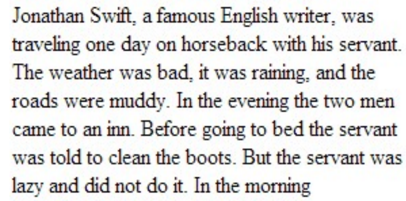
Pogledajmo značenje pojedinih svojstava iz modela.

- Border

  Predstavlja ivicu oko sadržaja elementa. Može imati definisanu debljinu, boju i stil. Podrazumevano ivica ne postoji.



Sa definisanim svojstvom `border`

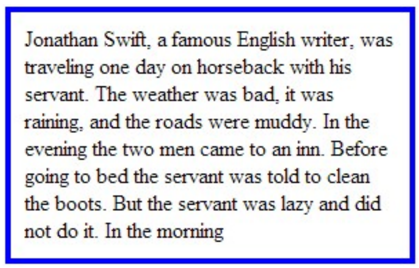


Bez svojstva `border`

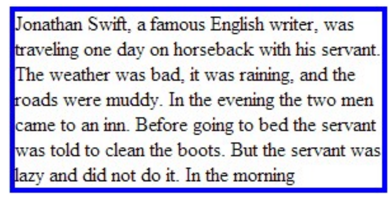
Slika 3.7. Svojstvo border

- Padding

  Predstavlja prostor između sadržaja elementa i njegove ivice. Za prethodno definisanu ivicu, na slici 3.8 prikazano je ovo svojstvo.



Sa definisanim svojstvom `padding`

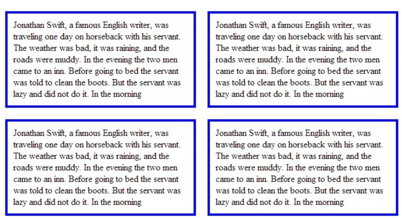


Bez svojstva `padding`

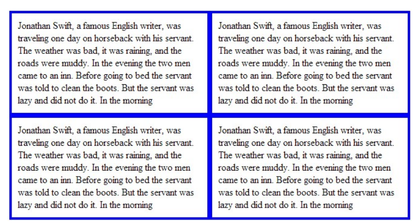
Slika 3.8. Svojstvo padding

- Margin

  Predstavlja prostor između ivica susednih elemenata.



Sa definisanim svojstvom `margin`



Bez svojstva `margin`

Slika 3.9. Svojstvo `margin`

Pri radu sa ovim modelom, osim navedenih CSS svojstava, koriste se pomoćni pojmovi kao što su:

- Unutrašnja ivica (eng. inner edge). Ivica koja odvaja oblast sadržaja od ostatka predstavlja unutrašnju ivicu elementa.

Mada se ova ivica prikazuje u box modelu, ona nije vidljiva u prikazu.

- Spoljna ivica (eng. outer edge). Spoljna ivica je ujedno i spoljna ivica margine i predstavlja ivicu celog elementa. Ona odvaja ukupnu površ koju zauzima element na stranici uključujući sadržaj, peding, border i margine. Na slici je označena bledosivom linijom, mada u stvarnosti ova linija nije vidljiva.
- Napomena: Ukoliko se vrednost ovog svojstva navodi u procentima onda se odnosi na procenat širine (čak i kada se odnosi na gornji i donji prostor) roditeljskog elementa.

Dimenzije blok elemenata

Širina i visina blok elementa računa se automatski od strane čitača. Podrazumevana širina je širina klijentske oblasti prozora čitača ili drugog elementa koji ga sadrži, a visina je ona koja je potrebna da se podesi sadržaj za prikaz.
Međutim, dimenzije mogu da se definišu unapred, što omogućava da se unapred definiše izgled stranice. Dimenzije se zadaju svojstvima `width`, odnosno `height`, a dodela vrednosti se vrši na način kako se dimenzioniraju objekti. Ove veličine se ne nasleđuju.
Box model pokazuje da element ne zauzima prostor samo svojim sadržajem već i dodatnim svojstvima koja su prethodno opisana. Pogledati primer koji sledi. Razmisliti kolika je ukupna širina elementa oivičena označenim `div`-om.

```
<html>
<head>
<style>
  div {
    width: 200px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0px;
  }
```

Ovo je neki tekst u div elementu prikazan sa prametrima:
div {
width: 200px;
padding: 10px;
border: 5px solid gray;
margin: 0px;
}

200px + 2x10px + 2x5px + 2x0px

```
</style>
</head>
<body>
  <div class="ex">Ovo je neki
tekst...
  </div>
</body>
</html>
```

Slika 3.10. Ukupna širina elementa

Ograničavanje veličine elementa

Ograničavanje se može izvesti po obe dimenzije elementa tj. po širini odnosno po visini. Ograničavanje može biti u pogledu minimalne ili maksimalne vrednosti određene dimenzije. Zbog toga postoje i po dva svojstva za svaku dimenziju koja to regulišu:

- `max-width` - postavlja se maksimalna širina,
- `max-height` - postavlja se maksimalna visina,
- `min-width` - postavlja se minimalna širina,
- `min-height` - postavlja se minimalna visina elementa.

Vrednosti koje se pridružuju ovim atributima su iskazani u apsolutnim ili relativnim jedinicama. Ako se koristi procenat onda se on odnosi na širinu, odnosno visinu elementa koji sadrži tekući element.
U narednim poglavljima ovo svojstvo će biti iskorišćeno za kreiranje ograničena dimenzija elemenata i za izbegavanje izvesnih nedostataka određenih tipova dizajna.

Selektori

Selektor stila definiše elemente na koje se deklaracije primenjuju. Pravila pisanja selektora se grupišu na nekoliko tipova:

- selektor po tipu elementa: `h2 { color:blue; }`,
- po ID elementa, tj odnosi se na tačno određeni element: `h2#glavni { color:red; }`,
- po klasi elemenata, odnosi se na elemente određene klase: `em.myClass { color:green; }`,
- koristeći pseudo klase elementa: `a:visited { color:red; }`,
- pomoću konteksnih selektora: `h1 em { color: blue }`,
- koristeći atribute sa ili bez vrednosti elemenata: `h2[name] { color:red; }`.

U nastavku ovog poglavlja pozabavićemo se detaljno navedenim tipovima selektora.

Selektori za tip elementa

Ovo je najjednostavnija vrsta selektora. Korišćen je u prvim primerima ovog poglavlja. Za selektor se koristi tip HTML elementa na koji se selektor odnosi, na primer:

```
h2 { color:blue; }
```

deklaracija `color:blue;` se primenjuje na sve naslove `h2`.
Ukoliko se isti skup deklaracija navodi za više istih tipova elemenata, onda se ti elementi navode sa zapetom između. Na primer, deklaracija `color:blue;` u okviru stila

```
h2,h3,h4 { color:blue; }
```

se odnosi na sve naslove tipa `h2`, `h3` i `h4`.

Selektori za samo jedan element

Upotreba atributa `id` u elementu označava taj element na jedinstven način. Pošto je vrednost atributa `id` jedinstvena, ona se koristi za primenu stila na tačno određeni element.
U selektoru se piše znak `#` iza koga sledi vrednost `id` atributa elementa na koji se stil odnosi. Na primer:

`#glavniNaslov { color: red; }`.

Ovaj selektor se odnosi na element čiji je `id="glavniNaslov"`. Selektori mogu koristiti kombinaciju više vrsta označavanja, na primer:
`div#zakljucak p{ color:blue; }`
označava primenu na paragraf odeljka čiji je `id="zakljucak"`.

Selektori za grupu

Svrstavanje više elemenata u grupe vrši se koristeći atribut `class = "grupaNaslova"`. Selektor koji treba da označi primenu stila na grupi (klasi) elemenata piše se sa tačkom ispred naziva klase, na primer:
`.grupaNaslova { color:blue;}`.
Tačka ispred naziva klase je obavezna. Ime klase se može koristiti zajedno sa tipom elementa, a u tom slučaju pravilo se primenjuje na elemente tog tipa iz označene klase. Na primer:
`h2.grupaNaslova{ color:red;}` .

Pomoću konteksnih selektora

Elementi u HTML stranici su hijerarhijski uređeni, organizovani na način da poštuju strukturu teksta u dokumentu. Ovu strukturu poštuje i CSS. Pomoću kombinatora vrši se selekcija elemenata na osnovu hijerarhijske relacije između njih.

- Označavanje potomaka. Pogledajmo primer. Ako crvenom bojom treba naglasiti tekst u `em` podelementu naslova `h1`, tj. `<h1>Ovaj deo je od<em> velike</em> važnosti</h1>`, da li bi primena stila `em { color: red }` bila dobro rešenje? Ne, jer će se stil primeniti, tj. promena boje će biti, na svim `em` elementima dokumenta, a ne samo na `em` podelementima u `h1`.

  Dakle, potrebno je pravilo napisati drugačije, uzimajući u obzir hijerarhiju elemenata. Tehničkim rečnikom, a u kontekstu relacija HTML elemenata u hijerarhijskoj strukturu, treba reći da je ciljani `em` element potomak elementa `h1`. Naravno, ne obavezno prvi

potomak, jer to nije naglašeno. Pravilo koje definiše hijerarhijski odnos piše se navodeći samo razmak između roditeljskog elementa i potomka, dakle ispravno bi bilo:
`h1 em { color: blue }`,
a pravilo bi se odnosilo i na slučaj:

```
<h1>Ovo je<span class="myClass">deo od
<em>velike</em> važnosti</span></h1>.
```

- **Označavanje podelementa.** Podelement (dete element) nekog elementa se označava tako što se iza roditeljskog elementa navede znak **>** pa podelement. Na ovaj način se ne selektuju svi elementi potomci, već samo prvi nivo potomaka - deca. Na primer,

```
div > p {
  color:blue;
}
```

selektor bi označio samo **p** elemente koji su deca od elementa **div**, pa bi došlo do promene boje samo u pasusima koji su na prvom podnivou od **div**.

- **Označavanje određenog elementa blizanca (na istom hijerarhijskom nivou).** Elementi istog nivoa nazivamo blizancima i oni imaju isti roditeljski element. Kada se označavaju naredni elementi „blizanci“ koristi se oznaka **+** . Na primer, selektorom

```
div + p{ color:blue; }
```

svi pasusi na istom nivou kao i **div** element, a koji slede iza elementa **div** se označavaju.

- **Označavanje svih elementa blizanca (na istom hijerarhijskom nivou).** Kada se označavaju svi elementi istog nivoa, koristi se znak **~**. Na primer

```
div ~ p{ color:blue; }
```

označiće sve pasuse na istom nivou kao i **div**.

Pseudo klase i elementi

Pseudo klase i pseudo elementi su deo CSS pravila kojima se postižu specijalni efekti selektora. Radi se o selektorima kojima se dodatno

označavaju karakteri, delovi elementa ili stanje elementa koji nisu opipljivi tj. koji ne mogu da se „ručno“ selektuju. Označavaju se dvotačkom (`:`) odnosno sa dve dvotačke (`::`) i navode se posle naziva elementa.
Sintaksa je:

```
selektor:pseudo {
  svojstvo1:vrednost1;
}.
```

Na primer `li:first-child` je primer primene pseudo klase koja detaljnije određuje selektor `li` značenjem da se selekcija odnosi na prvo dete element tipa `li`.
Pseudo klase se koriste zajedno sa svim selektorima, pa tako i sa CSS klasama. Ponašaju se kao sve ostale klase. Na primer, tu još spadaju klase za određivanje stanja linka:

```
a.vets:visited {
  color: red;
}
<a class="vets" href="http://">VISER</a>.
```

Pseudo elementi se koriste za označavanje dela elementa ili za ubacivanje sadržaja. Samo jedan pseudo element se može primeniti po selektoru, što je razlika u odnosu na pseudo klase.
Lista nekih češće korišćenih pseudo klasa i pseudo elemenata sa opisom je data u Tabeli 3.1, a ispod tabele su navedeni primeri primene istih.

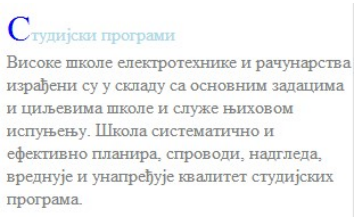
Tabela 3.1. Lista pseudo klasa i elemenata

Selektor	Opis
:first-child	Prvo dete element
::first-line	Prva linija u prikazu
::first-letter	Prvo slovo
::before	Ubacivanje sadržaja i stila ispred selektora
::after	Ubacivanje sadržaja posle selektora

:focus	Selektuje se stanje fokusa kontrole za unos
:link :visited :active :hover	Selektuje stanje veza, dakle ne element već stanje!

Označavanje prve linije, odnosno slova:

```
body{
  color:gray;
}
p::first-line {
  color:lightblue;
}
p::first-letter {
  color: blue;
  font-size: 2em;
  font-weight: normal;
}
```



Slika 3.11. Označavanje prve linije, odnosno slova

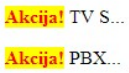
Označavanje fokusirane kontrole:

```
input:focus {
  background-color: yellow;
}
```

Slika 3.12. Isticanje fokusirane kontrole

Umetanje sadržaja:

```
p.akcija::before{
  content:"Akcija!";
  font-weight:bold;
  background-color:yellow;
  color:red;
}
```



Slika 3.13. Ubacivanje sadržaja

Selektori na osnovu atributa

- Selekcija elemenata se može izvršiti na **osnovu navedenog atributa** u elementu tako što se u selektoru doda naziv atributa oivičen uglastim zagradama, `[atribut]`. Pogledajte primer:

  `Primer selektora na osnovu atributa:`

  ```
  a[href] {
    text-decoration: none;
  }
  ```

  Selektuju se svi linkovi tj. elementi `a` koji imaju definisan atribut `href`, a kao rezultat primene stila iz primera, takvi linkovi neće biti podvučeni.
- Selekcija se može izvršiti **na osnovu vrednosti atributa**. Tada se u uglaste zagrade navodi naziv atributa, znak jednakosti i vrednost atributa: `[atribut=val]`. Na primer:

  ```
  a[href="http://www.viser.edu.rs"] {
    text-decoration: none;
  }
  ```

  Rezultat je da, gde god se pojavio `href` atribut u nekom elementu, a ukazuje na navedenu adresu, link neće biti podvučen.
- Izbor elemenata može da se izvrši **na osnovu posebnog uslova** koju zadovoljava vrednost atributa. Operatori odgovaraju uslovu koji se koristi, a pomenućemo sledeće: `~=`, `|=`, `^=`, `$=`, `*=`.
  U listi su navedeni primeri i objašnjenja.
  - `[atr ~= vrednost]` - Selekcija svih elemenata koji u vrednosti atributa sadrže navedenu vrednost kao jednu reč odvojenu razmaknicom.
    Npr: `[title ~= "upis"]`, odgovaraju: "upis", "junski upis", ...

- [atr |= vrednost] - Selekcija svih elemenata koji u vrednosti atributa sadrže samo navedenu vrednost ili počinju tom reči i crticom iza.
  Npr: [class |= "main"], odgovaraju: "main", "main-1", ...
- [atr ^= vrednost] - Selekcija svih elemenata čija vrednost atributa počinje sa navedenom vrednosti.
  Npr: h2[class ^= "nsl3"], odgovaraju: "nsl3", " nsl33", ...
- [atr $= vrednost] - Selekcija svih elemenata čija se vrednost atributa završava sa navedenom vrednosti.
  Npr: img[src $= ".png"], odgovaraju: "slika1.png", ...
- [atr *= vrednost] – Selekcija svih elemenata čija vrednost atributa sadrži navedeni podstring.
  Npr: a[href *= "viser"], odgovaraju: "www.viser.edu.rs", ...

Vrednosti više atributa

Kada se u selektoru navodi uslov za vrednosti više atributa jednog elementa ili više provera jednog atributa, tada se selektori na osnovu atributa navode jedan za drugim. Na primer:

```
p[class="main"][naziv="prvi"],p[class="main"][naziv="drugi"] { color: red; }.
```

Grupe selektora

Grupe selektora se pišu tako što se pre narednog selektora stavi zapeta. Na primer:

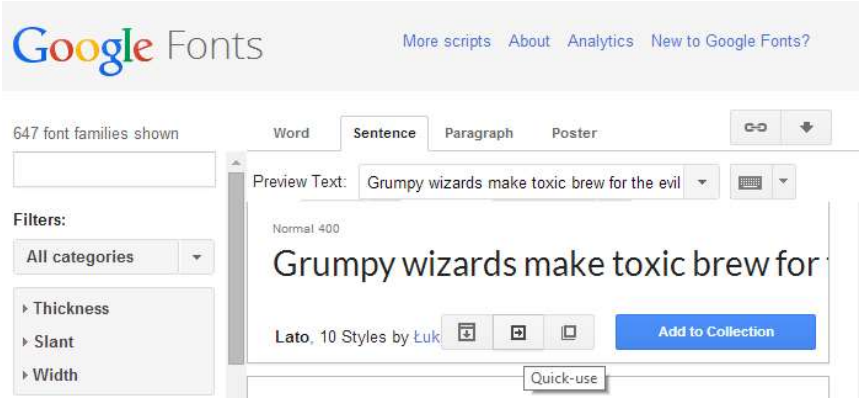
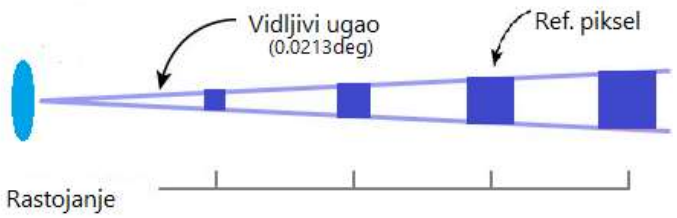
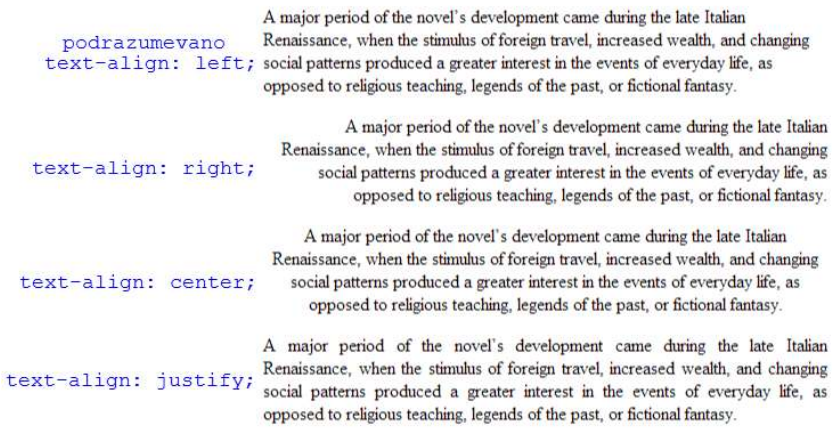
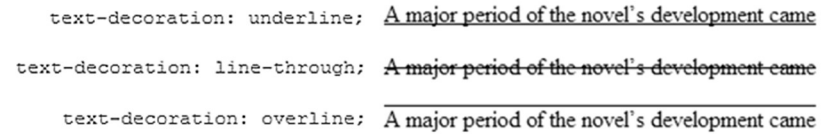
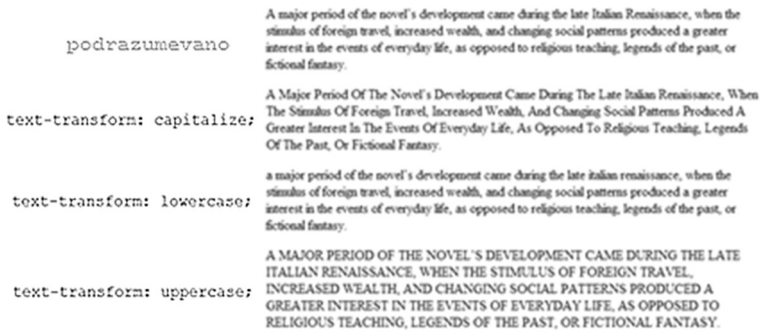
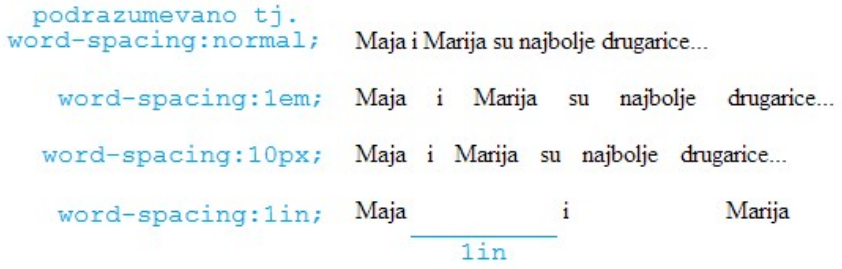
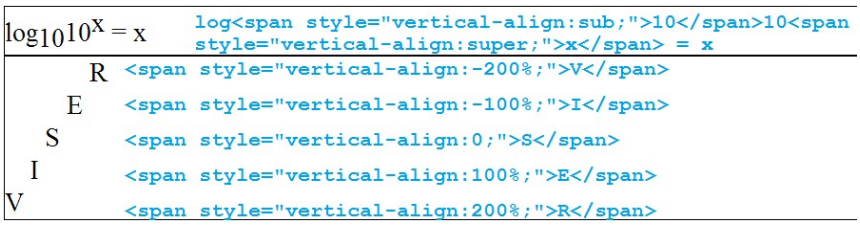
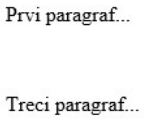
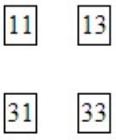
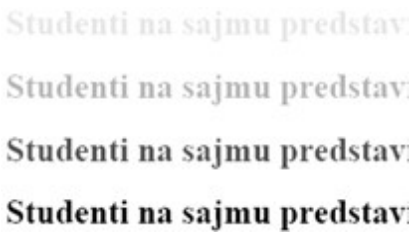
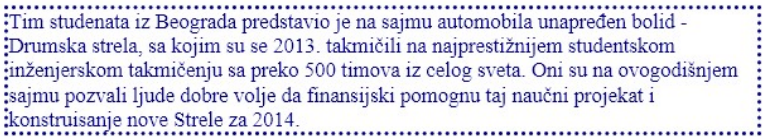
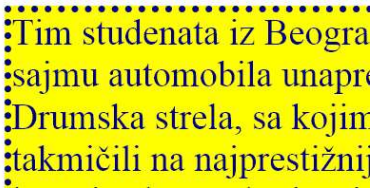
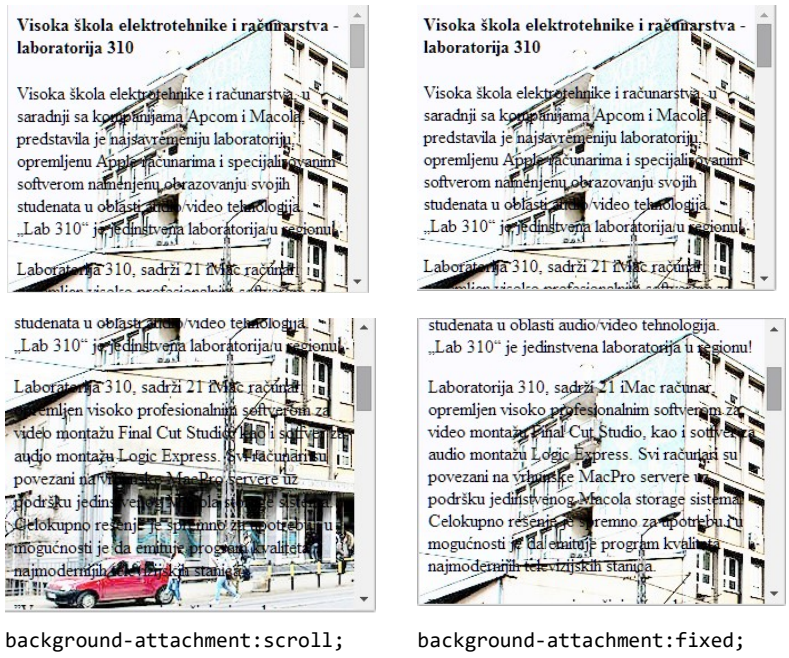
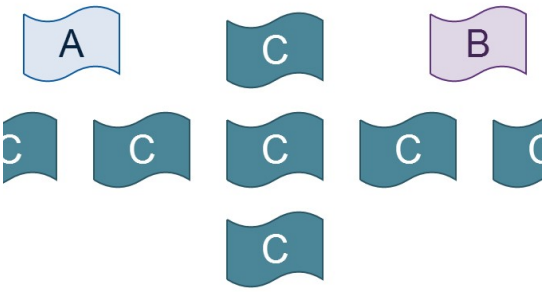
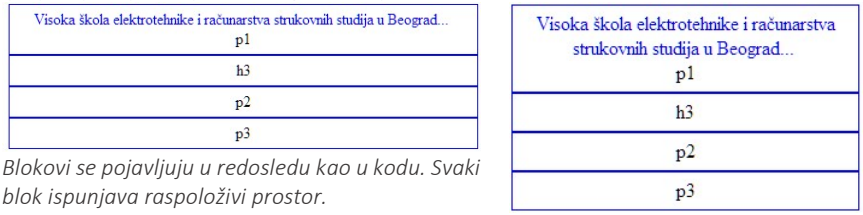
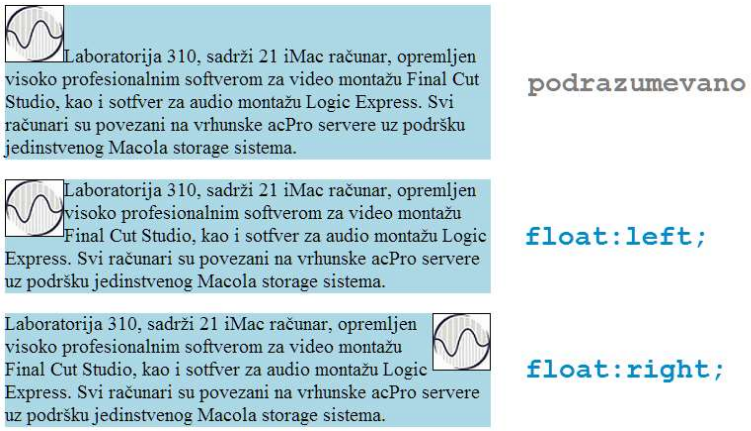
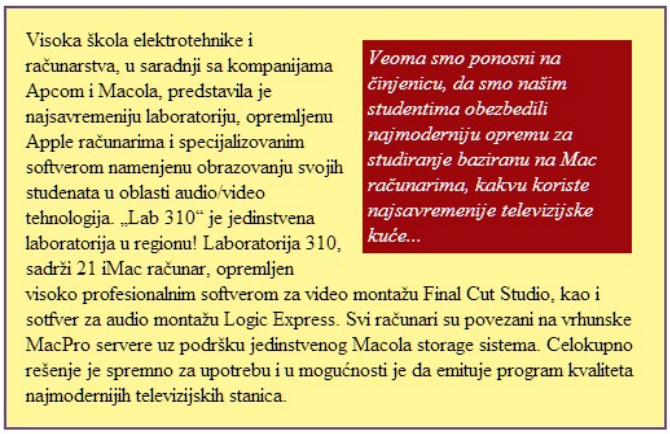
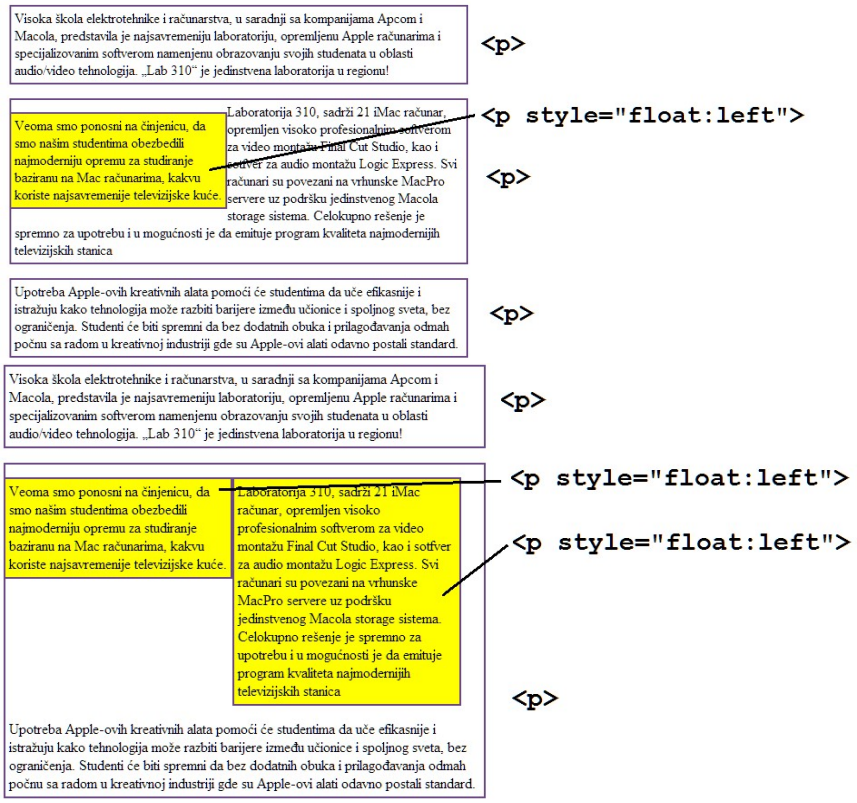
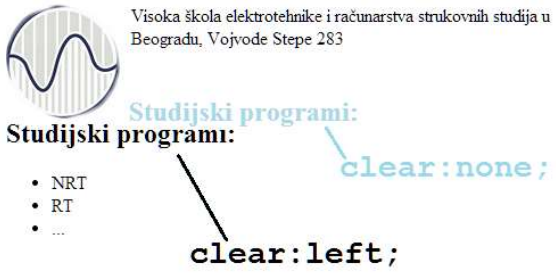
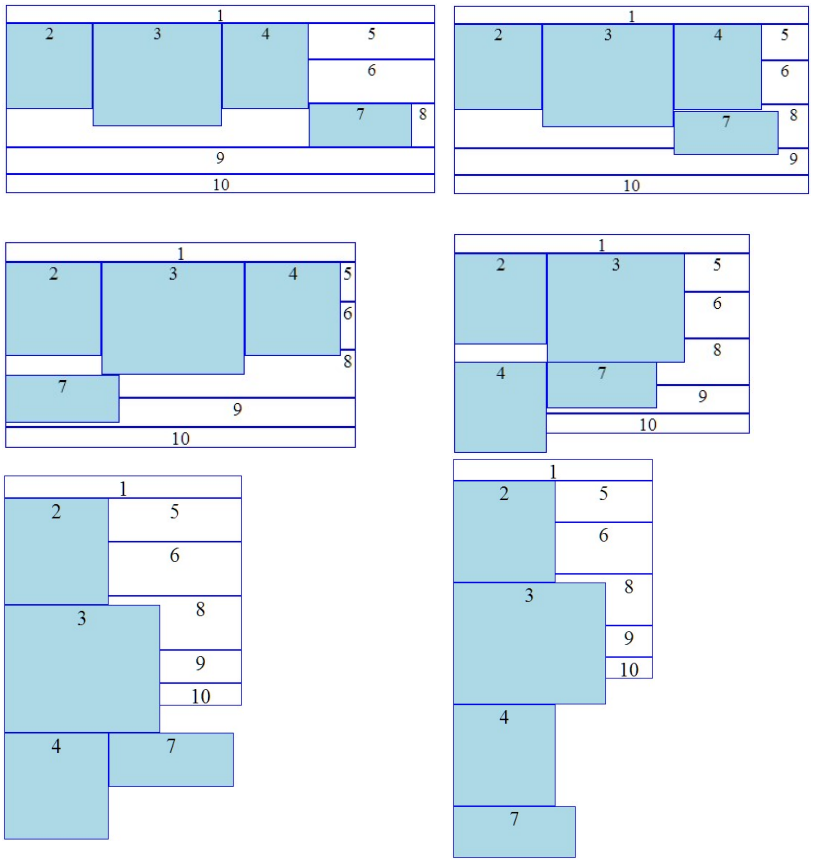
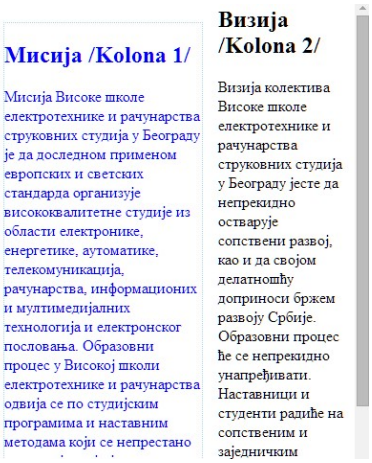
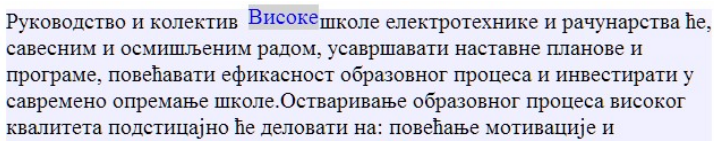
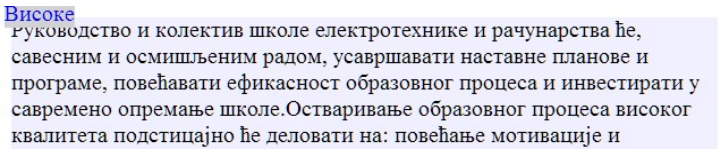
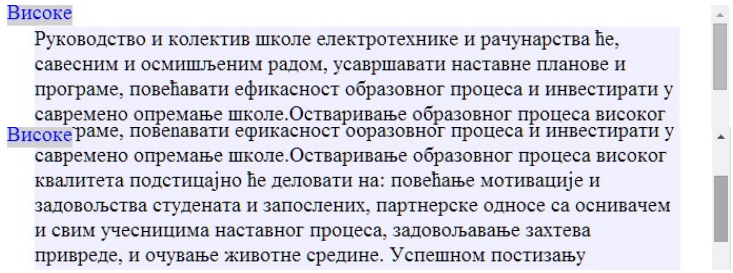
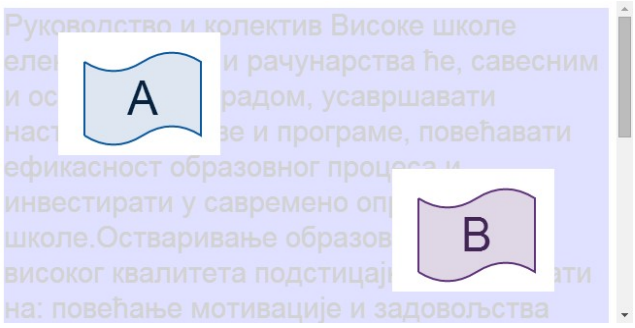
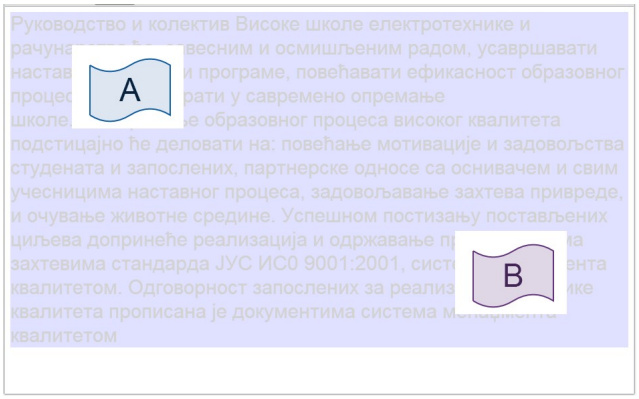
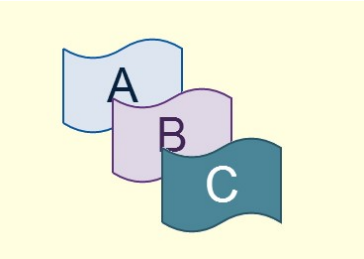
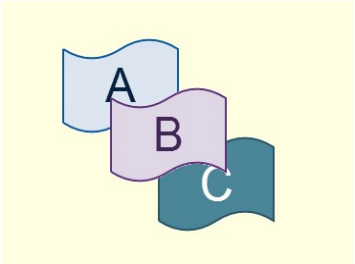
```
h1,h2 { color:red; }
```

Boja će biti primenjena na naslove h1 i h2.
Osim primene grupe selektora, možete se koristiti i bilo koje kombinacije ranije navedenih načina označavanja selektora.

Pregled selektora

- Univerzalni selector *, odgovara svim elementima:
  - `* { font-family:serif; }`
- Selektor tipa, odgovara svim elementima čije je ime u selektoru:
  - `div { font-style: italic; }`
- Konteksni selektori:
  - selektor za sve potomke:
    `p em { color: blue; }` - svi potomci elementa `p` tipa `em`.
  - selektor za dete element:
    `div.glavniOdeljak > p { color: red; }` - selektor za dete element p od elementa `div.glavniOdeljak`.
  - selector za blizance:
    `li + li { color: red; }` - selektuje sledeći element istog nivoa.
- Klasni selektor:
  - `P.novosti { font-size: 8em; }`, odgovara vrednosti atributa `class="novosti"` u svim elementima.
- ID selektor, odgovara tačno jednom elementu sa definisanim ID:
  - `#glavnaVest { font-weight: bold; }`

- Atribut selektor `El[atr]` , odgovara svim elementima koji imaju određeni atribut, ili ako je definisana vrednost u selektoru, onda se odnosi na elemente sa atributom sa tom vrednosti.

    - ```
      table [border] {
         background-color; white;
      }
      ```
      ```

Formatiranje teksta

Tipografija

Web tipografija (eng. typography) je izbor i implementacija fontova u web dizajnu. Po mišljenju mnogih web dizajnera, tipografija je osnova dizajna na webu. U prvim verzijama HTML-a, fontovi su bili kontrolisani isključivo podešavanjima web čitača. Tek od 1995. uveden je element `<font>` koji je kasnije standardizovan. Da bi se prikazao u čitaču, font je tada morao da bude instaliran na korisničkom računaru, inače bi se koristio rezervni font, tj. jedan od podrazumevanih fontova pregledača. Danas se definisanje fonta vrši primenom više CSS svojstava sa velikim brojem opcija, ali i uz podršku specijalnih „online“ servisa za upotrebu ili neposredno instaliranje fontova.

Vrsta fonta

Font je definisan sa nekoliko karakteristika kao što su: oblik, veličina odnosno debljina karaktera. Karakteristike fonta definišu se odgovarajućim CSS svojstvima ili jednim objedinjenim svojstvom kojim je moguće definisati sve osobine fonta.
Vrsta fonta, tj. oblik karaktera definiše se svojstvom: `font-family`. Ukoliko se naziv fonta eksplicitno ne navede, koristi se podrazumevani font koji zavisi od web čitača. Umesto jednog fonta može se navesti lista fontova. Na primer:

```
body{ font-family: Arial; }
p{ font-family:"Courier New",Courier,monospace }
```

Nazivi fontova uvek počinju velikim slovom, a ako naziv fonta sadrži više reči one se postavljaju između znakova navoda. Ako čitač ne prepozna prvi font, pokušaće sa sledećim. Zato se preporučuje da se lista željenih fontova završi sa tzv. generičkom familijom fontova.

Generičke familije fontova

Generičke familije fontova obuhvataju grupe fontova koje imaju zajedničke osobine. U tabeli 3.2 navodimo nazive generičkih familija, neke fontove koji pripadaju tim familijama i kratak opis.

Tabela 3.2. Generičke familije fontova

Familija	Fontovi	Opis	Uvećano
Serif	Times New Roman, Georgia	Fontovi imaju male ukrasne linije na krajevima karaktera.	
Sans-serif	Arial, Verdana	"Sans" znači „bez“. Ovi fontovi nemaju male linije na krajevima karaktera.	
Monospace	Courier New, Lucida Console	Svi karakteri imaju istu širinu.	
Cursive	Comic Sans MS Edwardian Script ITC	Simulira skript ili rukopis.	
Fantasy	Impact STENCIL	U potpunosti dekorativan. Podesan za naslove i posebne namene	

Web fontovi

Web čitač prikazuje neku web stranicu sajta na strani korisnika koristeći fontove koji stoje na raspolaganju. Kada se koriste standardni fontovi, kao što su na primer Times New Roman, Arial, Verdana, Helvetica, tada se sa velikom sigurnošću može očekivati da postoji isti font i na korisničkoj strani.
Međutim, ukoliko to nije slučaj, na korisničkoj strani se mora obezbediti korišćeni font. Ovo se postiže **ugradnjom** fonta u sajt ili odgovarajućim **referisanjem font servisa**.
Ako se font ugrađuje, datoteka koja je izvor za ugrađivanje fonta ne može biti bilo kog formata već je standardizovana. Odgovarajući

format je moguće dobiti iz već postojećih fontova na računaru primenom odgovarajućih programa za izvoz ili preuzimanjem fonta sa specijalizovanih sajtova. Treba zapamtiti da se najbolji kvalitet dobija preuzimanjem sa sajta samog kreatora fonta.
Različiti web čitači podržavaju različite formate fontova. Zato se font obično isporučuje u više formata. Većina čitača koriste formate OpenType (OTF) ili TrueType (TTF), dok Internet Explorer prihvata Embedded Open Type (EOT) format. Osim pomenutih postoji novi standard za upakivanje fontova - Web Open Font Format (WOFF) koji mogu da prihvate svi noviji čitači.
Veliki broj fontova je zaštićen autorskim pravima. Neki su licencirani tako da se mogu koristiti na ograničenom broju mašina i obično „download" i upotreba nisu mogući bez ograničenja!
Postoje dva pristupa koja obezbeđuju korišćenje fontova: samostalno uključivanje fontova ili korišćenjem font servisa.

- Uključivanje fontova. Pošto se font odabere postavlja se na server u svim zahtevanim formatima, obično u odgovarajući folder, a zatim se uključuje u stranicu koristeći @font-face komande.

  Izbor fonta se obično obavlja pomoću specijalizovanih sajtova koji nude fontove, na primer: *fontspring.com*, *fontsquirrel.com*. Uvek voditi računa o uslovima korišćenja!
  Nakon preuzimanja fonta, dobijaju se fajlovi formata TTF, EOT, WOFF, SVG, kao i CSS kod za ugradnju fonta. Na primer:

```
@font-face {
font-family: 'Font_Name';  src: url('myfont-
webfont.eot?#iefix') format('embedded-opentype'),
url('myfont-webfont.woff') format('woff'),
url('myfont-webfont.ttf')  format('truetype'),
url('myfont-webfont.svg#svgFontName')
format('svg');
}
```

  Ugrađeni font se zatim može koristiti pri definisanju stilova, na primer:
  `p {font-family: Font_Name; }`.

Još jednom, mora se voditi računa da se koriste samo fontovi koji su dozvoljeni za web upotrebu.

- **Povezivanje sa font servisima**

  Drugi način za uključivanje posebnih fontova je korišćenje specijalizovanih servisa. Servisi pružaju interfejs i alate koji omogućavaju upotrebu fontova kao i kopiranje.
  Neki od servisa su besplatni i nude fontove visokog kvaliteta, ali takodje nude i opciju licenciranja i zaštite autorskih prava.
  Google Web Fonts
  Google Web Fonts je font servis na lokaciji *www.google.com/fonts*. Jedan je od servisa koji nudi pristup velikom broju besplatnih fontova čak i za komercijalnu upotrebu. Sve što morate da uradite je da odaberete neki font, a zatim kopirate i nalepite kod koji je generisan.



Slika 3.14. Google Fonts

Veličina fonta

Da bi se definisala veličina fonta, slike, margine ili nekog drugog HTML objekta, neophodno je najpre da definišemo mere koje se koriste za definisanje veličine.
U dizajnu web stranica koriste se dve vrste mere:

- apsolutne,
- relativne.

Apsolutne mere

Apsolutne jedinice mere koje se koriste za definisanje veličine su:

- inči (eng. inches), `[in]`,
- centimetri, `[cm]`,
- milimetri, `[mm]`,
- tačke (eng. points), `[pt]`, pri čemu je: `1pt = 1/72in`,
- pikas (eng. picas), `[pc]`, pri čemu je: `1pc = 12pt`.

Veza između navedenih jedinica mere je sledeća:
`1in = 2.54cm = 25.4mm = 72pt = 6pc`.
Pri zadavanju veličine fonta negativne vrednosti nisu dozvoljene, što je i razumljivo.
Apsolutne veličine ne zavise od uređaja na kom se mereni objekat prikazuje, pa se kaže da uvek specificiraju fiksne veličine. Na primer, primenom pravila:

```
p {margin: 1.25in;}
```

širina margine u pasusu će uvek biti `1.25in` bez obzira na veličinu i vrstu uređaja.
Napomena: Apsolutne mere se ne mogu skalirati na klijentskom prikazu.
Savet: Apsolutne mere treba koristiti samo kada je poznata tačna veličina klijentskog prikaza.

Relativne mere

Relativne mere predstavljaju odnos merene i referentne vrednosti, pri čemu referentna vrednost zavisi od uređaja do uređaja. Omogućavaju projektovanje skalabilnih web stranica koje se prilagođavaju različitim veličinama i tipovima ekrana. Po W3C preporukama ove mere treba uvek koristiti tokom projektovanja.
Tu spadaju:

- em
  Ako se drugačije ne navede, veličina fonta je 1em tj. ovo je podrazumevana jedinica za veličinu fonta čitača.
  Inače, tradicionalno predstavlja širinu velikog slova M, tj. nema nikakve veze sa HTML elementom *em*.
  Kada se postavi veličina fonta u em jedinicama, zadata vrednost predstavlja faktor skaliranja podrazumevane veličine, slično procentu. Odnosi se i na horizontalnu i vertikalnu meru. Na primer, ako je veličina fonta u `body` elementu `16px`, a podesi se da veličina fonta u elementu `h1` bude `1.5em` , to znači da će biti `24px`.

- ex
  Zavisi od tekućeg fonta, kao i em, ali se ređe koristi. Predstavlja relativnu meru iskazanu u odnosu na visinu malog slova x (eng. x-height) za tekući font.
  Na primer, ako postoje dva paragrafa sa tekstom iste veličine, na primer `font-size:2em`, ali sa različitim fontom u paragrafu, tada veličine fonta iskazana u ex jedinicama mogu da se razlikuju. Na primer:

  x x

  su mala slova x u fontu Arial, odnosno Courier New, a iste veličine u pikselima, konkretno 18px.

- px
  Poznato je da piksel predstavlja osnovni element slike na displeju, to je najmanja tačka koja može da se prikaže. Takođe, veličina piksela je određena rezolucijom displeja. Prema W3C specifikaciji piksel nije definisan na ovaj način. Zapravo, prvo se uvodi pojam referentnog piksela. **Referentni piksel** predstavlja ugao vidljivosti za jedan piksel na uređaju za prikaz gustine 96dpi, dakle veličine 1/96in=0.26mm na rastojanju od posmatrača koje iznosi 71cm (ili 28in što se smatra dužinom ruke). Taj ugao je 0.0213^{o} stepeni.



Tako na primer, ako je uređaj na rastojanju od 3.5m (na primer TV), dobija se veličina referentnog piksela od 1.3mm. Za nas je od posebnog značaja veličina piksela na mobilnim uređajima. Pošto su takvi uređaji značajno manji, oni se obično i koriste na manjem rastojanju od monitora onda je jasno da je potrebno da veličina piksela bude značajno manja. Međutim, istovremeno sa razvojem tehnologija, pogotovo u oblasti virtuelne stvarnosti, gustina piksela na ovim uređajima dostiže veoma velike vrednosti. U tabeli ispod prikazani su referentni podaci nekih mobilnih telefona radi poređenja.

UREĐAJ	Rezolucija uređaja	CSS PIXELS	DPPI - gustina piksela na uređaju	RATIO	CSS PPI - gustina piksela u dizajnu
Telefoni i mali tableti					
iPhone 3GS	480×320	480	163	1	163
iPhone 4S	960×640	480	326	2	163
Galaxy Nexus	1280×720	598	316	2.14	147.7
Nexus 7	1280×800	966	216	1.33	162
Pad mini	1024×768	1024	163	1	163
Veći tableti					
iPad 1 / 2	1024×768	1024	132	1	132
iPad 3 / 4					
Laptop / Desktop					
Macbook Air 11	1366×768	1366	135	1	135
Macbook pro 15R	2880×1800	1440	220	2	110
Dell 19” Monitor	1600×1200	1600	105	1	105

TVS					
32″ HDTV	1920×1080	1050	69	1.83	37.7

Očigledno je da bi korišćenje fizičkog piksela u dimenzionisanju elemenata na web stranicama na mobilnim telefonima bilo jako teško pošto bi morali voditi računa o velikom broju različitih uređaja i njihovim specifikacijama. Zato pri dizajnu piksel ne predstavlja fizički najmanju tačku već CSS piksel tj. jedinicu mere koja je posebno definisana za svaki uređaj na osnovu veličine ekrana i gustine piksela.
Napomena: Zapazite i to da jedan element definisane veličine od 100px zadržava istu veličinu bez obzira na vrednost zoom-a tokom prikaza.

- %

  Procenti definišu veličinu fonta relativno tj. u odnosu na onu koja bi bila inače tj u odnosu na podrazumevanu. Na primer, pravilo:

  ```
  body {font-size: 150%;}
  ```

  postavlja veličinu fonta za `<body>` element na 1.5 puta veličine fonta koja je podrazumevana u čitaču.
  Važno: Deca elementi nasleđuju vrednosti od roditeljskih elemenata, pa i vrednosti o veličini fonta. Na primer,

  ```
  p {font-size: 12pt;}
  p b {font-size: 125%;}
  ```

  boldovani tekst u pasusu je 125% veći od ostatka teksta u elementu `<p>` koji ga sadrži.
  Procenti kao i `em` su relativne jedinice mere. Razlika je u tome u odnosu na koju veličinu su relativne. Zato važi da je:
  - `em` uvek relativan u odnosu na veličinu fonta, dok je
  - `%` relativan u odnosu na roditeljski blok, obično `body`, `div`, `table`, osim kada se primenjuje na `font-size` u tom slučaju se primenjuju u odnosu na roditeljski font.

- vh, vw

  Novo u jeziku CSS3. Predstavlja relativnu veličinu u odnosu na visinu odnosno širinu prozora. U poglavlju koje se bavi prilagodljivim dizajnom radićemo detaljnije ove mere.

- ključne reči

  Veličina fonta se može definisati i pomoću ključnih reči: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` i `xx-large`.

  Ključne reči ne odgovaraju posebnim merama, već su mere skalirane konzistentno jedna u odnosu na drugu. Podrazumevana veličina je `medium` u važećem čitaču. Na primer:

Slika 3.15. Veličine fontova zadate ključnim rečima

Savet: Jedna od popularnih tehnika upravljanja fontova je postavljanje `font-size` svojstva u elementu `body` koristeći ključne reči. Zatim se koriste relativne izmene na stranici gde god je to potrebno. Ovakva tehnika daje mogućnost lakog i konzistentnog upravljanja veličinom fonta na stranici. Međutim, ključne reči ne nude finu kontrolu na fontovima, pa su mogućnosti ograničene.

Stil, debljina, varijanta

- Za jedan font se dodatno definišu **stilovi** koji mogu biti:
  - `normal` – standardni font,
  - `italic` – kurziv tj. iskošeni,
  - `oblique` – iskošeni, sličan italic stilu.

  Zadaje se primenom `font-style` svojstva, na primer:

```
p.italic {
  font-style: italic;
}
```

- **Debljina** fonta se definiše koristeći svojstvo `font-weight`. Može imati jednu od definisanih vrednost:
  - `bold` – podebljani font,
  - `normal` – standardna debljina,
  - `lighter` – tanki font,
  - `100, 200, ..., 900` (sa korakom od 100).

- **Varijante** slova se odnose na izgled malih slova. Mala slova oblikom postaju ista kao velika slova samo sa veličinom malih. Koristi se naredba `font-variant`, a moguće vrednosti su:
  - `normal` – standarni oblik malih slova
  - `small-caps` – mala slova oblika velikih

  Na primer:
  CSS:

```
  p.normal {font-variant:normal;}
  p.small {font-variant:small-caps;}
</style>
```

HTML:

```
<p class="normal">Ovo je primer za: font-
variant:normal.</p>
<p class="small">Ovo je primer za: font-
variant:small-caps.</p>
```

Ovo je primer za: font-variant:normal.
OVO JE PRIMER ZA: FONT-VARIANT:SMALL-CAPS.

Slika 3.16. Primer primene varijante fonta

Skraćeno definisanje fonta

Umesto da se font definiše navodeći sva CSS svojstva posebno, na primer:

```
p {
  font-weight: bold;
  font-size: 18pt;
  line-height: 24pt;
  font-family: arial;
}
```

moguće je skraćeno zapisivanje, pomoću jednog svojstva `font`. Na primer:

```
p {font: bold 18pt/24pt arial;}
```

Puna sintaksa je za skraćeno zapisivanje je:

`font: font-style font-weight font-variant font-size/line-height font-family`

Skraćeno zapisivanje zahteva poštovanje sledećih **pravila**:

- Svojstva fonta su odvojena razmakom.
- Redosled zadavanja vrednosti je važan.

- Minimalni sadržaj mora imati veličinu i familiju fonta. Izbacivanjem jedne ili izmenom redosleda, zadavanje fonta postaje nevažeće. Primer minimalnog stila je:

  `p { font: 1em sans-serif; }`.

Uz veličinu fonta se može zadati i visina reda, `line-height`, tako što se iza veličine fonta dodaje kosa crta pa visina reda. Visinu reda ćemo kasnije dodatno obraditi.
Ako se u fontu zadaje više familija, kao i ranije, one se zadaju koristeći zapetu:

```
p.stil1 {
  font:15px arial,sans-serif;
}
p.stil2 {
  font:italic bold 12px/30px Georgia,serif;
}
```

Boja teksta

Boja teksta se menja pomoću svojstva `color.` Primenjuje se na HTML elemente koji sadrže tekst, kao što su: paragraf, naslovi i liste. Inače, boja teksta se nasleđuje od roditeljskog elementa, a ukoliko se posebno zadaje, onda se dodeljuje kao vrednost svojstvu `color.` Vrednost je naziv za boju ili njena RGB vrednost. Više o načinima zadavanja boje biće rečeno u nastavku poglavlja.
Primeri zadavanja boje:

`h1 { color: gray; }` – tekst u naslovu h1 se boji u sivo.
`h1 { color: #aabb66; }` – boja je mešavina tri komponente crvene, zelene i plave sa iznosima AA_{hex}, BB_{hex}, 66_{hex} u odnosu na maksimum FF_{hex}. Ako se umesto heksadecimalnog koristi dekadni zapis onda su iznosi 170, 187, 102, a maksimum 255. Ispred heksadecimalne vrednosti koja označava boju stoji znak #.
`h1 { color: #ab6; }` - jednostavno se duplira cifra za svaku boju, pa je boja ista kao u prethodnom primeru.

`h1 { color: rgb(170,187,102); }` - boja se zadaje primenom `rgb` funkcije, ali sa dekadnim vrednostima, dakle u odnosu na maksimalnih 255. Takođe se može koristiti i `%` za oznaku količine boje. Na primer `rgb(66.67%,73.33%,40%);` bi bila boja najsličnija boji u prethodnom primeru.

Pozadinska boja

Koristeći svojstvo `background-color` postavlja se pozadinska boja bilo kog elementa. Na primer:

`background-color : boja;`

Boja se zadaje na identičan način kao što je već objašnjeno. Na primer:

```
p.primer {
  border: 4px dashed;
  color: #409A29;
  background-color: #C3EAF6;
}
```

Visina i uvučenost redova

Visina reda predstavlja rastojanje između baznih linija susednih redova u tekstu. Bazna linija je zamišljena linija na kojoj su smešteni karakteri u jednoj liniji, kao slici 3.16:

Slika 3.16. Visina reda

Mada se visina reda računa od jedne do druge bazne linije, većina čitača ubacuje dodatni prostor iznad i ispod teksta radi razdvajanja dve linije, na taj način centrirajući tekst u jednoj liniji.

`p {font-family:Tahoma, line-height:150%}`

Ista visina reda dobijena zadavanjem vrednosti od `150%` bila bi i zadavanjem vrednosti: `1.5em, 1.5`.

Uvlačenje teksta prve linije (eng. indent) reguliše se svojstvom `text-indent`, a vrednost svojstva se može definisati u jedinicama dužine ili u procentima. Procentualne vrednosti se računaju u odnosu na širinu roditeljskog elementa.
Na primer:

Slika 3.17. Primene svojstva indent

Horizontalno poravnanje

Poravnanje teksta u web stranicama može da bude: na levu stranu, na desnu stranu, centrirano i uz obe strane (kao blok). CSS obezbeđuje navedena poravnanja primenom svojstva `text-align` koje može dobiti vrednosti: `left|right|center|justify`.
Podrazumevano poravnanje je na levo, tačnije za jezike u kojima je čitanje s leva na desno. Primeri horizontalnog poravnanja su dati na slici 3.18.



Slika 3.18. Primeri poravnavanja teksta

Dekoracije

Dekoracije teksta se koriste kada treba da se tekst podvuče, precrta ili postavi crta iznad teksta. CSS svojstvo za dekoraciju je `text-decoration`. Moguće vrednosti svojstva su:

- `underline` – podvučeni tekst,
- `line-through` – precrtani tekst,
- `overline` – crta iznad teksta.

Primeri primene dekoracija dati su na slici 3.19.



Slika 3.19. Primena svojstva `text-decoration`

Napomena: Ovo CSS svojstvo se često koristi da bi se kod linkova isključilo podvlačenje, na primer: `a{text-decoration:none;}`. Međutim, treba voditi računa da se dekoracijom ne mogu isključiti svojstva linkova koja nisu obuhvaćena dekoracijom, kao na primer promena boje.

Transformacija slova

CSS ima mogućnost transformacije teksta u velika ili mala slova. Transformacije se definišu pomoću svojstva `text-transform` koje može imati vrednosti :

- `capitalize` - transformacija prvog slova u reči u veliko slovo, a ostala u mala,
- `lowercase` - transformacija u mala slova,
- `uppercase` - transformacija u velika slova.



Slika 3.20. Primer transformacija teksta

Razmak između reči i slova

Razmak između reči definiše se pomoću svojstva `word-spacing`. Vrednost svojstva može biti iskazana primenom neke od ranije opisanih jedinica mera.



Slika 3.21. Primeri primene svojstva `word-spacing`,

Razmak između slova se definiše pomoću svojstva `letter-spacing` na sličan način kao i u slučaju razmaka između reči.

Vertikalno poravnavanje

Svojstvo kojim se vrši poravnavanje unutrašnjeg elementa po vertikali je `vertical-align`. Vrednosti koje se mogu definisati za ovo svojstvo su:

- `baseline` - poravnavanje sa baznom linijom, ovo je podrazumevano,
- `sub` - poravnavanje u nivou indeksa,
- `super` - poravnavanje u nivou eksponenta,
- `middle` - element je na sredini roditeljskog elementa,
- `bottom` - dno elementa je poravnat sa najnižim elementom na liniji,
- `top` - vrh elementa je poravnat sa najvišim elementom na liniji,
- `%` - podiže ili spušta element u procentu u odnosu na visinu linije,
- `length` - slično kao u slučaju procenta, ali se vrednost pomeranja u odnosu na tekuću liniju zadaje u [px], [em], …



```
log<span style="vertical-align:sub;">10</span>10<span
style="vertical-align:super;">x</span> = x
      R <span style="vertical-align:-200%;">V</span>
    E   <span style="vertical-align:-100%;">I</span>
  S     <span style="vertical-align:0;">S</span>
I       <span style="vertical-align:100%;">E</span>
V       <span style="vertical-align:200%;">R</span>
```

Slika 3.22. Primene svojstva `vertical-align`

Beli prostor

Naziv beli prostor koristi se ravnopravno kao naziv beline ili praznine. Koristi se da označi jednu ili više razmaknica u tekstu, odnosno novi red. Način obrade belina definiše se svojstvom `white-space`, a moguće vrednosti su:

- `pre` – beline se u prikazu javljaju kao i u HTML kodu elementa. Nema prelamanja, osim ako postoji novi red u HTML kodu.
- `pre-wrap` – beli prostor se čuva, a tekst se prelama samo ako je neophodno i to na belinama, ili ako postoji novi red u HTML kodu.
- `nowrap` – tekst se prelama samo ako postoji `<br>` (slično kao kod `pre`), a višestruke beline se odbacuju.
- `pre-line` – višestruke beline se odbacuju. Tekst se prelama kada je neophodno i ako postoji novi red u HTML kodu.

Ovakva značenja mogu da izgledaju prilično konfuzna i teška za razlikovanje. Zato je u nastavku data tabela koja skraćeno ukazuje na značenje pojedinih svojstava.

Tabela 3.3. Pregled svojstava kojima se definišu beline

vrednost	novi red	razmaci i tab.	prelom teksta
`pre`	kao u HTML	kao u HTML	kao u HTML
`normal`	saž. višestrukih	saž. višestrukih	prelom
`nowrap`	saž. višestrukih	saž. višestrukih	nema preloma

`pre-wrap`	kao u HTML	kao u HTML	prelom
`pre-line`	kao u HTML	saž. višestrukih	prelom

Vidljivost

Koristi se da sakrije neki element. Pri tome, prostor koji taj element zauzima, može da se čuva ili ne (`collapse`). CSS svojstvo za određivanje vidljivosti je `visibility`, a moguće vrednosti su:

- `visible` – element je vidljiv,
- `hidden` – element je nevidljiv, , ali prostor za taj element se čuva,
- `collapse` – **važi samo za table elemente,** ponaša se kao vrednost `hidden`. Primenom ove vrednosti izbacuje se red ili kolona tabele, ali se prikaz tabele ne menja tj. prostor zauzet redom ili kolonom biće na raspolaganju drugim elementima. Na primer:

```
<p >Prvi paragraf...</p>
<p style="visibility:hidden;">Drugi paragraf...</p>
<p >Treci paragraf...</p>
```

Drugi paragraf nije vidljiv, mada je prostor za njega ostao rezervisan, kao na slici.



Slika 3.23. Svojstvo `visibility:hidden;`

Kada se radi o tabeli koristi se vrednost `collapse`. Na primer:

```
 <tr>
   <td>11</td>
   <td style="visibility: collapse;">12</td>
```

```
    <td>13</td>
  </tr>
  <tr style="visibility: collapse;">
    <td>21</td>
    <td>22</td>
    <td>23</td>
  </tr>
  <tr>
    <td>31</td>
    <td style="visibility: collapse;">32</td>
    <td>33</td>
  </tr>
</table>
```

Odgovarajuće ćelije postaju nevidljive sa odvojenim prostorom na njihovim pozicijama.



Slika 3.24. Svojstvo `visibility: collapse;`

Svojstvo *display*

Regulisanje vidljivosti elementa kao i načina prikaza vrši se podešavanjem svojstva `display`. Neke najčešće korišćene vrednosti ovog svojstva su:
`block` – prikazuje element kao blok, kao što je na primer pasus,
`inline` – podrazumevana vrednost. Element se prikazuje kao deo linije teksta, na primer kao element `<span>`,
`none` – element se ne prikazuje i takav element ne utiče na prikaz.
Ako bi pokušali da neki tekst učinimo nevidljivim podešavanjem veličine fonta na 0, onda se može dogoditi da neki čitači ignorišu tu deklaraciju zbog nelogične vrednosti.

Markeri uz stavke listi

Markeri su vizuelne oznake koje stoje uz stavke listi. Definisani su tipom nabrojivih odnosno nenabrojivih lista. CSS svojstvo `list-style-type` definiše markere listi. U tabeli 3.4. date su moguće vrednosti, opis i vizuelni oblik markera.

Tabela 3.4. Primeri markera za stavke listi

Vrednost	Opis	Izgled
disc	ispunjeni kružić	• Stavka • Stavka • Stavka
circle	prazan kružić tj. kružnica	○ Stavka ○ Stavka ○ Stavka
square	ispunjeni kvadratrić	▪ Stavka ▪ Stavka ▪ Stavka
decimal	decimalni broj	1. Stavka 2. Stavka 3. Stavka
decimal-leading-zero	decimalni broj sa vodećom nulom	01. Stavka 02. Stavka 03. Stavka
lower-roman	mali rimski brojevi	i. Stavka ii. Stavka iii. Stavka
upper-roman	veliki rimski brojevi	I. Stavka II. Stavka III. Stavka
lower-greek	mala grčka slova	α. Stavka β. Stavka γ. Stavka
lower-latin **lower-alpha**	mala latinična slova	a. Stavka b. Stavka c. Stavka
upper-latin **upper-alpha**	velika latinična slova	A. Stavka B. Stavka C. Stavka

armenian	jermenska azbuka	Ա. Stavka Բ. Stavka Գ. Stavka
georgian	gruzijska azbuka	ა. Stavka ბ. Stavka გ. Stavka
none;	podrazumevano	Stavka Stavka Stavka

Boje u CSS-u

Definisanje boje

Postoje dva načina da se definiše boja preko CSS-a.

- Prvi način je koristeći predefinisana imena boja.

  Na primer:

  ```
  color: red; color: olive; color: blue;
  ```

  Po standardu CSS2.1 postoji 17 standardnih boja, dok je po CSS3 taj broj uvećan na 140 boja.

- Drugi način je koristeći brojnu vrednost koja predstavlja RGB komponente boje koja se definiše.

  Na primer :

  ```
  color: #FF0000; color: #808000; color: #00F;
  ```

  Zapisivanje boje kao numeričke vrednosti

  Zapisivanje boje u CSS-u kao numeričke vrednosti moguće je izvesti na nekoliko načina.

  - Koristeći samo broj, ali sa oznakom **#** ispred broja.

    #RRGGBB

    Tada se zapis boje vrši koristeći tri komponente boje: crvena (oznaka RR), zelena (GG) i plava (BB). Tri komponente se

zapisuju kao tri heksadecimalna broja. Maksimalna vrednost je 255, a minimalna 0, dakle u heksadecimalnom zapisu cifre idu od 00-FF. Na primer za „lavender“ boju zapis bi bio sledeći:
`#C8B2E6`

- Vrednost može biti iskazana i preko decimalnih brojeva ili procenata. U ovom slučaju se koristi funkcija `rgb(r,g,b)`, koja ulazne vrednosti konvertuje u jednu boju.
```
color: rgb(78%, 70%, 90%);
color: rgb(200, 178, 230);
```

Ako se koriste parovi dupliranih cifara za svaku od komponenata boja, kao na primer:
`color: #FFCC00; color: #993366;`
može se za svaku komponentu koristiti samo jedna cifra, pa se za gornji primer može pisati:
`color: #FC0;` ili `color: #936;`

Transparentnost

Osim zadavanja boje koristeći tri komponente boje, u novijim web čitačima može se koristiti i transparentnost tj. prozirnost (eng. opacity). U ovom slučaju se koristi funkcija `rgba`. Ova funkcija važi za IE9+, Firefox 3+, Chrome, Safari, odnosno Opera 10+.
RGBA vrednosti su proširenje RGB vrednosti sa alfa kanalom koji specificira transparentrnost objekta na koji se primenjuje. Sintaksa je: `rgba(red, green, blue, alpha)`.
`Alpha` parametar je broj između 0.0 (potpuna prozirnost) i 1.0 (potpuna neprozirnost). Pogledajmo primer:

```
<h1 style="color:rgba(9,9,9,0.1)">Studenti …
<h1 style="color:rgba(9,9,9,0.3)">Studenti …
<h1 style="color:rgba(9,9,9,0.7)">Studenti …
<h1 style="color:rgba(9,9,9,1)">Studenti …
```



Slika 3.25. Primena transparentnosti u definisanju boja

Boja u prednjem planu

Prednji plan elementa (eng. foreground) se sastoji od sadržaja i ivice, ako je ivica definisana. Prema tome, definisanje boje u prednjem planu znači definisanje boje sadržaja elementa i ivice elementa. Boja ivice se može posebno definisati. Na primer:

```
<style>
  blockquote {
    border: 3px dotted;
    color: darkblue;
  }
 </style>
```



Slika 3.26. Promena boje u prednjem planu

Boja u zadnjem planu

Pozadinska boja ispunjava površ iza elementa koja uključuje prostor sadržaja elementa, ekstra prostor dodat oko sadržaja (eng. padding), proširen ivicom do spoljne granice ivice.
Pomoću svojstva `background-color` vrši se postavljanje pozadinske boje nekog elementa. Sintaksa:

```
background-color: boja ;
```

Podrazumevana vrednost za boju je `transparent`, tj. pozadina elementa je podrazumevano prozirna. Inače, ovo svojstvo se ne nasleđuje.

```
blockquote
{
  border: 3px dotted;
  color: darkblue;
  background-color: yellow;
}
```



Slika 3.27. Promena boje u zadnjem planu

Pozadinske slike

Pozadinska slika se dodaje određenom elementu koristeći svojstvo:
`background-image: url(`*`url_Vrednost`*`);`

Na primer:

```
blockquote {
  background-image: url(viser.png);
  padding: 2em;
  border: 4px dashed;
}
```

Slika 3.28. Pozadinska slika kada je manja od elementa

Napomena: Ako se koristi svojstvo `background-color` zajedno sa svojstvom `background-image`, slika će doći preko boje, tj. prekriće pozadinsku boju stranice.
Međutim, kada se koristi format slike koji podržava transparentost, kao na primer PNG format, tada će pozadinska boja biti vidljiva ispod transparentnih delova slike.

Ponavljanje pozadinske slike

U prethodnom primeru pozadinska slika se ponavlja. To je podrazumevano ponašanje u slučaju kada je slika manja od elementa na koji se primenjuje. Ovo svojstvo ponavljanja se može kontrolisati primenom CSS svojstva `background-repeat`, koje može imati vrednosti:

- `repeat` – podrazumevano, ponavljanje se vrši po x i po y osi.
- `repeat-x` – ponavljanje se vrši po x osi,
- `repeat-y` – ponavljanja se vrši po y osi,
- `no-repeat` – nema ponavljanja.

Pozicioniranje pozadinske slike

Svojstvo `background-position` određuje poziciju slike na pozadini. Ukoliko se slika ponavlja u prikazu, ovo svojstvo određuje poziciju prve sliku u prikazu. Sintaksa je `background-position`, a moguće vrednosti se zadaju na sledeće načine:

- `xpos ypos` – Prva vrednost je horizontalna pozicija, a druga vertikalna. Jedinice su pikseli ili bilo koje druge CSS jedinice mere, pri tome se mogu mešati. Na primer, gornji levi ugao je 0 0. Ako nedostaje jedna vrednost, onda je podrazumevana vrednost 50% (center).
- `x% y%` – Ako se koordinate zadaju u procentima, onda je prvi % pozicija gornjeg levog ugla slike u odnosu na širinu prozora, a drugi % je pozicija donjeg desnog ugla slike u odnosu na visinu prozora. Važno je da se uoči da se ove koordinate odnose ne samo na prozor, već i sliku koja se postavlja. Dakle, ako je zadato 0% 100%, gornji levi ugao slike je u gornjem levom uglu prozora, a donji desni ugao slike je u donjem desnom uglu prozora, tj. slika je potpuno razvučena u prozoru. Ako je 0% 0% slika je skroz levo i skroz gore u

prozoru, ovo je podrazumevano. Međutim, ako je definisana samo jedna vrednost, druga dobija vrednost center.

```
background-color:#aaaabb;
background-image:url('testSlika.jpg');
background-repeat:no-repeat;
background-position: 33% 66%;
```

Slika 3.29. Pozicioniranje primenom procenata

- Ključnim rečima se definiše pozicija slike na razumljiv način. Radi se o nekoliko ključnih pozicija:
  `left top | left center | left bottom | right top | right center | right bottom | center top | center center | center bottom`

„Pričvršćivanje“ pozadinske slike

Kada se postavi pozadinska slika na web stranicu sa sadržajem koji je veći od prozora za prikaz, tada je standardno ponašanje pozadinske slike da se vezuje za celokupan sadržaj tj. sa skrolovanjem dolazi do pomeranja i pozadinske slike.
Ovako podrazumevano ponašanje moguće je promeniti tako da pozadinska slika bude nezavisna tj. da stoji fiksna u pozadini i u slučaju skrolovanja. Ovo se postiže svojstvom `background-attachment`, a moguće vrednosti su: `scroll`, odnosno `fixed`. Podrazumevana vrednost je: `scroll`.



Slika 3.30. Primena svojstva `background-attachment`

Skraćeno definisanje pozadinske slike

Umesto pojedinačnih svojstava, za kompletan opis pozadinske slike, može se koristiti jedno. Sintaksa je:

```
background: background-color background-image
            background-repeat background-attachment
            background-position
```

Važno: Sve vrednosti su opcione i mogu se pisati u bilo kom redosledu. Jedino ograničenje je da se pri navođenju koordinata horizontalna vrednost se navodi prva.
Na primer, ovako bi glasilo jedno pravilo:

```
body {
  background: white url(viser.png) no-repeat
              right top fixed;
}
```

Višestruke pozadinske slike

Novija specifikacija CSS3 dozvoljava istovremeni prikaz nekoliko slika kao pozadinske. U ovom slučaju se podaci svojstava za svaku sliku posebno navode razdvojeni zapetom.

```
body{
  background-image: url('a.png'), url('b.png'),
                    url('c.png'), url('c.png');
  background-position: left top, right top,
                       center center, center center;
  background-repeat: no-repeat, no-repeat,
                       repeat-x, repeat-y;}
```



Slika 3.31. Primena višestrukih pozadinskih slika

Regulisanje prekoračenja

Podrazumevano, visina elementa se podešava tako da se sav sadržaj elementa prikaže. Ako se postavlja veličina nekog elementa, moguće je specificirati način kako će se element prilagoditi eventualnom prekoračenju sadržaja koji prikazuje. Za takvo podešavanje koristi se svojstvo `overflow`. Moguće vrednosti su:

- `visible` – podrazumevano, sadržaj će biti prikazan van definisane oblasti,
- `hidden` – sadržan van definisane oblasti neće biti prikazan,
- `scroll` – sadržaj se pojavljuje u definisanoj oblasti, ali se javlja se skrol-bar koji omogućava vidljivost ostatka sadržaja.

Skrol-bar se javlja i kada je sadržaj manji od definisane oblasti.

- `auto` – skrol-bar se javlja po potrebi tj. samo ako je sadržaj veći od definisane oblasti.

Slika 3.32. Primena svojstva `overflow`

Svojstvo “padding“

Svojstvom `padding` definiše se prostor između sadržaja elementa i ostalog sadržaja. Podrazumevano je ovaj prostor uvek veličine 0. Moguće je posebno definisati veličinu ovog prostora u odnosu na svaku ivicu elementa:
`padding-top`, `padding-bottom`, `padding-right`, `padding-left`.

Takođe postoji i skraćeno zapisivanje koje omogućava zadavanje pojedinačnih vrednosti u jednom svojstvu:

```
padding: top right bottom left;
```

U ovom slučaju sve vrednosti su brojevi i redosled zadavanja je od značaja. Nekada se ovaj redosled pamti kao "TRouBLe" redosled ili smer kazaljke na satu. Moguće je skraćeno zadavanje. Ako se navedu tri vrednosti one su u redosledu:

```
padding: top rightleft bottom;
```

Ako se navedu dve vrednosti:

```
padding: topbottom rightleft;
```

Na kraju ako se navede jedna vrednost, ona se odnosi na sve 4 ivice.
Napomena: Ukoliko se vrednost ovog svojstva navodi u procentima onda se odnosi na procenat širine (čak i kada se odnosi na gornji i donji prostor) roditeljskog elementa.

Stilovi za ivice

Ako ne postoji definisan stil za ivice, ivica ne postoji, tačnije podrazumevani stil je `none`. Drugim rečima, mora se uvek navesti stil ivice koja se prikazuje. Stilovi ivica se primenjuju na svaku ivicu posebno:

```
border-top-style, border-right-style,
border-bottom-style, border-left-style
```

ili koristeći skraćeni opis za sve ivice `border-style`. Najčešće vrednosti koje se dodeljuju stilovima za ivice su:

- `none` – podrazumevano, ivica se ne iscrtava,
- `dotted` – ivica je tačkasta,
- `dashed` – ivica je isprekidana linija,
- `solid` – puna linija,
- `double` – dvostruka linija.

Redosled navođenja vrednosti je od značaja i važe ista pravila kao i kod svojstva `padding`, primenjuje se tzv *„trable"* pravilo, na primer:

- `p.A {border-style: solid dotted dashed double;}` – gornja ivica je puna, desna tačkasta, donja isprekidana a leva dvostruka.
- `p.B {border-style: solid dotted dashed;}` – gornja je puna linija, leva i desna su tačkaste, a donja je isprekidana.
- `p.C {border-style: solid dotted;}` – gornja i donja su pune linije, a leva i desna su tačkaste.
- `p.D {border-style: solid;}` – sve četiri ivice su pune linije.

Plutajući elementi

Podrazumevani prikaz elemenata na HTML stranici znači prikaz elemenata od vrha ka dnu, odnosno s leva na desno, u redosledu kako se navode u HTML dokumentu. Za neke jezike kod kojih se čita s desna na levo, redosled je takođe s desna na levo.
Blokovi elemenata grade stek od vrha i popunjavaju raspoloživu širinu prozora sa elementima. Unutrašnji elementi i karakteri teksta popunjavaju linije svakog bloka posebno, kao na slici 3.33.



Blokovi se pojavljuju u redosledu kao u kodu. Svaki blok ispunjava raspoloživi prostor.

Slika 3.33. Popunjavanje prozora čitača za dve širine prozora

Na ovaj način je obezbeđeno da se elementi ne preklapaju, ne nagomilavaju i popunjavaju prostor prozora čitača. Međutim, moguće je promeniti ovaj standardni način rasporeda elemenata.
Svojstvo `float` prebacuje elemente iz vertikalnog u horizontalni redosled, dozvoljavajući da se ostali sadržaj postavlja oko tih elemenata. Naziv svojstva potiče od prirode postavljanja elemenata kada oni dobijaju svojstvo plutanja nalik objektu u nekom toku.

Ovo je važno svojstvo modernog CSS dizajna i jedno od osnovih alata pri kreiranju prikaza sa više kolona, navigacionih kontrola, prilagodljivog dizajna... **Ovo svojstvo se ne nasleđuje.** Moguće vrednosti su:

- `left` – element je plutajući uz levu ivicu roditeljskog elementa.
- `right` – element je plutajući uz desnu ivicu.
- `none` – podrazumevano.

Pogledajmo na sledećoj slici primere pozicioniranja slike sa i bez plutanja, uz poravnavanje uz levu odnosno desnu ivicu. Slika se poravnava u odnosu na tekst u pasusu.



Slika 3.34. Pozicioniranje slike primenom svojstva `float`

Zapazite da slika menja poziciju u odnosu na normalni tok utičući na okolne elemente. Zato se koristi analogija sa tokom u kome je element koji je označen sa `float` zapravo ostrvo u toku koji ga okružuje.

Važno: Plutajući element jeste deo sadržaja roditeljskog elementa, tj. predstavlja deo roditeljskog elementa smešten u oblast tog elementa. Uz pomoć CSS-a možemo unutar elementa smestiti ne samo sliku, već i neki element koji sadrži tekst, na primer:

```
em {
  float: right;
  margin: 10px;
  width: 200px;
```

```
  color: #FFF;
  background-color: #9D080D;
  padding: 4px;
}
p {
  padding: 15px;
  background-color: #FFF799;
  border: 2px solid #6C4788;
}
```



Slika 3.35. Ubacivanje teksta u pasus

Ponašanje plutajućeg elementa dosledno poštuje pravila koja su navedena.
U narednom primeru je drugi od četiri pasusa dobio svojstvo `float:left` i ograničene je širine. Naredni pasusi popunjavaju prostor prozora web čitača ispunjavajući celu širinu, dok plutajući element ostaje obuhvaćen narednim pasusima tj. odgovarajućim sadržajem.



Slika 3.36. Plutajući pasusi

Isključivanje plutanja

Svojstvo `float` se može pridružiti elementu kako bi element bio plutajući u odnosu na okolni sadržaj. Ako se selektorom odabere više elemenata onda se svojstvo primenjuje na sve njih. Međutim, nekada je neophodno da se ovo svojstvo isključi kako bi se dobilo podrazumevano svojstvo prikaza. Isključivanje plutajućeg prikaza postiže se pomoću svojstva `clear`. Moguće vrednosti su:

- `left` – isključivanje plutajućeg svojstva uz levu ivicu,
- `right` – isključivanje plutajućeg svojstva uz desnu ivicu,
- `both` – isključivanje plutajućeg svojstva uz obe ivice,
- `none` – podrazumevano, bez isključivanja plutajućeg svojstva.

Primenjuje se na blok elemente, a ne nasleđuje se. U narednom primeru je prikazan slučaj kada je neophodno isključiti `float` svojstvo kako bi se naslov `h2`, koji je treći element u kodu, pojavio očekivano ispod slike, a ne u nastavku slike, koja je inače prvi element u kodu.

```
<style type="text/css">
  img {
    float: left;
    margin-right: 10px;}
  h2 {
    clear:none;
    margin-top: 2em;}
</style>
```



Slika 3.37. Primer primene svojstva `clear`

Više plutajućih elemenata

Kada se na jednoj stranici nalazi više plutajućih elemenata, u pozadini web čitača se vrši kompleksno računanje izgleda stranice. Web čitač formira raspored elemenata tako što se u definisanoj visini blok elementa postavljaju plutajući elementi s leva na desno ili obrnuto, kako je definisano svojstvom float.
U narednom primeru je definisano deset odeljaka. Odeljci su označeni brojevima. Svi odeljci imaju širinu 100% širine prozora osim plutajućih koji imaju fiksnu širinu. Plutajući elementi su obojeni da bi se lakše prepoznali na stranici, a podešeno je da budu plutajući uz levu ivicu. Raspored elemenata zavisi od ukupne širine prozora web čitača, pa je prikazan u šest različitih varijanti.



Slika 3.38. Više plutajućih elemenata u različitim slučajevima

Primer horizontalnog menija

Horizontalni meni se u praksi često realizuje primenom plutajućih elemenata. Zbog svog značaja posebno izdvajamo primer koji sledi.

`Horizontalni meni:`

```
<ul>
  <li><a href="#">File</a></li>
  <li><a href="#">Edit</a></li>
  <li><a href="#">Format</a></li>
  <li><a href="#">View</a></li>
  <li><a href="#">Help</a></li>
```

```
</ul>
```

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

```
ul li {
  float: left;
  margin: 1px;
  padding: 5px 10px;
  background-color:gray;
}
```

```
ul li a:link {
  text-decoration: none;
  color:white;
}
```

```
ul li a:visited {
  text-decoration: none;
  color: white;
}
```

Primeniti `clear` *na elemente koji slede iza menija!*

Slika 3.39. Primer realizacije horizontalnog menija

Prikaz u dve kolone

Prikaz sadržaja na web stranicama je organizovan često u vidu kolona. Kolone omogućavaju bolju čitljivost i preglednost sadržaja. Ipak, organizacija sadržaja po kolonama zavisi i od veličine i rezolucije uređaja za prikaz.

Svojstvo `float` se može uspešno iskoristiti kod dizajna stranica u kojima se sadržaj prikazuje u više kolona, zbog prilagodljivosti u odnosu na krajnje uređaje za prikaz.
Za kreiranje dve kolone moguće je primeniti tri scenarija:

1. Jedan `div` element se definiše plutajućim, a dodaje se šira margina sa strane tekst elementa koji okružuje `div`.

```
#kol1 {
  width: 55%;
  float: left;
  padding 1em;
  margin: 1em;
  border: dotted 1px
          lightblue;
  color: blue;
}
```



Slika 3.40. Kreiranja dve kolone

2. Definiše se svojstvo `float` na oba `div` elementa uz levu ili desnu stranu.

3. Definiše se svojstvo `float` na jedan `div` levo, odnosno na drugi `div` desno.

Pozicioniranje

Pri prikazu jedne web stranice elementi popunjavaju prostor s leva na desno i odozgo na dole u zavisnosti od definisane veličine odnosno sadržaja u elementima. Ovo je normalna šema pozicioniranja. Međutim, pozicioniranje elemenata se može eksplicitno definisati koristeći svojstvo `position`. Moguće vrednosti su:

- `static` – podrazumevano. Normalna šema pozicioniranja u kom su elementi pozicionirani kao u normalnom toku dokumenta.
- `relative` - relativno pozicioniranje. Elementi se pomeraju relativno u odnosu na njihovu normalnu poziciju.
- `absolute` - ovako pozicionirani elementi se ne obrađuju u okviru toka dokumenta, a pozicioniraju se u odnosu na prozor čitača, odnosno elementa koji ih sadrži. Na primer, odeljak koji je apsolutno pozicioniran u drugom odeljku ne utiče na njegovu veličinu. Ovako pozicionirani elementi ne utiču na vizuelni raspored okolnih elemenata, tj. zasebni su u odnosu na druge.
- `fixed` – elementi ostaju na istoj poziciji i tokom skrolovanja.

Specificiranje pozicije

Specificiranje pozicije se sastoji od definisanja načina pozicioniranja i od postavljanja vrednosti. Vrednosti za pozicije se postavljaju koristeći svojstva:

`top, right, bottom, left`

kojima se pridružuju vrednosti u obliku: broja, procenta ili `auto` (podrazumevana vrednost). Za specificiranje pozicije ne postoji nasleđivanje.

Statičko pozicioniranje

Statičko pozicioniranje je podrazumevano pozicioniranje. Eksplicitno zadavanje ovog načina pozicioniranja, `position:static;` , gotovo se nikada ne radi. Zato ovde nećemo davati poseban primer pošto su prethodni bili oslonjeni baš na ovakvo pozicioniranje.

Relativno pozicioniranje

Relativno pozicioniranje vrši pozicioniranje u odnosu na poziciju koja bi inače bila tj. u odnosu na podrazumevanu poziciju.
Važno: Pomereni element ne remeti raspored ostalih elemenata u normalnom toku. Dakle, može se dogoditi preklapanje elemenata.
U narednom primeru izvršeno je relativno pozicioniranje dela teksta iz jednog pasusa. Ovaj tekst je označen plavom bojom i pomeren je u odnosu na svoju podrazumevanu poziciju koristeći svojstva `top` i `left`.

`Relativno pozicioniranje:`

```
position: relative;
top: -0.2em;
left: 0.2em;
color: blue;
background-color:lightgray;
```



Slika 3.41. Relativno pozicioniranje

Apsolutno pozicioniranje

Apsolutno pozicioniranje se obavlja u odnosu na poziciju roditeljskog elementa.

Važno: Apsolutno pozicionirani element je potpuno izbačen iz normalnog toka u dokumentu.

Ovo je važna razlika u odnosu na relativno pozicioniranje. Dakle, ostali elementi u toku će popuniti prostor koji bi bio podrazumevano zauzet.

U narednom primeru označena reč je pozicionirana u odnosu na poziciju pasusa:

`Apsolutno pozicioniranje:`

```
position: absolute;
top: -0.2em;
left: 0.2em;
color: blue;
background-color:lightgray;
```



Slika 3.42. Apsolutno pozicioniranje

Kontekst pozicioniranja

Pozicija i veličina HTML elementa se određuje u odnosu na element koji ga sadrži. Zato je pri pozicioniranju elemenata važno imati u vidu u odnosu na koji element se vrši apsolutno pozicioniranje. Ovo se naziva kontekst pozicioniranja.

a. Ako element nije sadržan unutar drugog pozicioniranog elementa, tada će biti relativno pozicioniran u odnosu na inicijalni HTML blok.

b. Ali, ako element ima pozicionirane pretke elemente, koji imaju relativnu, apsolutnu ili fiksnu poziciju, element se pozicionira relativno u odnosu na ivice tog elementa.

Fiksno pozicioniranje

Fiksno pozicioniranje predstavlja apsolutno pozicioniranje elementa u odnosu na prozor web čitača, tako da se pozicija ne menja bez obzira na poziciju skrol-bara. U narednom primeru, dodata je margina za pasus veličine 2em, a pozicioniranje postavljeno na vrednost `fixed`. Na slici je prikazana pozicija označene reči za dve pozicije skor-bara.

```
position: fixed;;
top: -0.2em;
left: 0.2em;
color: blue;
background-color:lightgray;
```



Slika 3.43. Fiksno pozicioniranje

Pozicioniranje u procentima

Pozicioniranje u procentima može biti apsolutno, relativno ili fiksno, ali se vrednost pozicije definiše procentom u odnosu na roditeljski element ili blok. Ako roditeljski element ne postoji, onda se procenat računa u odnosu na dimenzije prozora web čitača. Na primer:

```
#a{
position: absolute;
top: 10%;
left: 10%;
}
#b{
position: absolute;
bottom: 10%;
right: 10%;
}
```



Pasus nije pozicioniran, pa su pozicije slika u pasusu definisane u odnosu na prozor čitača.



Pasus je pozicioniran, pa su pozicije slika u pasusu u odnosu na sam pasus.

Slika 3.44. Pozicioniranje u zavisnosti od pozicije roditeljskog elementa

HTML elementi **`img`** se nalaze u pasusu koji se vidi na slici. U prvom slučaju, pasus nije pozicioniran, pa se pozicija računa u odnosu na web čitač, dok je u drugom slučaju pasus pozicioniran, pa se pozicija postavlja u odnosu na pasus.

Z-indeks

Elementi koji koriste apsolutno ili relativno pozicioniranje mogu da se preklope u prikazu tj. može se dogoditi međusobna kolizija. Prioritet u prikazu preklopljenih elemenata je definisan tzv. z-indeksom. Element koji ima prednost u prikazu preklapa onaj drugi element i ima vizuelni efekat elementa koji je iznad.
Podrazumevano, elementi koji su kasnije definisani imaju veći prioritet u prikazu tj. preklapaju elemente koji su prethodno definisani. Na primer:

```
<img id="a" src="a.png"/>
<img id="b" src="b.png"/>
<img id="c" src="c.png"/>
```



Preklapanje dobijeno apsolutnim pozicioniranjem, bez uticaja na z-indeks

Slika 3.45. Preklapanje slika

Redosled koji je podrazumevan može se promeniti podešavanjem svojstva `z-index`. Brojevi sa većim indeksom će biti iscrtavani posle onih sa manjim brojem i prirodno će biti pozicionirani iznad njih. Ukoliko u prethodnom primeru želimo da obrnemo redosled potrebno je uvesti sledeće stilove:

```
#a{ z-index: 2; }
#b{ z-index: 3; }
#c{ z-index: 1; }
```



Slika 3.46.Promena z-indeksa

Pitanja za proveru znanja

1. Objasniti osnovnu strukturu CSS zapisa.
2. Navedi različite delove sledećeg stila:

   `body { background-color: black; }`

   selektor:_____

   svojstvo: _____

   vrednost: _____

   deklaracija: _____

3. Kako se obrađuju praznine u CSS kodu?
4. Kako se pišu komentari u CSS kodu?
5. Koja su tri načina povezivanja CSS koda i HTML dokumenta?
6. Koji je način povezivanja i zašto najčešći u praktičnim realizacijama?
7. Na dva načina je moguće realizovati spoljašnje povezivanje. Koji su to načini i kakva je razlika između njih?
8. Objasni osobinu nasleđivanja kod CSS stilova.
9. Šta znači konflikt u primeni stilova i šta znači kaskadno rešavanje konflikata?
10. Koji je redosled primene stilova definisanih ako postoje tri različita načina povezivanja?
11. Koja boja paragrafa će biti nakon primene sledećeg stila? Zašto?

```
<style type="text/css">
  p { color: reg; }
  p { color: green; }
  p { color: blue; }
</style>
```

12. Kako se postiže da neki stil ima najveći prioritet?
13. Objasni box model.
14. Kako se računa ukupna veličina elementa na stranici ako su zadata svojstva `width`, `padding`, `border` i `margin`?
15. Šta definišu sledeći selektori:
    a. `{ color:blue; }`,
    b. `h2#glavni { color:red; }`,
    c. `em.myClass { color:green; }`,

d. `a:visited { color:red; }`,
e. `h1 em { color: blue }`,
f. `h2[name] { color:red; }`,
g. `h2[class ^= "nsl3"]`,
h. `a[href *= "viser"]`,
i. `p::first-line { color:lightblue;}`,
j. p[class="main"][naziv="prvi"],p[class="main"][naziv="drugi"] { color: red; }.

16. Kako se definiše veličina fonta?
17. Objasniti razlike između apsolutnih i relativnih mera.
18. Koje jedinice mere se koriste za apsolutne, a koje za relativne mere?
19. Koji stilovi se koriste za definisanje visine linije, a koji za dekoraciju teksta?
20. Navedi primer primene stila kojim se reguliše vidljivost elementa.
21. Objasni svojstvo `display`.
22. Formatirati jednu nabrojivu i jednu nenabrojivu listu.
23. Kako se zadaje pozadinska boja elementa?
24. Koji od navedenih načina zadavanja boje nije primenljiv u CSS-u?
    a. #FFFFFF b. #FFF c. rgb(255, 255, 255)
    d. rgb(FF, FF, FF) e. white f. rgb(100%, 100%, 100%)
25. Šta znači ponavljanje pozadinske slike i kako se reguliše?
26. Kako se pozadinska slika može na različite načine „pričvrstiti“ za element?
27. Objasni svojstvo `overflow` za regulisanje prekoračenja.
28. Kako se ponašaju elementi koje nazivamo plutajućim?

29. Da li se plutajući element može ubaciti u pasus?
30. Kako se svostvo plutanja može prekinuti?
31. Napisati sopstveni primer sa više plutajućih elemenata i prikazati njihov raspored u zavisnosti od širine prozora.
32. Kako se realizuje horizontalni meni? Navedi primer.
33. Objasni šta je i kako se pomoću stilova menja z-indeks?

Rezime

Selektori:

- univerzalni:
  - `* { font-family:serif; }`,
- selektor tipa:
  - `div { font-style: italic; }`,
- konteksni:
  - selektor za sve potomke:
    `p em { color: blue; }`,
  - `selektor` za dete element:
    `div.glavniOdeljak > p { color: red; }`,
    `div.glavniOdeljak`,
  - `selector` za blizance:
    `li + li { color: red; }`,
- klasni:
  - `P.novosti { font-size: 8em; }`,
- ID selektor:
  - `#glavnaVest { font-weight: bold; }`,

- atribut selektor:
  - ```
    table [border] {
        background-color; white;
    }.
    ```

Redosled primene stilova:

1. podrazumevani stilovi browser-a,
2. korisnički definisani stilovi u browser-u,
3. spoljni stilovi,
4. importovani stilovi,
5. ugrađeni stilovi,
6. linijski,
7. važni stilovi.
    ```

Deo 4: Vrste dizajna

- Vrste dizajna
- Fiksni prikaz
- Fluidni dizajn
- Elastični dizajn
- Hibridni dizajn
- Pozadinska dekoracije

U ovom poglavlju biće prezentovano nekoliko praktičnih primera i različitih načina kreiranja web stranica. Primeri su zasnovani na primeni CSS-a koji je obrađen u prethodnom poglavlju. Naglasak je dat na dizajnu web stranica sa više kolona i u nekoliko varijanti. Imajući u vidu sve veću raznolikost krajnjih uređaja za prikaz poseban akcenat je na tehnikama prilagodljivog dizajna.

Vrste dizajna

Web stranice se prikazuju na različitim uređajima i rezolucijama, od malih na mobilnim uređajima do onih ogromnih na bioskopskim platnima.
Osim zahteva da prikaz bude adekvatan za tako različite „uređaje“, neophodno je i da korisnik može da podešava veličinu prikazanog teksta koji takođe utiče na prikaz stranice. Vremenom se izdvojilo nekoliko standardnih prikaza stranica koji se koriste u različitim slučajevima. To su:

- fiksni prikaz – fiksna širina stranice u pikselima bez obzira na veličinu čitača ili veličinu teksta.

- fluidni prikaz – prikaz se menja istovremeno sa promenom veličine prozora čitača.
- elastični prikaz – promena prikaza je proporcionalna, zasnovana na veličini fonta.
- hibridni prikaz – kombinuje više vrsta prikaza.

Fiksni dizajn

Fiksni prikaz, kako naziv govori, kreira se tako da je širina stanice iskazana u pikselima nepromenljiva. Ovako dizajnirana stranica dozvoljava laku kontrolu: pozicija svih elemenata, njihovih veličina, poravnavanja i dužine linije. Ovaj tip prikaza je postao popularan u praktičnoj primeni, usled činjenice da su korisnici do skoro imali tradicionalni pogled na Web - samo sa računarskih monitora. Takođe, dizajnerima se omogućava da prikaz koji projektuju u fazi razvoja tačno odgovara prikazu na svakom ekranu u fazi eksploatacije. Naravno, ovakav dizajn ne može da odgovori sve većim potrebama korisnika u pogledu raznovrsnosti krajnjih uređaja.
Kada se dizajnira stranica sa fiksnim prikazom potrebno je unapred utvrditi nekoliko podataka.
Prvo treba odrediti širinu stranice. Imajući u vidu da je krajnji uređaj za prikaz računarski monitor, bira se širina koja će biti manja od većine monitora, a opet dovoljna za kvalitetnu prezentaciju. Obično se bira širina od 800 do 960 piksela. Izbor širine je, prvenstveno, dizajnerska odluka.
Pošto je širina fiksna, ona je obično manja od širine prozora za prikaz. Podrazumevano, poravnavanje bi bilo uz levu ivicu, a celokupan prazan prostor bi bio između desne ivice stranice i desne ivice prozora. Dakle, drugi podatak koji se mora utvrditi je pozicija stranice u odnosu na širinu prozora. Najčešće se koristi centralna pozicija stranice tj. polovina praznog prostora se pojavljuje sa leve, a polovina sa desne strane.

Kreiranje

Postoji više načina za definisanje web stranica fiksnog prikaza. Tipično, sadržaj cele stranice se stavi u jedan odeljak `div` (koji se obično naziva content, container, wrapper ili "page"), a onda se za taj `div` postavlja određena širina u pikselima. Ovaj `div` može biti centriran u prozoru čitača. Širine elemenata u koloni, margina i pedinga se takođe zadaju u pikselima.
U naredna dva primera biće pokazano kreiranje stranice sa dve kolone.

Primer: Dve kolone sa apsolutnim pozicioniranjem.

1. Najpre se definiše odeljak širine 900 piksela.

```
#container{
  width:900px;
  ...
}
```

Slika 4.1. Kreiranje kontejnera

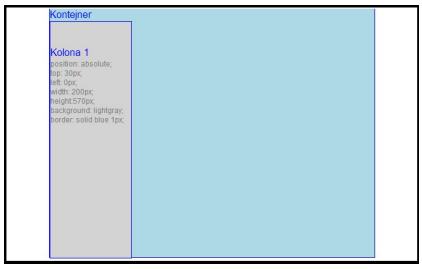
2. Zatim se uradi centriranje odeljka postavljanjem bočnih margina na vrednost `auto`.

```
#container{
  width:900px;
  ...
  margin:0 auto;
}
```

Slika 4.2. Centriranje kontejnera

3. Zatim se dodaje prva kolona pozicionirana sa leve strane kontejnera i širine od 200 piksela. Dakle radi se o apsolutnom pozicioniranju u odnosu na kontejner. Zbog toga treba `#container` modifikovati sa `position: relative;`

```
#container {
  ...
  position: relative;
}
#col1 {
  position: absolute;
  left: 0px;
  width: 200px;
}
```



Slika 4.3. Dodavanje prve kolone

4. Druga kolona se dodaje takođe apsolutnim pozicioniranjem u odnosu na kontejner. Pozicioniranje se vrši podešavanjem leve margine u odnosu na kontejner. Imajući u vidu da je širina prve kolone 200px, margina treba da bude veća od 200px za prostor kojim će biti razdvojene kolone. U našem slučaju to je urađeno primenom `margin-left:220`. Isti efekat bi bio primenom istog svojstva kao i u slučaju prve kolone.

```
#col2 {
  position: absolute;
  margin-left:220;
 /* ili left: 220px; */
  width: 680px;
}
```

Slika 4.4. Dodavanje druge kolone

Primer: Dve kolone fiksne širine uz primenu svojstva `float`.

Ovaj primer prezentuje način kreiranja više kolona pozicioniranjem uz primenu svojstva `float`. Takođe, primer će obuhvatiti jedan način kreiranja navigacije i futera.

1. Prvo se formira kontejner za sve sadržaje na stranici.

```
#container{
  margin: 0 auto;
  width: 900px;
}
```

Slika 4.5. Kreiranje kontejnera

2. U kreirani kontejner se ubace odeljci za: zaglavlje, navigaciju i sadržaj. HTML i CSS kod kao i šematski prikaz ključnih vrednosti iz koda prikazani su u nastavku. Deo za navigaciju, zbog veličine koda, ostavljen je za sledeći korak.

HTML:

```
<body>
  <div id=”container”>
    <div id=”header”> . </div>
    <div id=”navigation”> . </div>
    <div id=”content”>
      <div id=”col1”> .</div>
      <div id=”col2”> .</div>
      <div id=”footer”> .</div>
    </div>
  </div>
</body>
```

CSS:

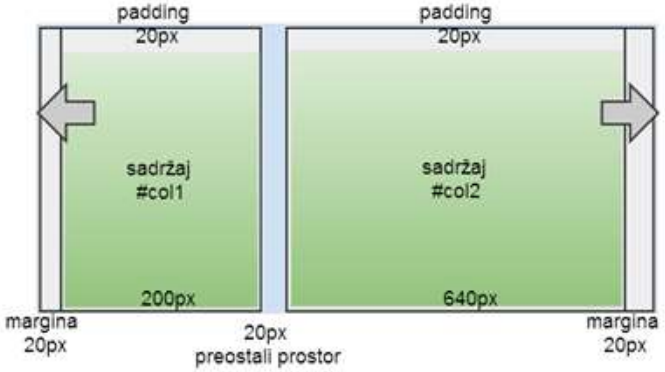
```
#header{
  background: #ddd;
  padding: 20px;
}
#content{
  float: left;
  width: 900px;
  background: #fff;
}
#col1{
  float: left;
```

```
#col2{
  float: right;
  width: 640px;
  padding: 20px 0;
  margin: 0 20px 0 0;
}

#footer{
  clear: both;
  background: #ddd;
```

```
  width: 200px;
  padding: 20px 0;
  margin: 0 0 0 20px;
}
```

```
  padding: 10px;
  padding: 10px;
}
```

Prva kolona je plutajuća uz levu ivicu, a druga uz desnu ivicu kontejnera. Margine su iskorišćene za definisanje razmaka između kolona i bočnih ivica, kao i međusobnih rastojanja.



Slika 4.6. Izgled stranice sa označenim odeljcima

Svakako da prikazano dodavanje kolona nije jedini način da se dve kolone organizuju u željeni raspored. Recimo, da druga kolona može da bude plutajuća uz levu ivicu.

1. Na kraju, kod dopunjujemo navigacijom koju smo u nekom od ranijih poglavlja razmatrali.

HTML:

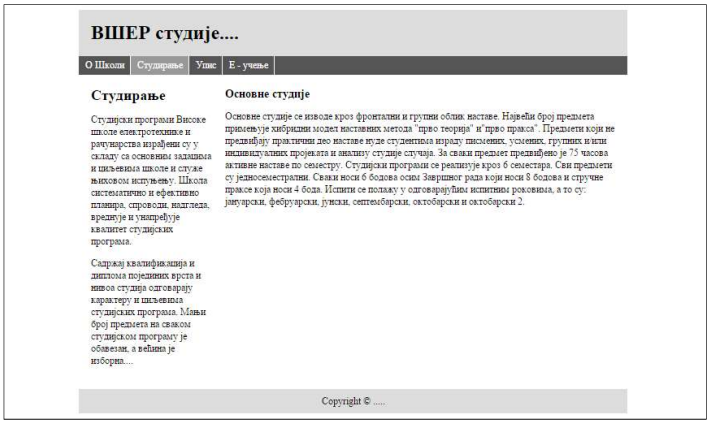
```
<div id="navigation">
  <ul>
    <li><a href="#">О Школи</a></li>
    <li><a href="#">Студирање</a></li>
    <li><a href="#">Упис</a></li>
    <li><a href="#">Е - учење</a></li>
  </ul>
</div>
```

CSS:

```
#navigation{
  float: left;
  width: 900px;
  background: #555;
}
#navigation ul {
  list-style-type:
  none;
  margin: 0;
  padding: 0;
}
#navigation ul li{
  display: inline;
}
```

```
#navigation li a{
  display: block;
  float: left;
  padding: 5px 10px;
  color: #fff;
  text-decoration:
               none;
  border-right: 1px
         solid #fff;
}
#navigation li a:hover {
  background: #999;
}
```

Konačan izgled sa navigacijom i futerom dat je na slici 4.7.



Slika 4.7. Konačan izgled web stranice

Primer: Zaglavlje i futer duž cele širine prozora

Ukoliko je zaglavlje i futer potrebno prikazati u punoj širini prozora, a istovremeno sačuvati da sadržaj bude između njih, potrebno je

izmeniti prethodni scenario tako da se sekcije `header` i `footer` izmeste iz kontejnera. CSS kod bi ostao ne promenjen.

HTML:

```
<body>
  <div id=”header”> . </div>
  <div id=”container”>
    <div id=”navigation”> . </div>
    <div id=”content”>
      <div id=”col1”> .</div>
      <div id=”col2”> .</div>
    </div>
  </div>
  <div id=”footer”> .</div>
</body>
```

Slika 4.8. Zaglavlje i futer duž cele širine prozora

Važno: Bez obzira što je kod većine monitora rezolucija dovoljna da se prikaže celokupan sadržaj, ipak to nije uvek slučaj. Takođe, promenom parametara na web čitaču, kao što su veličina fonta ili širina prozora može biti nedovoljna da se obuhvati celokupna širina stranice. Tada deo sadržaja sa desne strane neće biti vidljiv na prvi

pogled. Pojaviće se horizontalni skrol-bar pomoću koga se može pristupiti delu sadržaja koji nije vidljiv.

Primer: Tri kolone sa apsolutnim pozicioniranjem, bez futera

U ovom slučaju, koristi se apsolutno pozicioniranje za podešavanje tri kolone u web stranicama sa fiksnim širinama. Pošto visine kolona nisu unapred poznate, nije moguće predvideti poziciju futera, pa je taj odeljak izbačen iz primera.

HTML:

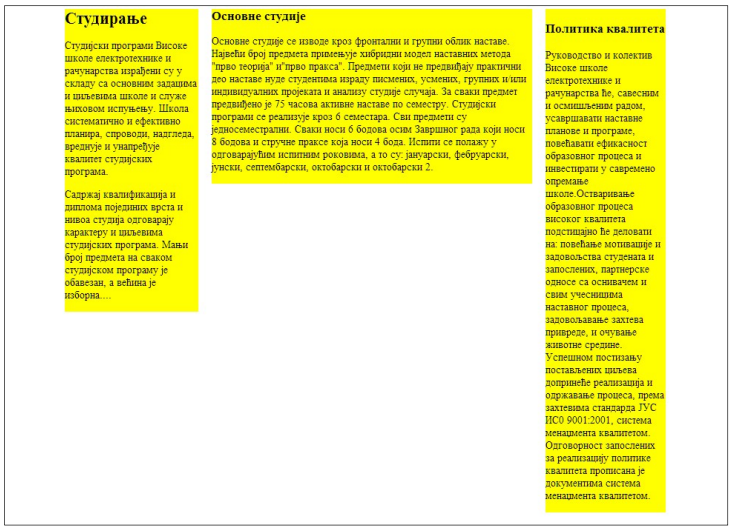
```
<div id="container">
  <div id="content">
    <div id="col1"> </div>
    <div id="col2"> </div>
    <div id="col3"> </div>
  </div>
</div>
```

CSS:

```
#container {
  margin: 0 auto;
  width: 900px;
  border: green 2px solid;
}
#content {
  position: relative;
  width: 900px;
  border:red 2px solid;
}
#col1 {
  position: absolute;
  top: 0; left: 0;
  width: 200px;
}
```

```
#col2 {
  position: absolute;
  top: 0; left: 220px;
  width: 480px;
}
#col3 {
  position: absolute;
  top: 0; right: 0;
  width: 180px;
}
```

Apsolutnim pozicioniranjem mora se unapred precizno definisati raspored i širine kolona. Primenom gore navedenih vrednosti dobija se izgled stranice kao na slici.



Slika 4.9. Fiksni dizajn apsolutnim pozicioniranjem

Fiksni dizajn sa organizacijom sadržaja po kolonama ima svoje prednosti i nedostatke:

- **Prednosti:**
  - Predvidivost prikaza i bolja kontrola dužine linije.
  - Lakši za dizajn tj. programiranje.
  - Predstavlja najčešći šablon za dizajn web stranica, dovoljno dobar za web prikaz na desktopu.

- **Nedostaci**:
  - Sadržaj uz desnu ivicu će biti sakriven ako je širina prozora čitača manja od stranice.
  - U slučaju širih ekrana pojavljuje se višak prostora sa leve strane.
  - Dužine linija teksta mogu narasti toliko da budu neprijatne.
  - Korisnik nema kontrolu prikaza.

Optimalna dužina linije

Dužina linije je mera iskazana brojem reči ili karaktera u jednoj liniji teksta. Postoji pravilo da je optimalna dužina linije orijentaciono od 10 do 12 reči ili od 60 do 75 karaktera.
Ako linija teksta postane duga, tekst postaje težak za čitanje. Ne samo zato što je teško fokusirati tekst do kraja linije, već je potreban i dodatni napor da bi se započelo čitanje sledećeg reda.
Dužina linije je često odlučujuća za određivanje dizajna stranice. U odnosu na fiksni i fluidni dizajn, elastični dizajn ima najbolje osobine u pogledu očuvanja dužine linije teksta u različitim slučajevima.

Fluidni dizajn

U fluidnom dizajnu, oblast stranice, odnosno kolone u stranici imaju promenljivu širinu tako da popunjavaju celokupan prozor čitača. U ovom dizajnu stranica se popunjava na podrazumevani način tj. primenjuje se normalni tok prikaza elemenata u stranici. Ne vrši se kontrola širine ili prelom linija tj. dozvoljeno je da tekst menja tok u skladu sa medijumom za prikaz.
Dobar primer fluidnog dizajna je stranica www.w3.org, koja je prikazana na slici 4.10.

Slika 4.10. Prikaz www.w3.org u dve širine prozora

Kaže se da fluidni dizajn stranica predstavlja stil dizajna koji **vodi računa o korisnicima**. Danas postoji velika raznovrsnost čitača, posebno onih koji su prilagođeni uređajima koji su manji od tradicionalnih monitora, što proizvodi sve veću pažnju ka ovoj vrsti dizajna.

Kreiranje

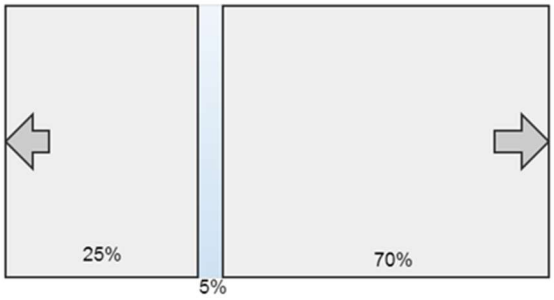
Kreiranje web stranice, ili odeljka na stranici, koji treba da bude potpuno popunjen podelementima odnosno njihovim sadržajem, može se izvesti bez dodatnog kodovanja. Na primer, doslednim poštovanjem da se u dizajnu ne zadaje širina elementa, tj. da se ne koristi svojstvo `width`. Tako definisani elementi poprimaju podrazumevanu širinu tj. širinu koja odgovara vrednosti `auto` za svojstvo `width`.
Ako se kreira sadržaj u više kolona, potrebno je prvo definisati kolone čija će širina biti proporcionalna širini prozora. Ovakav dizajn podrazumeva definisanje širine kolona u procentima u odnosu na ukupnu širinu prozora.

Moguće je na više različitih načina organizovati sadržaj stranice po kolonama. U nastavku biće prikazana dva primera sa dva načina za organizovanje po kolonama u fluidnom dizajnu.

Primer: Sadržaj u dve kolone

Prva kolona koristi svojstvo `float:left`, a druga `float:right`. Pri određivanju širine kolona mora se uzeti u obzir i rastojanje između kolona.

Neka je leva kolona ukupne širine 25% širine prozora, a desna kolona je 70% širine prozora, dok se preostalih 5% se koristi kao razmak između kolona, bez obzira na veličinu prozora. Šematski je ovo prikazano na slici 4.11.



Slika 4.11. Šematski prikaz organizacije stranice

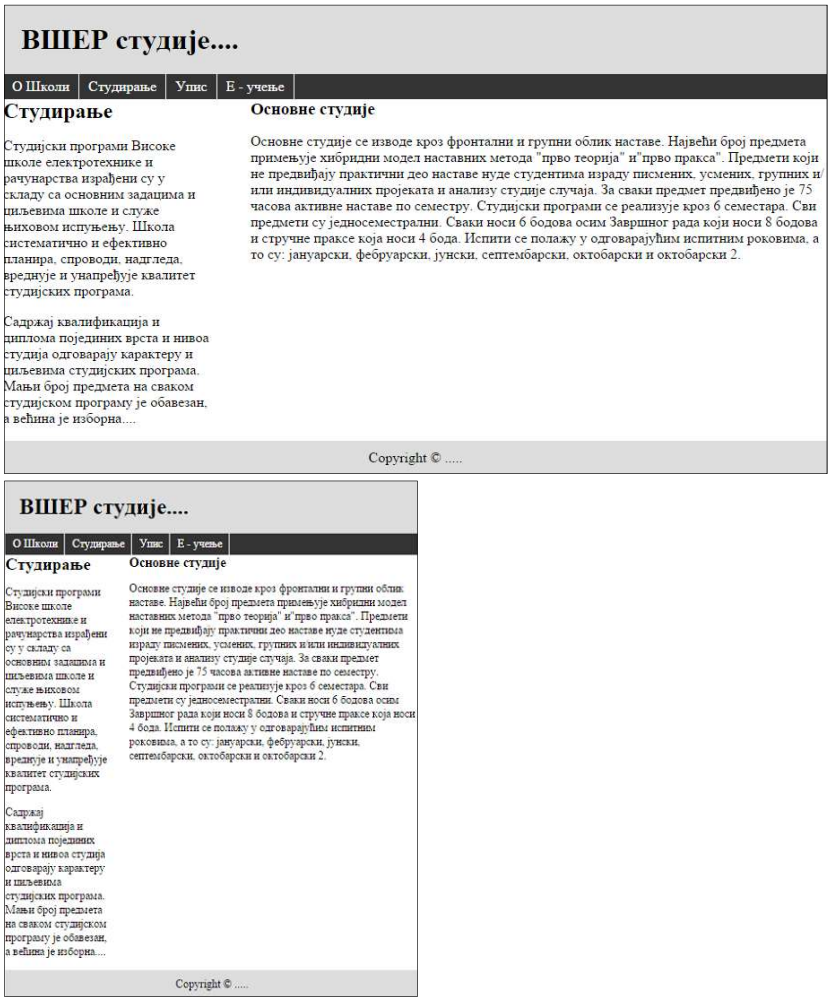
Kada se dodaju ostale sekcije za zaglavlje, navigaciju i futer CSS kod je:

```
#container{
  margin: 0 auto;
  width: 100%;
}
#header{
  background: #ddd;
  padding: 20px;
}
#navigation{
  float: left;
  width: 100%;
  background: #333;
}
#content{
  float: left;
```

```
#col1{
  float: left;
  width: 25%;
}
#col2{
  float: right;
  width: 70%;
}
#footer{
  clear: left;
  background: #ddd;
  text-align: center;
  padding: 10px;
}
```

```
  width: 100%;
  background: #FFF;
}
```

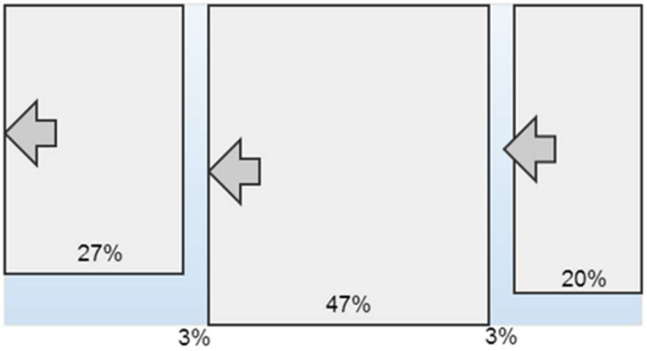
Dobijena web stranica je prikazana na narednoj slici. Stranica poseduje fluidni dizajn sa dve kolone. Širine kolona, kao i margine su proporcionalne širini prozora čitača.



Slika 4.12. Fluidni dizajn sa dve kolone

Primer: Tri (ili više) kolona, sve kolone uz levu ivicu

Prvo se definišu širine kolone. Neka se, na primer, projektuje web stranica sa tri kolone sa širinima od 30%, 50% i 20%.
Zatim se odredi rastojanje između kolona. To rastojanje takođe treba da bude izraženo u procentima, i ono utiče na smanjivanje širine kolone za sadržaj, za isti procenat. Neka je to rastojanje 3%.
Na slici 4.13. šematski je prikazan raspored kolona.



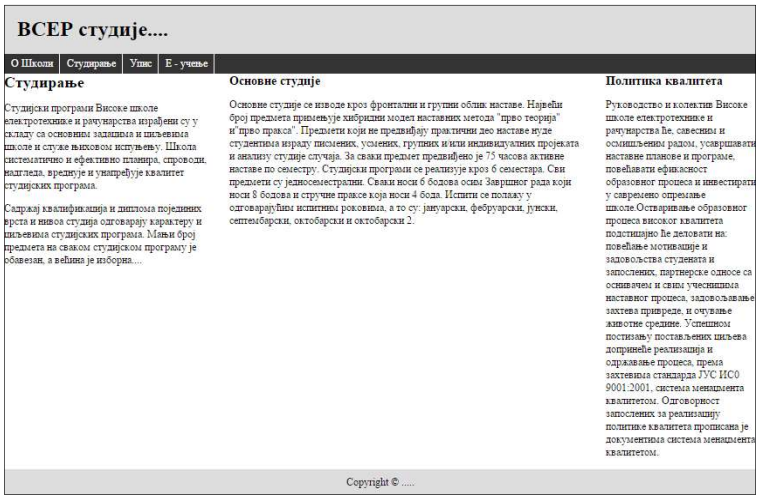
Slika 4.13. Raspored kolona u kontejneru

Sve kolone u kontejneru imaju svojstvo `float:left`, a margine su dodate za prvu i drugu kolonu sa desne strane tj. za one kolone za koje postoje rastojanje između kolona. CSS kod bi bio sledeći:

```
#col1 {
  float: left;
  width: 27%;
  margin: 0 3% 0 0;
}
#col2 {
  float: left;
  width: 47%;
  margin: 0 3% 0 0;
}
```

```
#col3 {
  float: left;
  width: 20%;
}
```

Ostatak koda bi ostao identičan. Dobijeni prikaz je dat na slici.



Slika 4.14. Prikaz web stranice sa 3 kolone

Prethodni dizajn se prilagođava samostalno veličini ekrana odnosno čitačima na kojima se prikazuje sadržaj. Međutim, u nekim graničnim slučajevima kada je širina ekrana jako velika ili jako mala, ovaj dizajn gubi na preglednosti, pa bi u tim slučajevima bolje bilo da bude fiksirana širina, uz cenu da se prikaže prazan prostor ili horizontalni skrol-bar.
Ovo se može izbeći postavljanjem ograničenja na maksimalnu, odnosno minimalnu širinu upotrebnom atributa `max-width`, odnosno `min-width`. Na taj način se mogu sačuvati neke prednosti fiksnog dizajna uz dodavanje fleksibilnosti u pogledu veličine medijuma za prikaz.

Primer. Tri kolone koje su apsolutno pozicionirane

Fluidni dizajn se ostvaruje i pomoću apsolutnog pozicioniranja, ali koristeći procente za određivanje dimenzija. U slučaju tri kolone, bez futera i navigacije, HTML, odnosno CSS kod, bio bi sledeći:

HTML:

```
<div id="container">
  <div id="header"> </div>
  <div id="content">
    <div id="col1"> </div>
    <div id="col2"> </div>
    <div id="col3"> </div>
  </div>
</div>
```

CSS:

```
#container {
  margin: 0 auto;
  width: 900px;
}
#content {
  position: relative;
  width: 900px;
}
#col1 {
  position: absolute;
  top: 0; left: 0;
  width: 25%;
}
```

```
#col2 {
  position: absolute;
  top: 0; left: 30%;
  width: 45%;
}
#col3 {
  position: absolute;
  top: 0; left: 80%;
  width: 20%;
}
```

Dobijeni izgled stranice je dat na slici.

Slika 4.15. Fluidni dizajn fiksnim pozicioniranjem

I u ovom dizajnu blok `#content` služi za pozicioniranje svih kolona. Pošto su kolone smeštene ispod `#header`-a, blok `#content` je postavljen iza hedera u izvornom kodu. Ako bi se relativno pozicionirali u odnosu na prozor čitača, inicijalni blok u kom bi bile sadržane, kolone bi mogle biti na pogrešnom mestu u zavisnosti od hedera, na primer ako bi se veličina teksta u elementu `h1` izmenila. Kolona `#col1` je zadata sa širinom od 25% i istovremeno je apsolutno pozicionirana na vrh do leve ivice sekcije `#content`. Kolona `#col2` je pozicionirana na 30% tako da ostavlja 5% razmaka od prethodne kolone. Slično se ponavlja i za narednu kolonu.
Prednosti i nedostaci fluidnog dizajna su:

- **Prednosti**:
  - Fluidni dizajn omogućava da se ostvari veza sa veličinom i rezolucijom čitača korisnika.
  - Izbegava se potencijalno nekontrolisani prazni prostor.
  - Na desktop čitačima, korisnik može kontrolisati širinu prozora i sadržaja.
  - Nema horizontalne skrol-barove.

- **Nedostaci**:

- Na velikim monitorima, dužine linija mogu biti teške za čitanje.
- Ovaj dizajn je delimično nepredvidiv. Elementi mogu biti raštrkani ili sabijeni u slučaju čitača ekstremnih dimenzija.
- Mnogo je teže postići željeni prazni prostor.
- Mnogo je više matematike uključeno u računanje veličina.

Elastični dizajn

Treći dizajnerski pristup sažima osobine promenljive veličine teksta sa predvidljivom dužinom linije. Elastični dizajn povećava ili smanjuje veličinu teksta.
Ako korisnik učini tekst većim, okvir koji sadrži taj tekst će se proširiti proporcionalno. Slično, ako korisnik učini tekst manjim, okvir za taj tekst će se takođe smanjiti. Rezultat je da dužina linije (broj karaktera po liniji) ostaje ista bez obzira na veličinu teksta. Ovo je značajna prednost u odnosu na fluidni i fiksni dizajn u kom dužine linije mogu postati veoma duge odnosno sa neočekivano malim brojem karaktera po liniji. Elastični dizajn podseća na opcije zumiranja na stranici kada se veličina prikaza zajedno menja sa veličinom fonta. Takođe, kada veličina teksta postane veća, veća je i površina koja tekst sadrži, tako prelomi linija ostaju na istim pozicijama.
Pri ovom dizajnu proporcije stranice su vezane za proporcije tipografskog sadržaja. Imajući u vidu trendove u pravcu sve većeg broja mobilne uređaje kao korisnika interneta, ovaj koncept ima sve veći značaj.

Kreiranje

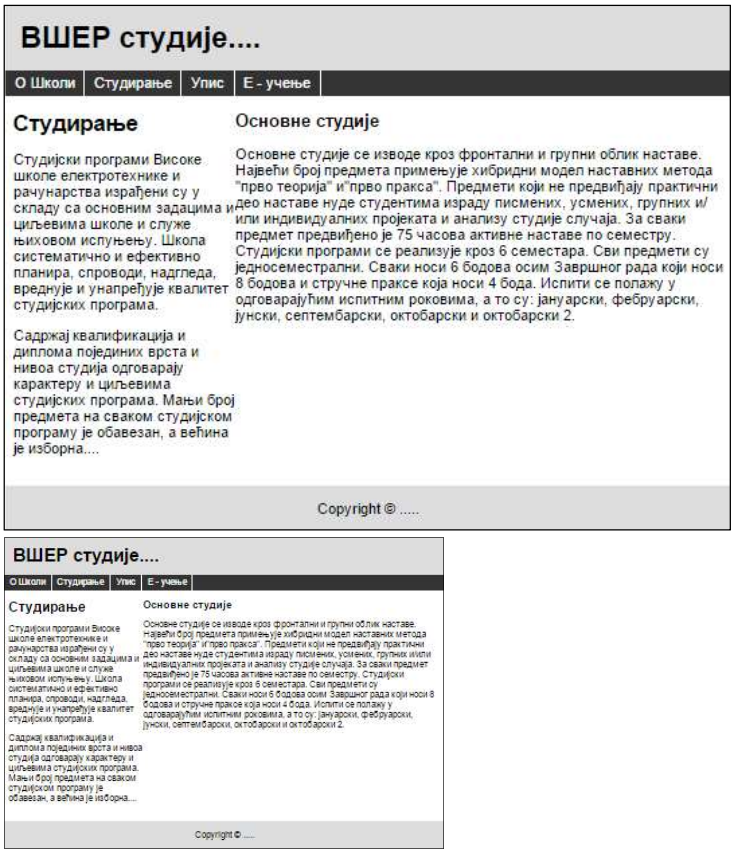
Ključ za elastični dizajn je jedinica mere `em`, kojom se definiše veličina teksta. U ovom dizajnu dimenzije elemenata koji su kontejneri tj sadrže druge elemente treba da budu u `em` jedinicama.
Na primer, ako je veličina teksta u sekciji `body` 16px, što je u većini slučajeva podrazumevano tj. 1em, a stranica je podešena na širinu od 50em, rezultujuća širina će biti 800 px. (50em x 16px/em). Ako

korisnik promeni veličinu teksta na 22 px, stranica se poveća na 1100px.
U primeru koji smo prethodno koristili izvršili smo izmene tako što smo procente zamenili em jedinicama. Dobijeni kod nije značajno izmenjen, a u zavisnosti od veličine fonta prikaz je proporcionalan.

```
#container{
  margin: 0 auto;
  width: 50em;
}
#navigation{
  float: left;
  width: 50em;
  background: #333;
}
#content{
  float: left;
  width: 50em;
  background: #FFF;
}
#col1{
  float: left;
  width: 15em;
  padding: 1em 0;
  margin:0 0 0 0,5em;
}
```

```
#col2{
  float: right;
  width: 33em;
  padding: 1em 0;
  margin:0 0.5em 0 0;
}
#footer{
  clear: left;
  background: #ddd;
  text-align: center;
  padding: 1em;
}
```

Dobijeni rezultat prikazan na narednoj slici i predstavlja elastični dizajn stranice u slučaju dve veličine fonta: 22px i 16px.



Slika 4.16. Elastični dizajn sa dve kolone

Prednosti i nedostaci elastičnog dizajna su:

- **Prednosti**
  - Omogućava konzistentan i očekivani prikaz uz omogućavanje fleksibilnosti veličine teksta.
  - Jača kontrola dužine linije teksta u odnosu na fluidne i fiksne prikaze.
- **Nedostaci**
  - Slika i video se ne skaliraju sa promenom veličine teksta.
  - U slučaju najvećih tekstova, veličina sadržaja može biti veća od veličine prozora čitača
  - Nije od velike koristi kada se koristi za različite uređaje tj. čitače različitih veličina.

- Teži je za kreiranje i održavanje od dizajna sa fiksnim prikazom.
- Sa promenom teksta se menja stranica, ali ne slike i filmovi.

Hibridni dizajn

Dizajn koji koristi kombinaciju piksela, procenata i em jedinica mere se po nekada naziva hibridni dizajn.
U mnogim scenarijima, ima smisla koristiti mešavinu fiksnih i skalabilnih sadržaja. Na primer, ako sa jedne strane treba postaviti kolonu fiksne širine (eng. sidebar) koja sadrži banere sa oglasima koji moraju ostati tačno određene veličine. Tada se može definisati da je ta kolona sa posebnom širinom iskazanom u px, a pri tome dozvoljavajući sledećoj koloni da se podešava na osnovu preostalog prostora.
U narednom primeru je prikazana realizacija prethodnog scenarija, tj. kada je prva kolona fiksne širine 200px u koju su smeštene slike širine 150px. Druga kolona će biti proširena do širine prozora.
U ovom primeru, ulogu druge kolone preuzima odeljak `#content` koji obuhvata i prvu kolonu i navigaciju. Pri tome prva kolona je sa svojstvom `float:left`, dok futer koristi `clear:left` kako bi došao ispod kolona.

HTML:

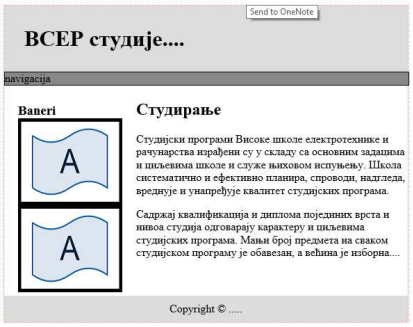
```
<body>
  <div id=”container”> . </div>
  <div id=”header”> . </div>
  <div id=”navigation”> . </div>
  <div id=”content”>
    <div id=”sidebar”> .</div>
    sadržaj druge kolone
    <div id=”footer”> .</div>
  </div>
</body>
```

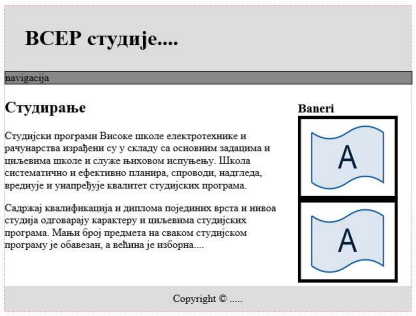
CSS:

```
#container{
  margin: 0 auto;
  width: 100%;
}
#header{
  background: #ddd;
  padding: 20px;
}
#content{
  width: 90%;
  padding: 5%;
  background: #ddd;
}
img{
  position: relative;
  left:20px;
}
```

```
#sidebar{
  float: left;
/*float: right;*/
  width: 190px;
  padding: 5px 0;
  background:lavender;
}
#footer{
  clear: both;
  background: #ddd;
  text-align: center;
  padding: 10px;
}
```

Dobijeni prikaz dat je na slici 4.17. u varijanti kada je kolona fiksne širine sa leve odnosno sa desne strane:





Slika 4.17. Hibridni prikaz

Pozadinska dekoracije

Dodavanje proizvoljne dekoracije u pozadinu kolona predstavlja efikasan način da se istakne podela u podacima uz istovremenu dekoraciju stranice.
U slučaju kada imamo više kolona na stranici, kolone nemaju istu visinu niti je visina unapred zadata, mada to vizuelno nije očigledno. U većini primera visina kolone je određena sadržajem u koloni. Ovo se može proveriti postavljanjem tankih ivica oko kolona. Futer dolazi ispod naviše kolone, dok sve ostale budu manje visine.
Ne postoji CSS podrška za postavljanje visine kolona elementa na 100% visine stranice. Takvo rešenje bi bilo moguće uz upotrebu JavaScript-a ili nekog dopunskog frejmvorka. Međutim, i bez toga, postoje rešenja koja mogu dati zadovoljavajuća rešenja u slučaju primene šablona sa fiksnim širinama.
Rešenje koje prezentujemo na ovom mestu je prvi primenio Dan Cederholm 2004. godine u knjizi *Web Standards Solutions.* Rešenje upravo razmatra problem nedefinisane visine kolone i samim tim nejasnoće dimenzije pozadinske slike. Zato bi postavljanje jako velike slike tj. slike koja bi pokrila pozadinu kolona i u slučaju kada su kolone jako visoke neefikasno, pošto se velika pozadinska slika dugo učitava i samim gubi na kvalitetu dizajna.
Međutim, ukoliko se ograničimo na pozadine koje ističu strukturu kolone, onda pozadina duž iste kolone može biti ista, pa umesto proizvoljne pozadinske slike može se postaviti slika veoma male visine, ali sa uključenim ponavljanjem po y osi. To je ključni koncept koji ćemo prikazati u narednim primerima.
Dakle, sliku koja sadrži boje po kolonama treba postaviti u kolone sa tačnom proporcijom širina. Visina slike nije bitna jer se replikacijom, tj. ponavljanjem te slike po vertikali dobija celokupna pozadinska slika. Zato je dovoljno da slika bude visine 1 piksel. U narednim primerima koristićemo sliku male visine, tek tolike visine da bi je mogli grafički da prikažemo.

Fiksni prikaz

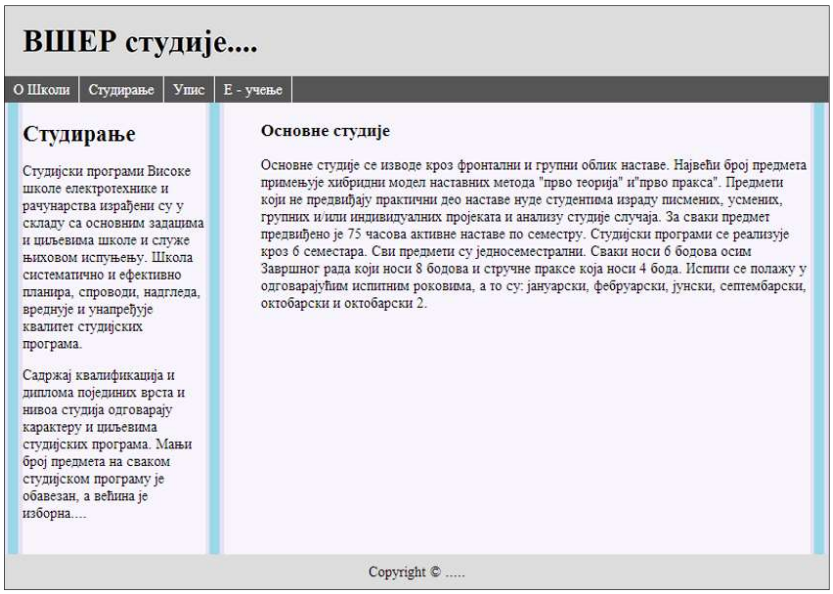
U slučaju fiksnog prikaza, kada su kolone na stranici sa fiksnom širinom, pozadinska slika će biti definisana dimenzijama kolona i margina u prikazu.
Koristićemo primer koji smo već ranije razmatrali, a u kome su širine margina i kolona redom: 20px, 200px, 20px, 640px, 20px, što ukupno čini 900px širine prozora. To znači da slika može imati proizvoljnu visinu, a širininu od 900px, sa segmentima dimenzija koje smo naveli. Sa ciljem da se istaknu ovi segmenti, a ne sa ciljem konačnog dizajna dat je primer takve slike.

Slika 4.18. Slika za dekoraciju pozadine - `fix2col.jpg`

Slika se ubacuje samo u odeljak koji sadrži kolone. U našem primeru to je `#content`, pa bi CSS kod bio sledeći:

```
#content{
  float: left;
  width: 900px;
  background-image: url(fix2kol.jpg);
  background-repeat: repeat-y;
}
```

A kao rezultat dobija se centralni deo stranice sa podlogom kao na slici 4.19.



Slika 4.19. Fiksni dizajn sa 2 kolone i pozadinskom slikom

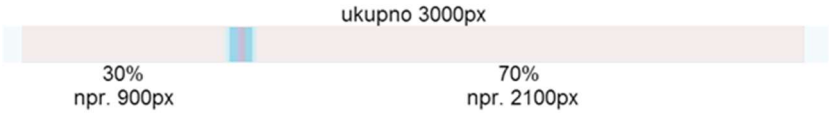
Pozadinska slika za fluidni dizajn

U fluidnom dizajnu nije unapred poznata širina kolona, ali su poznate proporcije između kolona. Imajući u vidu da se sa promenom širine prozora web čitača proporcionalno menjaju širine kolona može se postaviti pozadina čiji je izgled tako pripremljen da se sa promenom širine prilagođava kolonama.
Na primer, ako je naš dizajn web stranice sastavljen od dve fluidne kolone sa kolonama koje zauzimaju proporcionalno 30% odnosno 70%, tada bi slika koju sa ubacuje trebala da ima delove u istoj proporciji. Osnovni deo CSS koda kojim se definišu kolone bi bio:

```
#col1{
  float: left;
  width: 27.5%;
  margin: 0 2.5% 0 0;
}
#col2{
  float: right;
  width: 67.5%;
```

```
..margin: 0  0 0 2.5%;
}
```

U ovom slučaju iskoristiće se svojstvo `background-position`, pomoću koga se pozadinska slika postavlja tako da granica odgovara granici kolone. Ako je granica na 30% onda će biti `background-position:30%`. Takođe, slika bi trebala da bude dovoljno široka da obuhvati sve slučajeve krajnjih korisnika tj. onih sa najširim monitorima. I u ovom slučaju visina slike nije od značaja, pa se može odabrati visina od svega 1 piksel.
Na primer, ako je naša slika širine 3000px, onda je širina prve kolone 30% x 3000 = 900px, a druge 2100px. Na slici smo sredinu granicu kolone posebno istakli.

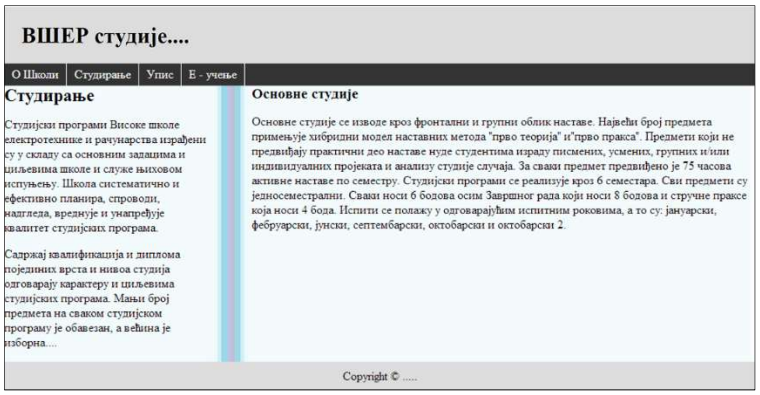


Slika 4.20. Slika za dekoraciju pozadine - `fluid2col.jpg`

Podsetimo se: kada se koristi procenat za pozicioniranje slike, onda je on relativan. Na primer, ako je prvi podatak 0%, onda je 0% od slike na 0% od prozora tj. slika je u gornjem levom uglu prozora. Ako je 30% onda je 30% od slike na 30% od prozora, što je slučaj koji odgovara primeru. Druga vrednost se odnosi na donji desni ugao ako nema ponavljanja, što ovde nije slučaj. Na osnovu ovoga sledi da je CSS kod za odeljak koji sadrži kolone:

```
#content{
  float: left;
  width: 100%;
  background-image: url(fluid2col.jpg);
  background-repeat: repeat-y;
  background-position: 30%;
}
```

Dobijena web stranica ima izgled kao na slici. Vidi se da je granica kolona istaknuta bez obzira na stvarnu širinu prozora čitača.



Slika 4.21. Fluidni dizajn sa 2 kolone i pozadinskom slikom

Pitanja za proveru znanja

1. Koje vrste dizajna postoje?
2. Navedite osnovne karakteristike fiksnog dizajna.
3. U sledećem CSS kodu započet je dizajn stranice sa dve kolone.

   Dovršite ga.

```
#container{
  width:960px;
  margin:0 auto;
  position: relative;
}
#col1 {
  position: absolute;
  left: 0px;
  width: 220px;
}
```

Data je HTML struktura sa dve kolone, futerom i pripadajući CSS kod za fiksni dizajn:

```
<div id=”container”>
    <div id=”content”>
      <div id=”col1”> .</div>
      <div id=”col2”> .</div>
      <div id=”footer”> .</div>
    </div>
</div>
```

CSS:

```
#content{
  float: left;
  width: 900px;
  background: #fff;
}
#col1{
  float: left;
  width: 200px;
  padding: 20px 0;
  margin: 0 0 0 20px;
}
```

Dopuniti CSS kod sa dizajnom za preostalu kolonu i futer.

4. Koje su osnovne karakteristike fluidnog dizajna?
5. Dat je fluidni dizajn stranice sa tri kolone:

HTML:

```
<div id="container">
  <div id="header"> </div>
  <div id="content">
    <div id="col1"> </div>
    <div id="col2"> </div>
    <div id="col3"> </div>
  </div>
</div>
```

CSS:

```
#container {
  margin: 0 auto;
  width: 960px;
}
#content {
  position: relative;
  width: 960px;
}
#col1 {
  position: absolute;
  top: 0; left: 0;
  width: 20%;
}
```

Dodati CSS kod za preostale dve kolone proizvoljne širine.

6. Koje su osnovne karakteristike elastičnog dizajna?
7. Napisati jednu stranicu u stilu elastičnog dizajna sa tri kolone i zaglavljem.

Rezime

	PREDNOST	NEDOSTATAK
FIKSNI	Predvidivost. Kontrola dužine linije. Lak za upotrebu. Dovoljno za desktop.	Sadržaj uz desnu ivicu može biti sakriven. Može se pojaviti višak prostora sa leve strane. Dužine linija teksta mogu prevelike. Korisnik nema kontrolu prikaza.
FLUIDNI	Veza sa veličinom i rezolucijom čitača.	Dužine linija mogu biti teške za čitanje.

	Izbegava se nekontrolisani prazni prostor. Na desktop-u korisnik kontroliše širinu prozora. Nema horizontalne skrol-barove.	Delimično nepredvidiv. Elementi mogu biti raštrkani ili sabijeni. Teže se dobija željeni prazni prostor. Mnogo je više matematike uključeno u računanje.
ELASTIČNI	Omogućava konzistentan i očekivani prikaz. Jača kontrola dužine linije teksta u odnosu na fluidne i fiksne prikaze.	Slika i video se ne skaliraju. Veličina sadržaja može biti veća od prozora čitača Nije od velike koristi kada se koristi za različite uređaje. Teži je za kreiranje i održavanje od dizajna sa fiksnim prikazom. Sa promenom teksta se ne menja veličina slika i filmova.

Deo 5: Odabrane CSS tehnike

- CSS reset
- CSS sprajtovi
- Stilizovanje formi

U ovom poglavlju opisujemo nekoliko praktičnih CSS tehnika koje se primenjuju pri dizajnu web stranica. Obrađene su teme: reset tj. poništavanje već postojećeg CSS-a, upotreba slika umesto teksta i dizajn formi.

CSS reset

U poglavlju posvećenom redosledu primene stilova navedeno je da svi web čitači imaju sopstvene tzv. ugrađene stilove ili predefinisane stilove koji se koriste za prikaz HTML, elemenata. Ovi stilovi imaju niži prioritet od onih koji se nalaze u izvornom kodu i definišu prikaz elemenata kada sopstveni stilovi nisu navedeni. Tako na primer, element **h1** ima ugrađeni stil koji tekst prikazuje naglašeno (bold), odvojeno od susednih elemenata i većeg fonta u odnosu na normalni tekst. Stvarna veličina fonta i prostor koji odvaja **h1** je definisan ugrađenim stilovima čitača. Dakle, različiti ugrađeni stilovi web čitača mogu dati različit prikaz za isti HTML odnosno CSS kod.
Uzimajući u obzir prethodnu činjenicu i to da HTML elementi nasleđuju određene stilove od stilova koji su ugrađeni, moguće je da

se dogodi „neočekivani“ prikaz u nekim čitačima. Ovaj problem dizajneri rešavaju primenom tzv. „CSS reseta“ kojim poništavaju ugrađene stilove, odnosno dobijaju potpunu kontrolu u dizajnu.
CSS reset je kolekcija stilova koji preklapaju ugrađene stilove web čitača tako kreirajući početni, maksimalno nezavisni stil. Najpopularniji reset je napisao Eric Meyer:
http://meyerweb.com/eric/tools/css/reset.
Taj kod dajemo u nastavku. Zapazite da CSS reset zahteva da se eksplicitno specificira font, text, margina i peding za svaki element u dokumentu.

```
/*http://meyerweb.com/eric/tools/css/reset/ v2.0 | 20110126
License: none (public domain) */
html, body, div, span, applet, object, iframe, h1, h2, h3, h4,
h5, h6, p, blockquote, pre, a, abbr, acronym, address, big,
cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd,
ol, ul, li, fieldset, form, label, legend, table, caption,
tbody, tfoot, thead, tr, th, td, article, aside, canvas,
details, embed, figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary, time, mark, audio,
video {
      margin: 0;
      padding: 0;
      border: 0;
      font-size: 100%;
      font: inherit;
      vertical-align: baseline; }


/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure, footer, header,
hgroup, menu, nav, section {
      display: block;
}
body {
      line-height: 1;
}
```

```
ol, ul {
      list-style: none;
}
blockquote, q {
      quotes: none;
}
blockquote:before, blockquote:after, q:before, q:after {
      content: ''; content: none;
}
table {
      border-collapse: collapse;
      border-spacing: 0;
}
```

Da bi se primenio reset, navedeni kod sa stilovima treba se postavi na vrh sopstvenih stilova. Više o ovome možete pročitati na: *http://meyerweb.com/eric/thoughts/2007/04/18/reset-reasoning/*.

Zamena teksta slikama

Pre nego što su web fontovi postali upotrebljivi na način kako danas funkcionišu, slike su se koristile svaki put kada je dizajner želeo ili morao da koristi specijalne fontove. Zamena teksta slikama danas ima drugu primenu. Na primer, ako se u dizajnu koristi stilizovani logo neke kompanije ili određene ikone umesto teksta, neophodno je ubaciti slike umesto dela teksta.
Razmotrimo jednostavan postupak. Slika se ubacuje umesto teksta, tako što se izvrši izbacivanje dela teksta, a na isto mesto se postavi slika. Ovakav postupak uključuje i gubitak dela sadržaja, što nije dobro rešenje. Sadržaj treba sačuvati jer se tako omogućava pretraživačima pretraga po sadržaju, ali i kasnije lakše održavanja sadržaja. Zato je mnogo bolje rešenje da se, umesto izbacivanja teksta, upotrebom CSS-a, izvrši vizuelna zamena teksta sa slikom. Na taj način se u vizuelnoj reprezentaciji pojavljuju slike umesto teksta, a istovremeno se čuva tekstualni sadržaj.

Jedno dobro rešenje (koje je prvi prezentovao Scott Kellum), zasniva se na korišćenju svojstva `text-indent` kojim se tekst pomera u stranu van vidljivog pravougaonika. Pogledajmo konkretan primer. Neka je potrebno umesto naziva kompanije „123netdoo“, koji se nalazi u naslovu web stranice, postaviti logo te kompanije, i neka je početni HTML odnosno CSS kod:

HTML

```
<h1 id="logo">123netdoo</h1>
```

CSS

```
h1#logo {
  width: 200px;
  height: 50px
}
```

Naslov je definisan elementom `h1`, prikazuje se na podrazumevani način, primenom ugrađenih stilova, uz definisanu širinu i visinu. Dimenzija naslova odgovara dimenzijama slike koja će naknadno biti postavljena u pozadini. U primeru to je: 200x50 piksela.
Da bi se sadržaj naslova pomerio iz vidljivog okvira, dodaje se stilom `text-indent` , a istovremeno se postavlja pozadinska slika primenom `background` svojstva, tako da se naslov minimalno pomeri, a istovremeno pojavi pozadinska slika u naslovu. Prošireni kod je:

CSS kod za zamenu teksta slikom:

```
h1#logo {
  width: 200px;
  height: 50px
  background: url(123netdoo.png) no-repeat;
  text-indent: 100%;
  white-space: no-wrap;
  overflow: hidden;
}
```

Svojstvom `text-indent` pomera se reč „123netdoo” udesno za čitavu širinu bloka koji se koristi, što je u kodu označeno sa 100%, tako da će reč biti izvan vidljivog pravougaonika. Takođe, postavljanjem svojstva `white-space` na vrednost `no-wrap` osiguravamo da u

slučaju dužeg stringa tekst neće biti prelomljen i na taj način postati vidljiv. Na kraju, postavkom `overflow: hidden` daje se instrukcija čitaču da bilo šta što će se naći izvan okvira elementa, u ovom slučaju je to samo tekst u `h1`, neće biti prikazan.

Slika 5.1. Ilustracija zamene teksta slikom

Osim pomenute CSS tehnike, mogu se naći u primeni i druge. Jedna od najpopularnijih je tehnika Phark-a koji koristi jako velike negativne vrednosti za `text-indent`, tipično: -9999px, kako bi „izgurao“ HTML tekst izvan leve ivice vidljive oblasti.

```
h1#logo {
  width: 200px;
  height: 50px
  background: url(123netdoo.png) no-repeat;
  text-indent: -9999px;
}
```

Nedostatak ove tehnike je što se čitači forsiraju da računaju, a neki čak iscrtavaju velike elemente iako oni neće biti uključeni u prikaz, pa se na taj način gubi na efikasnosti.
Uključivanje slika u web stranicu znači dodatne konekcije ka serveru i dodatno usporavanje učitavanja stranice. Ovo je naročito vidljivo u slučaju kada se koristi veći broj slika. Zato se zamena teksta slikama izvodi samo kada je to neophodno.

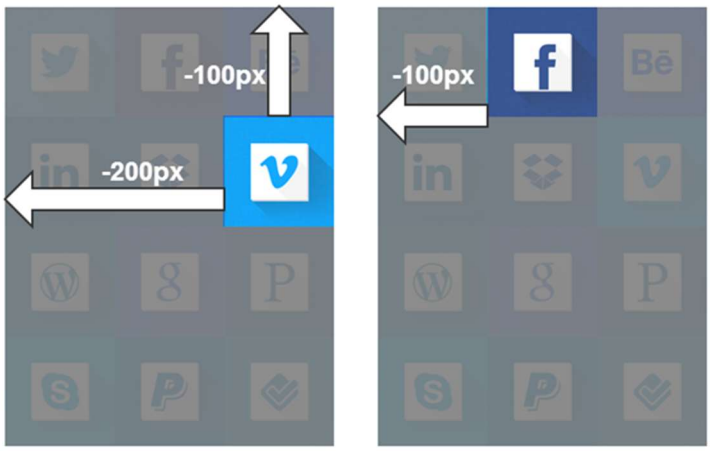
CSS sprajtovi

Vreme pristupa nekoj web stranici zavisi od količine podataka koje je potrebno učitati da bi se stranica prikazala. Osim količine podataka na

vreme učitavanja utiče i broj zahteva ka serveru koji prati sadržaj. Jedan od načina da se smanji broj zahteva ka serveru je da se smanji broj slika na stranici, imajući u vidu da svaka slika znači posebnu vezu ka serveru.
U slučaju HTML stranica koje koriste puno manjih slika, može se umesto više manjih slika koristiti jedna velika zajednička slika, koja će sadržati sve manje slike, koje tada nazivamo **sprajtovima**. Pomoću pozicioniranja pozadinske slike elementa, tj. koristeći svojstvo `background-position`, iz velike slike se može izdvojiti svaki sprajt posebno. Na ovaj način se izbegava otvaranje više konekcija zbog više manjih, a time se uvećava efikasnost.
Na primer, ako se na sajtu koriste sličice za predstavljanje dvanaest često korišćenih linkova, umesto dvanaest posebnih slika moguće je koristiti jednu, kao na slici 5.2.

Slika 5.2. Slika sa sprajtovima

Slika sadrži 4 kolone i 3 reda dimenzije 300 x 400 piksela, tako da je svaki sprajt dimenzije 100 x 100 piksela.
Izdvajanje određenog sprajta iz slike se izvodi pozicioniranjem pozadinske slike u HTML elementu, imajući u vidu da se x koordinata za pozicioniranje proteže s leva na desno, počev od vrednosti 0, a y koordinata odozgo na dole, takođe od vrednosti 0.



Slika 5.3. Pozicioniranje prozora za izdvajanje sprajtova

Na slici 5.3. ilustrovan je primer pozicioniranja slike tako da se izdvoje određeni sprajtovi. Ako HTML element treba da prikaže u pozadini sprajt koji se nalazi na trećoj poziciji u drugom redu, tj. njegov gornji levi ugao u odnosu na gornji levi ugao slike je (200, 100), potrebno je izvršiti pomeranje pozadinske slike u levo i na gore tako da taj sprajt bude u koordinatnom početku, dakle za 200px u levo i za 100px na gore, što se definiše pozicijom -200px, -100px.
U sledećem primeru dat je HTML kod kojim se iz prikazane slike izdvajaju sprajtovi i postavljaju u jednu listu linkova.

HTML:

```
<li><a href=http://twitter.com
     class="sprites1 twitter">Twitter</a></li>
<li><a href=http://facebook.com
     class="sprites1 fb">Facebook</a></li>
<li><a href=https://www.behance.net
     class="sprites1 be">Behance</a></li>
<li><a href=https://www.linkedin.com
     class="sprites1 linkedin">LinkedIn</a></li>
<li><a href=https://vimeo.com
     class="sprites1 vimeo">Vimeo</a></li>
<li><a href=http://google.com
     class="sprites1 google">Google</a></li>
<li><a href=http://www.skype.com
```

```
    class="sprites1 skype">Skype</a></li>
```

Svakoj stavci u listi je pridružen element `a`, dimenzije 100x100 piksela koji dimenzijom odgovara jednom sprajtu. Takođe, svaki link unutar stavke ima istu pozadinsku sliku `ikone.png`, koja sadrži sve sprajtove, što je definisano CSS kodom. Na ovaj način, samo je jedna slika učitana pri učitavanju stranice tj. samo je jedna dodatna konekcija otvorena.

```
sprites1 {
  text-indent: 100%;
  white-space: nowrap;
  overflow: hidden;
}
li a {
  display: block;
  width: 100px;
  height: 100px;
  background-image: url(ikone.png);
}
```

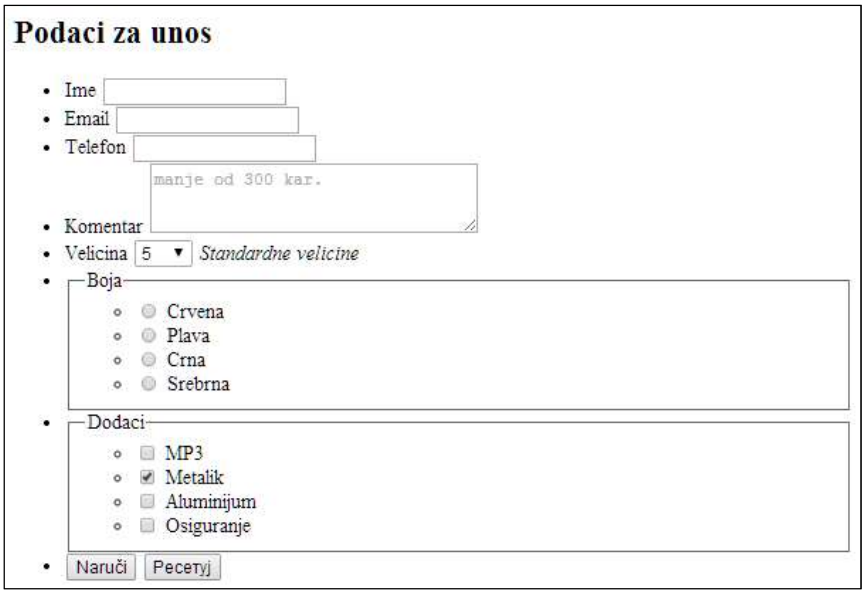
Kodom je još definisano da se tekst u listi ne prikazuje čak i u slučaju dugih linija sa belinama. Ista slika je postavljena uz svaku stavku, ali sa različitom pozicijom.
U narednom kodu je dat primer izdvajanja nekoliko različitih slika sa pratećim linkovima.

```
li a.twitter {
  background-position: 0 0;
}
li a.fb {
  background-position: -100px 0;
}
li a.be {
  background-position: -200px 0;
}
li a.linkedin {
  background-position: 0px -100px;
}
li a.vimeo {
  background-position: -200px -100px;
}
li a.google {
  background-position: -100px -200px;
}
li a.skype {
  background-position: 0 -300px;
}
```

Slika 5.4. Izdvajanje sprajtova i prikaz u listi

Stilizovanje formi

Inicijalni izgled web formi, zasnovan na podrazumevanim stilovima, je takav da se obavezno dodatno stilizuje. Razlog je pre svega lakši i brži unos podataka, ali i postizanje željenog vizuelnog izgleda. U nastavku je dat primer jedne forme sa poljima za unos podataka bez dodatne stilizacije. U primeru su korišćeni gotovo svi tipovi elemenata za unos smešteni u jednu listu.



Slika 5.5. Primer jedne nestilizovane forme

Generalno, za stilizovanje formi ne postoje specijalna svojstva, potrebno je koristiti standardne boje, pozadinu, font, ivice i sl.

Osnovna stilizacija kontrola

Najpre treba grupisati kontrole na osnovu zajedničkih svojstava koje ćemo koristiti pri stilizaciji.

- Unos tekstualnih podataka (*text, password, email, search, tel, url*)

  Najčešće se podešavaju dimenzije, pozadinska svojstva, ivice i margine. Može se stilizovati i unutrašnji tekst sa bojom i fontovima.
- Element *textarea*

  Za ovaj element se vrše podešavanja kao i za prethodnu grupu, ali sa dodacima. Na primer, ugrađeno pravilo za `textarea` znači podrazumevanu primenu *„monospace"* fonta, pa stilizacija obično uključuje promenu fonta. Pošto kontrola sadrži više linija teksta, obično se definiše i visina linije.

Takođe, neki čitači prikazuju u donjem desnom uglu grafičku oznaku koja ukazuje da se može menjati veličina, što se može poništiti dodavanjem stila `resize:none;`.

- Ulazna Dugmad (*submit, reset, button*)

  Dugmad su podrazumevano podešena tako da se veličina kontrole prilagođava minimalnoj veličini teksta koji se prikazuje. Neki čitači dodaju izvesni prostor - peding. Svojstva koja se najčešće koriste za stilizaciju su: `width, height, border`.
- Radio, checkbox dugmad

  Dobra praksa je da se ovi elementi ne menjaju.
- Padajuća lista *drop-down*

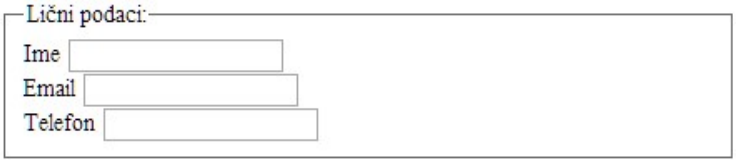
  Obično treba podesiti širinu i visinu ovih elemenata. Neki čitači daju mogućnost da se upravlja bojom, pozadinskom bojom, odnosno fontom, ali je jedno od najboljih rešenja ne dirati ta svojstva tj. ostaviti ih pod kontrolom čitača odnosno operativnog sistema.

Grupisanje elemenata u formi

Za grupisanje elemenata u formi koristi se element `<fieldset>`. Primenom ovog elementa iscrtava se pravougaonik oko obuhvaćenih elemenata. Pravougaonik može imati i tekst kojim se dodatno opisuje grupa. Dodatni tekst označava se posebnim elementom `<legend>`.

`Grupisanje kontrola u formi:`

```
<fieldset>
  <legend>Lični podaci:</legend>
  Ime <input type="text"> <br>
  Email <input type="text" > <br>
  Telefon <input type="text"> <br>
</fieldset>
```



Slika 5.6. Grupisanje kontrola primenom elementa `fieldset`

Element `fieldset` u novoj HTML5 specifikaciji ima atribute:

- `disabled` – označava da je unos podataka za celu grupu elemenata onemogućen. Npr. `<fieldset disabled>...</fieldset>`.
- `form` – označava formu kojoj pripada grupa kontrola. To znači da se van forme može definisati grupa koja bi pripadala formi.
- `name` – ime grupe.

Labele uz elemente

Element `<label>` definiše labelu za neki ulaz forme tj. za neki `<input>` element. Element `<label>` ne nudi nikakva posebna svojstva korisniku, osim što omogućava kontrolu u slučaju klika mišem. Bez ovog elementa, kontrola za unos podataka dobija fokus kada se mišem klikne baš na nju. Međutim, u slučaju kada su te kontrole vizuelno male, na primer dugme za potvrdu, korisniku može biti naporno da mišem klikne na tu kontrolu. Zbog toga je bilo važno olakšati izbor malih kontrole. Ako se uz kontrolu za unos podataka nalazi tekst tj. labela kreirana pomoću HTML elementa `label` moguće je obezbediti da se korisničkim klikom na labelu dogodi isto što i klikom na kontrolu.
Element `label` ima atribut `for` kojim se određuje element na koji se labela odnosi. Vrednost ovog atributa treba da odgovara vrednosti atributa `id` ciljanog elementa. Takođe se povezivanje može obaviti i tako što se ciljani element smesti unutar elementa `label`. Na ovaj način se labela i ulazni element povezuju u jednu celinu, tj. klikom na labelu događa se isto kao i klikom na taj element. Na primer:

```
< label for="ime">Ime</label>

<input type="text" id="ime" class="textinput">
```

Ili

```
<label> <input type="text"> </label>
```

Jedan postupak stilizacije formi

Podešavanje izgleda forme se obično izvodi u nekoliko koraka, počev od osnovnih podešavanja ka specifičnim.

- Osnovna podešavanja definišu stilove koji se odnose na podešavanje elemenata celokupne stanice, naslova i listi. Takođe, ovaj skup pravila treba da sadrži pravila za `form` element, kao što su širina, pozadinska boja, peding i slično. U našem primeru je dodato svojstvo ograničenog prikaza `overflow:hidden`, kao i isključivanje plutajućeg rasporeda u stavkama liste.

```
ul li {
  clear: both;
  list-style:none;
}
form {
  width: 35em;
  border: 1px solid #333;
  background-color: lightblue;
  padding: 1em;
  overflow: hidden;
}
```

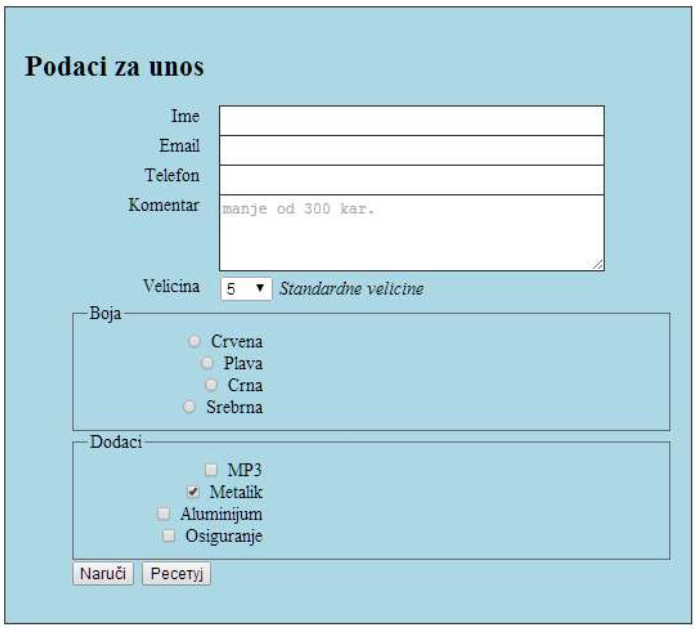
Slika 5.7. Izgled forme nakon osnovnih podešavanja

- Sledeći korak bi bilo poravnavanje. Da bi izvršili poravnavanje kontrola potrebno je da labele budu tačno određene širine, ali i da budu sa statusom plutajućih elemenata uz levu ivicu, dok njihov sadržaj treba da bude poravnat uz desnu ivicu, tako da budu blizu ulaznih kontrola na koje se odnose. Naravno, treba dodati i malu marginu koja doprinosi osećaju razdvajanja po kolonama.

```
label {
  float: left;
  width: 7em;
  text-align: right;
  margin-right: 1em;
}
```

Slika 5.8. Izgled forme nakon poravnanja

- Očekivano je i da sve kontrole za unos teksta budu poravnate, za to je potrebno definisati određenu širinu kontrola.



Slika 5.9. Izgled forme nakon podešavanja širine kontrola

Kontrola `textarea` se može dodatno podešavati svojstvima `margine`, `overflow` i `resize.`

- U nastavku ćemo podesiti stilove za grupisane elemente unutar kontrole `fieldset`, kao i za samu legendu koja se tiče ove kontrole. Umesto okvira sa naslovom videćemo da se grupa može organizovati na potpuno drugačiji vizuelni način. Postavićemo tekst koji je u elementu `legend` u vizuelno istu liniju kao i ostale labele, a opcije jednu do druge, ne jednu ispod druge kako je podrazumevano. Krenimo redom.
  Prvo ćemo isključiti okvir oko grupa.

```
fieldset {
  margin: 0;
  padding: 0;
  border: none;
}
```

Zatim, vršimo formatiranje naziva grupa na način da bude vizuelno isti kao i formatirane labele.

```
legend {
  width: 7em;
  float: left;
  margin-right: 1em;
  text-align: right;
  color: darkblue;
}
```

Stavke u prvoj listi postavljamo u jednoj liniji, a u drugoj listi jednu ispod druge poravnate uz levu ivicu.

```
#colors label, #dodaci
label {
  float: none;
  width: auto;}
#colors ul li{
  display: inline;
  margin-bottom: 0;}
```

```
#dodaci ul {
  margin-left: 6em;
}
#dodaci ul li {
  margin-bottom: 0;
  clear: none;
}
```

Ova podešavanja daju izgled kao na slici:

Slika 5.10. Nakon podešavanja grupisanih elemenata

- Na kraju sledi podešavanje dugmadi u formi: poravnavanje, podešavanje veličine, boje, margine…

```
input[type="submit"], input[type="reset"] {
  display: block;
  width: 7em;
  height: 2em;
  float: left;
  background: white;
  border: 1px solid black;
  margin: 1em;
}
input[type="submit"] {
  margin-left: 8em;
  color: darkblue;
}
```

Što daje sledeći izgled forme:

Slika 5.11. Izgled forme nakon podešavanja dugmadi

Navigacioni meni

Navigacioni meni predstavlja sastavni deo svih sajtova. U tehničkom smislu meni je skup linkova organizovanih u liste koji služe za odabir

određenih stranica odnosno pozicioniranje unutar njih. Vizuelni prikaz linkova u meniju je modifikovan i prilagođen ovoj nameni. Na primer, neka je osnovna lista:

```
<ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="novosti.html">Novosti</a></li>
    <li><a href="izdvajamo.html">Izdvajamo</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
</ul>
```

Za linkove treba isključiti podvlačenje, bojenje ali i dodatne margine i padding:

```
ul{
    list-style-type: none;
    margin: 0;
    padding: 0;
}
```

Dalje formatiranje zavisi od samog dizajna i vrste menija. Po načinu prikaza mogu se podeliti na horizontalne i vertikalne.

Vertikalni meni

Vertikalni meni sadrži stavke koje su pozicionirane po vertikali, obično uz levu stranu prikaza. Zato je potrebno da sve stavke budu vertikalnog menija budu iste širine, da mogu da reaguju na korisnički click kao i da mogu da se formatiraju kao celine.
Stavke u vertikalnom meniju se definišu kao blok elementi. Širina se definiše u samom linku tj. elementu `a` ili na celoj listi tj. elementu `ul`. Formatiranje stavki liste se obavlja na `a` elementima, a ne na `li` elementima zbog toga jer se inače vrši neophodno ukidanje dekoracija na `a` elementima pa se mogu primeniti i dodatna formatiranja.

```
ul{
    list-style-type: none;
    margin: 0;
    padding: 0;
```

```
    width: 100px;
}

li a{
    display: block;
    padding:5px 2px 5px 10px;
    background-color:lightgray;
    text-decoration:none;
}
```

Pri dizajnu menija, često se primenjuje efekat promene izgleda stavke kada se miš nađe iznad te stavke naglašavajući nameru korisnika. Jedna varijanta bi bila promena pozadinske boje `a` elementa.

```
li a:hover {
    background-color: #555;
    color: white;
}
```

Drugo, meniji često pokazuju stranica koja je trenutno aktivna. Na ovaj način korisniku se daje podatak i o trenutnoj poziciji tokom navigacije i mogućnost da se lakše snalazi u pozicioniranju. Jedan način je formiranje posebne klase. Na primer:

```
.active {
    background-color: lightblue;
}

<ul>
    <li><a class="active"
href="navigationBar.html">Home</a></li>
...
```

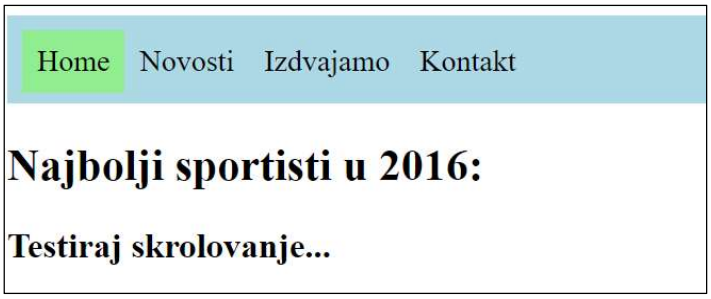
Horizontalni meni

Horizontalni meni smo pominjali u prethodnim poglavljima. Ovde navodimo ukratko rezime i kod.
Stavke liste treba da promene svojstvo prikaza u `inline` ili se, koristeći svojstvo `float`, prebace u jednu liniju. Tako za HTML kod:

```
<nav>
  <ul>
    <li><a href="navigationBar.html">Home</a></li>
    <li><a class="active" href="novosti.html">Novosti</a></li>
    <li><a href="izdvajamo.html">Izdvajamo</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
  </ul>
</nav>
<article>
  <h2>Najbolji sportisti u 2016:</h2>
  <h3>Testiraj skrolovanje...</h3>
  <p>Lorem  . . .
```

odgovarajući CSS kod za horizontalni meni bio bi:

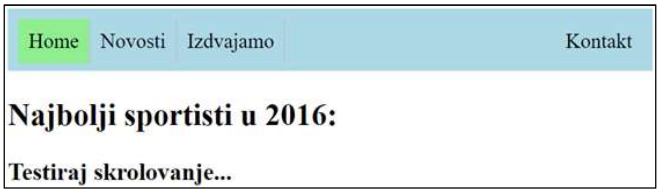
```
article {
    clear:both;
    width: 100%;
    height:100%;
}
nav {
    height:3em;
    width: 100%;
    background-color:
                lightblue;
}
nav ul{
    list-style-type: none;
    margin: 0;
    padding: 0.5em;
}
nav li{
    float:left;
}
nav li a{
    display: block;
    padding:0.5em;
    text-decoration:none;
    color:black;
}
nav li a:hover:not(.active) {
    background-color: #555;
    color: white;
}
nav .active {
    background-color:
                lightgreen;
}
```



Slika 5.12. Primer horizontalnog menija

Nekada je važno da se jedna stavka horizontalnog menija izdvoji i postavi uz desnu ivicu. Obično je razlog za to što se očekuje da na desnoj poziciji bude određena stavka, na primer Kontakt. U nastavku je prikazan CSS kod sa klasom koji poslednju stavku liste pozicionira udesno i dodaje separator između stavki.

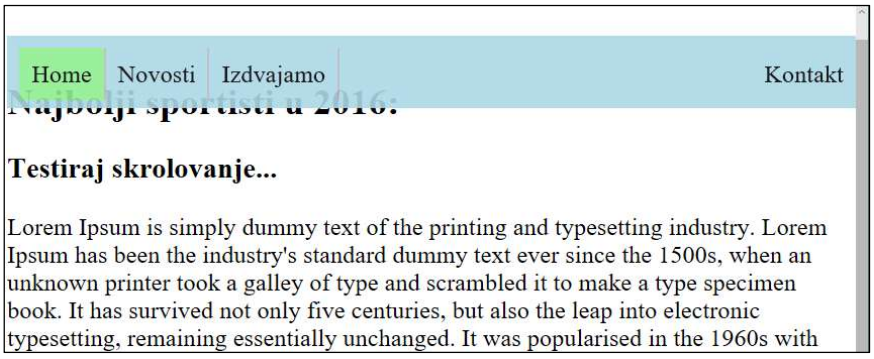
```
nav li{
    float:left;
    border-right: 1px solid #bbb;
}
nav .udesno{
    float:right;
}
li:last-child {
    border-right: none;
}
```



Slika 5.13. Primer horizontalnog menija sa izdvojenom stavkom

Horizontalni navigacioni meni je pozicioniran na vrhu stranice. Pri vertikalnom skrolovanju sadržaja meni se pomera van vidljivog prozora. Nekada je potrebno da se pri skrolovanju pozicija menija ne menja.

```
nav {
  height:3em;
  width: 100%;
  background-color: lightblue;
  position: fixed; /*!!!*/
  opacity:0.9;
  z-index:999; /*!!!*/
}
nav ul {
  list-style-type: none;
  margin: 0;
  padding: 0.5em;
}
nav li{
  float:left;
  border-right: 1px solid #bbb;
}
nav .udesno{
  float:right;
}
li:last-child {
  border-right: none;
}
nav li a{
  display: block;
  padding:0.5em;
  text-decoration:none;
  color:black;
}
nav li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
nav .active {
  background-color: lightgreen;
}
article {
  clear:both;
  width: 100%;
  position:relative; /*!!!*/
  top:3em;
}
```



Slika 5.14. Horizontalni meni fiksne pozicije

Padajući podmeni

Definisane stavke menija predstavljaju linkove koji vode do novih stranica i/ili pozicija. Međutim, u praksi se najčešće iza osnovnih stavki menija nalazi padajuća lista sa podstavkama. Dakle, vrši se kombinacija horizontalnog i vertikalnog menija.
Stavke padajućih menija grupišu se po odeljcima. Na vrhu svake grupe nalazi se dugme. Ispod dugmeta je postavljen odgovarajući vertikalni meni, opet u posebnom odeljku. Aktiviranje padajućeg menija se vrši ako je miš iznad dugmeta.
Odeljak podmenija mora imati **apsolutno pozicioniranje** kako pri njegovom prikazivanju ne bi došlo do uticaja na prikaz ostalih elemenata. Takođe bi z-index trebalo da bude postavljen na veću vrednost kako bi se prikaz podmenija ostvario posle svih ostalih elemenata tj. bio bi prikazan na vrhu. Jedan primer HTML koda sa pomenutim elementima i primenjenim klasama dat je u nastavku:

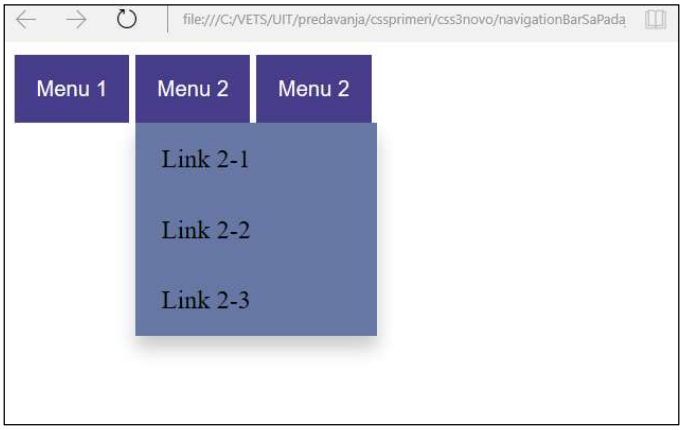
```
<nav>
    <div class=" padajucaLista">
        <button class=" prvaStavka">Menu 1</button>
        <div class=" padajucaLista-sadrzaj">
            <a href="#">Link 1-1</a>
            <a href="#">Link 1-2</a>
            <a href="#">Link 1-3</a>
        </div>
    </div>
    <div class="dropdown">
        <button class="dropbtn">Menu 2</button>
        <div ***
    <div class="dropdown">
        ****
```

Odgovarajući CSS stilovi bi bili:

```
.padajucaLista {
  position: relative;
  display: inline-block;
}
```

```
  box-shadow: 0px 8px 16px 0px
                rgba(0,0,0,0.2);
}
.padajucaLista-sadrzaj a {
```

```
.prvaStavka {
  background-color:
darkslateblue;
  color: white;
  padding: 1em;
  border: none;
  cursor: pointer;
  /*margin:0px -3px-3px-3px;*/
  /*postoji ugradjena margina*/
}
.padajucaLista-sadrzaj {
  display: none;
  position: absolute;
  /*pokušajte da isključite*/
  background-color:
      rgba(86, 105, 154, 0.9);
  min-width: 150px;
  color: black;
  padding: 0.8em 1em;
  text-decoration: none;
  display: block;
}
.padajucaLista-sadrzaj a:hover{
  background-color: lightgray
}
.padajucaLista:hover
.padajucaLista-sadrzaj {
  display: block;
}
.padajucaLista:hover
.padajucaLista {
  background-color: lightblue;
}
```

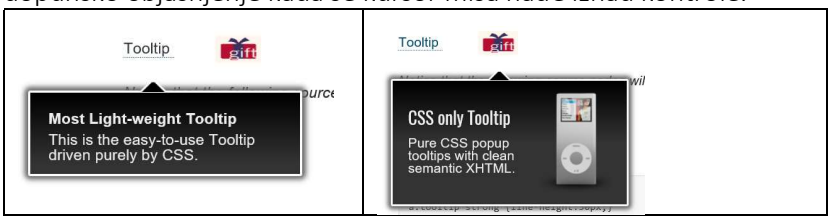


Slika 5.15. Padajući meniji u horizontalnom meniju

Napomena. Na kraju, treba naglasiti da dizajneri često koriste gotove šablone menija koje mogu naći na Internetu. Možete pogledati stranicu http://cssmenumaker.com/ i testirati neke vrste menija.

Realizacija kratkih iskačućih poruka (eng. tooltip)

Kontrola tipa *tooltip* predstavlja poruku koja se pojavljuje kao dopunsko objašnjenje kada se kursor miša nađe iznad kontrole.



Slika 5.16. Prikaz dve kratke poruke

Za realizaciju ove poruke pomoću CSS stilova treba da se:

- Realizuje prikazivanje poruke – svojstvo `hover`
- Odrediti pozicioniranje poruke
- Eventualno dodavanje efekata poput animacija

Dakle, potrebno je definisati element ili tekst za koji će biti definisana kratka poruka. Poruka može biti prikazana sa gornje ili donje strane, a ponekada i bočno. Što se tiče HTML koda, u okviru teksta za koji se definiše kratka poruka, definiše se i element koji će prikazivati kratku poruku. Za to je iskorišćen element `span`.

```
<a class="tooltips" href="#">
    Ovde je tekst za koji će se pojaviti kratka poruka.
    <span class="tooltip1">...Kratak opis...</span>

</a>
```

U primeru je kratka poruka vezana za element `a`, koji je podrazumevano `inline`. Zbog posebnog pozicioniranja poruke, za ovaj element dovoljno je definisati samo pozicioniranje: `a.tooltips{ position: relative; }`.
Za samu iskačuću poruku, tačnije za element `span` označen klasom `tooltip1` koji se nalazi u elementu `a` klase `tooltips`, biće:

```
a.tooltips span.tooltip1 {
    position: absolute;
```

```
    width:160px;
    color: white;
    background: #808080;
    height: 30px;
    line-height: 30px;
    text-align: center;
    visibility: hidden; /*isključiti pa pogledati*/
    border-radius: 3px;
}
```

Ukoliko privremeno promenimo kod komentarišući liniju `visibility:hidden;` dobijamo sledeći prikaz:

Slika 5.17. Međurezultat 1 u definisanju tooltip poruka

Pretpostavimo da želimo da iskačuća poruka ide ispod označenog teksta i to na sredini, bez obzira na njegovu širinu. Postavimo zatim iskačuću poruku ispod teksta dodavanjem stila:

```
a:hover.tooltips span.tooltip1 {
    visibility: visible;
    opacity: 0.8;
    /*120% znači da je 20% ispod kontejnera*/
    top: 120%;
    /*leva ivica do sredine kontejnera*/
    left: 50%;
    /*širina iskačućeg prozora*/
    width: 10em;
    /*postavljanje na sredinu kontejnera*/
    margin-left:-5em;
    z-index: 998; /*poslednji u redosl*/
}
```

Poruka se pojavljuje na 120% od vrha, dakle 20% ispod same linije teksta na koju se odnosi, a zatim se vrši pozicioniranje sredine prozora na sredinu teksta. Kako? Prvo se vrši pozicioniranje ulevo na 50% kontejnera, čime bi se leva ivica iskačućeg prozora poravnala sa

sredinom teksta, a zatim, dodatno se vrši pomeranje ulevo još za ½ širine prozora (-5em).

Slika 5.18. Međurezultat 2 u definisanju tooltip poruka

Na kraju, iskačuće poruke obično imaju i kratke strelice. Za kreiranje strelice iskorišćen je stil za kreiranje ivice oko sadržaja. Ako sadržaja nema, onda ivice kreiraju oblik . Jednostavnim postavljanjem transparentnih boja dobija se strelica u željenom pravcu.

```
.tooltips .tooltip1::after {
    content: ""; /*probati sa nekim sadržajem*/
    position: absolute;
    bottom: 100%;  /*dno strelice je tačno iznad*/
    left: 50%;
    margin-left: -3px;/*pomeraj za ½ širine ivice*/
    border-width: 6px;
    border-style: solid;
    /*postavite neke boje umesto transparent*/
    border-color:transparent transparent #808080 transparent;
    z-index: 999;
}
```

Slika 5.19. Konačan izgled iskačuće poruke

Galerija slika

Galerija predstavlja stranicu sastavljenu od skupa manjih slika koje omogućavaju pristup do originalne ili veće slike. Slike su raspoređene na neki pravilan način i često uz sliku ide i naslov.

Imajuću ovo u vidu možemo da napišemo sledeći HTML kod:

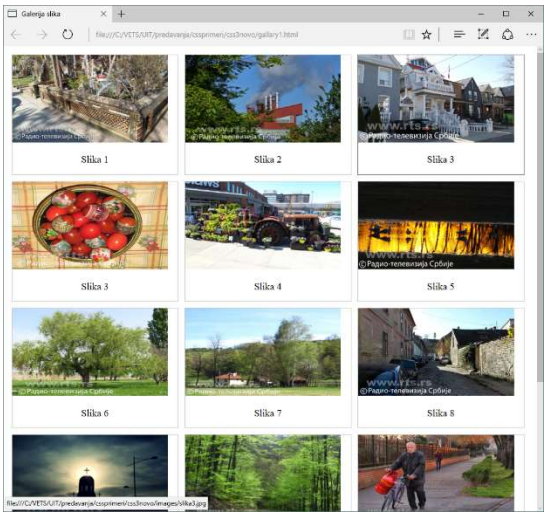
```
<div class="gallery">
    <a target="_blank" href="images/slika1.jpg">
       <img src="images/slika1.jpg">
    </a>
    <div class="desc">Slika 1</div>
</div>

<div class="gallery">
    <a target="_blank" href="images/slika2.jpg">
       <img src="images/slika2.jpg">
    </a>
    <div class="desc">Slika 2</div>
</div>
```

Svaka slika u galeriji, pošto predstavlja link, smeštena je u jedan `a` element. Zbog posebne stilizacije, naslov slike je u `div` elementu i zajedno sa `a` elementom smešteni su u jednu celinu tj drugi `div` element nad kojim se obavlja osnovna stilizacija za galeriju.

```
div.gallery {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 17em;
}

div.gallery:hover {
    border: 1px solid #777;
}
```

```
div.gallery img {
    width: 16em;
    height: 9em;
}

div.desc {
    padding: 15px;
    text-align: center;
}
```

Kao što se vidi, kontejner za galeriju sadrži odeljke u kojima je slika i naslov slike a koji su definisani stilom `div.gallery`. Odeljci su plutajući uz levu ivicu, određene širine, margine i ivice. Stil za odeljke je `div.desc` kojim se definiše da tekst bude na sredini slike uz određeni razmak sadržaja.



Slika 5.20. Primer jedne galerije slike

Rad sa ikonama u HTML dokumentu

Ikone su kolekcije unapred definisanih slika, obično često korišćenih i prepoznatljivih. Nisu unapred definisane veličine već se mogu lako skalirati, ali i menjati boju ili senku u zavisnosti od konkretne primene. Zato se kolekcije ikona realizuju preko posebnih, vektorski realizovanih, fontova koje sadrže slike. Tako se ne gubi na kvalitetu pri skaliranju, moguće je definisanje i boje i senke. Na ovaj način realizovane ikone se koriste pomoću posebnih klasa koje se obično primenjuju na elemente `<i>` i `<span>`.
Postoji nekoliko javno dostupnih kolekcija ikona. Pogledaćemo upotrebu sledećih:

- „Font Awesome Icons“
- „Bootstrap Icon“
- „Google Icons“.

Font Awesome Icon

Listu ikona ove kolekcije možete pogledati na stranici: http://fontawesome.io/icons/ . Da bi se koristila ova kolekcija potrebno je najpre u zaglavlju stranice dodati link do stilova:

```
<head>
    <link rel="stylesheet"
     href="http://cdnjs.cloudflare.com/ajax/libs/font-
     awesome/4.4.0/css/font-awesome.min.css">
</head>
```

U narednom primeru prikazan je primer u kome se ikonom zamenjuje stilska oznaka u neuređenoj listi sa ikonom iz kolekcije. Takođe je prikazana upotreba nekoliko slika sa definisanje veličine i boje. Izbor slika i prateće stilove možete preuzeti sa sajta.

```
<ul class="fa-ul">
  <li><i class="fa-li fa fa-check-square"></i>stavka1</li>
  <li><i class="fa-li fa fa-spinner fa-spin"></i>stavka2</li>
  <li><i class="fa-li fa fa-square"></i>stavka3</li>
  <li>automobil...<i class="fa fa-car" style="font-size:20px;
                    color:blue;"></i></li>
  <li>zabrana...<i class="fa fa-banlor:green;"></i></li>
  <li>twitter...<i class="fa fa-twitter"></i></li>
  <li>kombinacija više ikona: zabrana (red boje) + camera
     <span class="fa-stack fa-lg">
        <i class="fa fa-camera fa-stack-1x"></i>
        <i class="fa fa-ban fa-stack-2x text-danger"
                  style="color:red;"></i>
     </span>
  </li>
</ul>
```

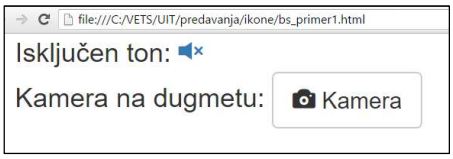
Bootstrap icons

Sličan je način primene ikona iz kolekcije *bootstrap*. Više detalja može se naći na stranicama: http://getbootstrap.com/components/ odnosno http://glyphicons.com/. Uključenje se obavlja pomoću:

```
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootst
rap.min.css">
```

Primer upotrebe nekoliko ikona:

```
Isključen ton:
<a href="#">
  <span class="glyphicon glyphicon-volume-off"></span>
</a> <br />
Kamera na dugmetu:
<button type="button" class="btn btn-default btn-sm">
  <span class="glyphicon glyphicon-camera"></span> Kamera
</button>
```



Slika 5.21. Primer jedne galerije slike

Google icons

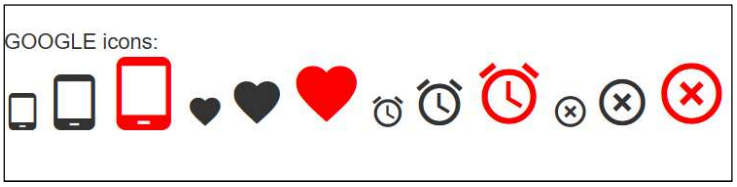
Na kraju pokažimo primer sa Google ikonama. Uključivanje stilova:

```
  <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

Primer HTML koda:

```
<i class="material-icons">tablet_android</i>
<i class="material-icons" style="font-
size:36px">tablet_android</i>
<i class="material-icons" style="font-
size:48px;color:red">tablet_android</i>
<i class="material-icons">favorite</i>
<i class="material-icons" style="font-size:36px">favorite</i>
. . .
```

```
<i class="material-icons">alarm</i>
<i class="material-icons" style="font-size:36px">alarm</i>
. . .
<i class="material-icons">highlight_off</i>
<i class="material-icons" style="font-
size:36px">highlight_off</i>
. . .
```



Slika 5.22. Prikaz nekih Google ikona

Pitanja za proveru znanja

1. Šta je CSS reset?
2. Kako se obavlja zamena teksta slikama i koji je opravdan razlog zamene u dizajnu?
3. Kako se koriste CSS sprajtovi? Šta se dobija njihovom primenom?
4. Kojim HTML elementom možete izvršiti grupisanje elemenata u formi?
5. Kako se vezuju labele uz elemente za unos podataka?
6. Uraditi stilizovanje jedne forme koja sadrži:
    a. tekstualna polja za unos sa labelama „ime“ i „prezime“,
    b. tekstualnu oblast sa labelom „poruka“
    c. grupu radio dugmadi označenu sa „muški“ i „ženski“,
    d. grupu dugmadi za potvrdu označenu sa „radio“, „tv“ i „dvd“.

Deo 6: Prilagodljivi dizajn

- Meta svojstvo **`viewport`**
- Mrežasta organizacija sadržaja
- Medija upiti
- Uvod u fleksibilni model okvira

Pojam prilagodljivi dizajn (eng. „Responsive Web Design - RWD“) uveo je Ethan Marcotte 2010, a predstavlja dizajn web stranica koji se prilagođava veličini ekrana za prikaz. Zbog dominantne upotrebe mobilnih uređaja, ova tehniha dizajniranja danas je postala *de facto* standard u web dizajniranju. Njegovom primenom korisniku se pruža tzv. najbolje iskustvo.
Prilagodljivi dizajn se zasniva na nekoliko osnovnih tehnika:

- medija upiti,
- skalabilne slike,
- fluidne mreže.

Meta svojstvo `viewport`

U okviru meta svojstava svake stranice postoji svojstvo za podešavanje prozora (otvora) za prikaz (eng. viewport). Prozor za prikaz definiše deo stranice koji je vidljiv korisniku. Definisanje prozora za prikaz značajno je za mobilne uređaje jer obezbeđuje

prilagođavanje prikaza različitim veličinama ekrana, pošto deo stranice koji se prikazuje može biti mali kod mobilnih uređaja. Sa druge strane može biti jako veliki ako se radi o televizijskim ekranima ili bioskopskim platnima.

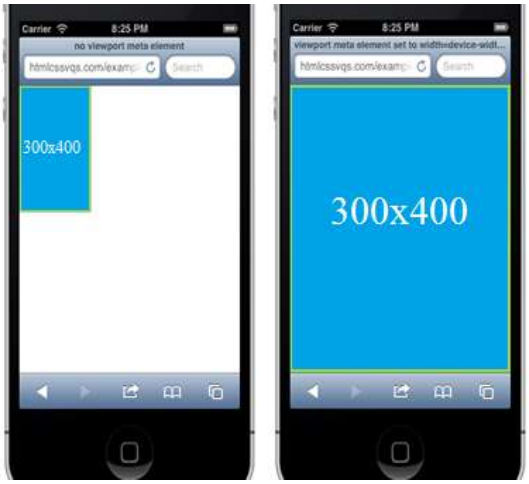
Primer definisanja jednog prozora za prikaz:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Značenje sadržaja atributa `content` su:

- `width` – širina koju korisnik vidi u čitaču, obično u odnosu na širinu uređaja, (CSS pikseli).
- `initial-scale` –vrednost početnog skaliranja (zoom-a)
- `maximum-scale`, `minimim-scale` –maksimalna odnosno minimalna vrednost za skaliranje

Na ovaj način kontroliše se prozor prikaza i dobija mogućnost da se, čak i kada postoji velika rezolucija, zbog manje fizičke veličine ekrana prozor za prikaz sadržaja smanji. Slika koja ilustuje primenu atributa `viewport` data je na slici 6.1.



Slika 6.1. Ilustracija prikaza jednog odeljka sa prilagođenim prozorom

Treba obratiti pažnju na vrednost koja se koristi za podešavanje širine: `width=device-width`, a postavlja širinu prozora za prikaz na širinu ekrana uređaja. Ono što se dobija definisanjem prozora za prikaz možemo ilustrovati na dva uporedna primera:

Primer 1.

```
<html>
<head>
    <style>
      img{
          width:474px;
          height:296px;
      }
    </style>
</head>
<body>
    <img src="pejzaz1.jpg"/>
    <p>Lorem ipsum
dolor...</p>
</body>
</html>
```

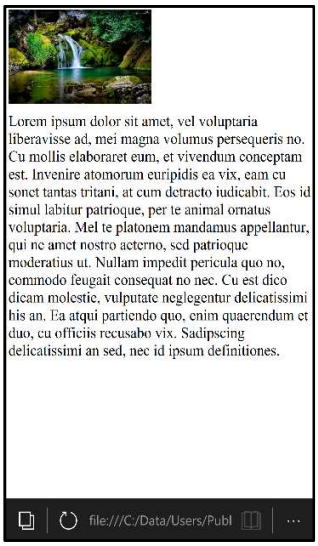
Primer 2.

```
<html>
<head>
   <meta  name="viewport"
content="width=device-width,
initial-scale=1.0"/>
   <style>
     img{
        max-width:100%;
        height:auto;
     }
   </style>
</head>
<body>
   <img src="pejzaz1.jpg"/>
   <p>Lorem ipsum
dolor...</p>
</body>
</html>
```

Za ova dva primera dajemo uporedni prikaz na dva različita uređaja: tradicionalni računar i mobilni telefon. Posebnu pažnju obratiti na prikaz za mobilne uređaje. Rezultat je više nego očigledan.

Na mobilnom uređaju

Primer 1.

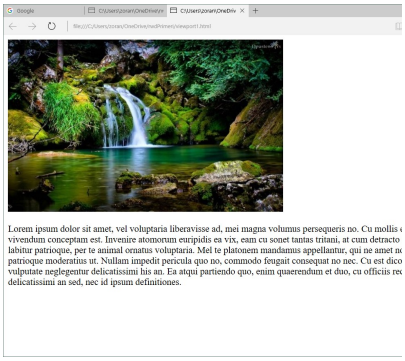


Primer 2.

Na desktop-u normalne širine

Primer 1.

Primer 2.



Na desktop-u smanjene širine

Primer 1.

Primer 2.

Slike 6.2. Prikaz sa i bez promene prozora za prikaz (viewport)

Sa slike je očigledno da za Primer 2. postoji prilagođenje prikaza širini uređaja. Takođe, može se zapaziti da se prilagođavanje ispoljava i pri promeni širine prozora čitača. Ukoliko se sadržaj prilagođava novoj širini, onda je dizajn projektovan da bude prilagodljiv uređaju. U slučaju desktop računara prikaz ostaje identičan.

Skrolovanje sadržaja

Prilagođavanje prikaza korisniku predstavlja osnovni cilj odnosno zahtev u dizajnu. Kada je reč o skrolovanju sadržaja na stranici, smatra se da su korisnici naučeni da skroluju web stranice vertikalno, ali ne i horizontalno! Kaže se još da je „loše korisničko iskustvo“ ukoliko je web dizajn takav da primorava korisnika da skroluje horizontalno ili da vrši promenu zoom-a da bi video neki sadržaj. Zbog toga treba poštovati nekoliko principa:

1. Ne koristite fiksne široke elemente – na primer, ukoliko je neka slika prikazana šire nego što je prozor za prikaz to znači da će korisnik morati da skroluje horizontalno.

2. Sadržaj ne sme da se oslanja na određenu širinu prozora za prikaz da bi imali dobar izgled.
3. Koristite CSS medija upite da bi ste primenili drugačiji stil za drugačije veličine ekrana.

Relativne jedinice mere

Pri projektovanju prilagodljivih stranica dizajnerima stoji na raspolaganju podrška novog CSS3 standarda i sa njim upotreba relativnih jedinica mere. U tabeli 6.1. dat je prikaz tih jedinica sa kratkim opisom.

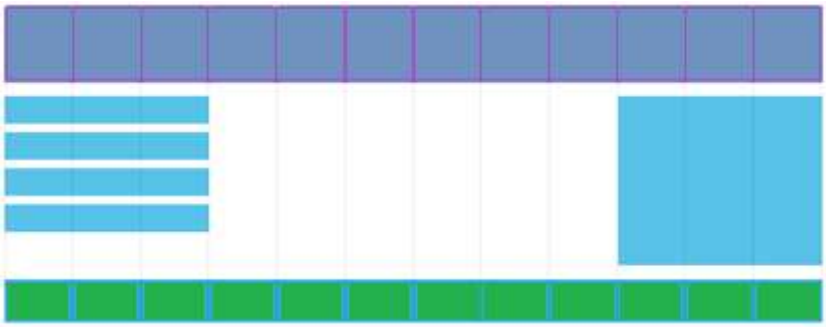
Tabela 6.1. Nove relativne jedinice mere

Jedinica	Opis
vw	Predstavlja 1% od širine viewport-a
vh	Predstavlja 1% od visine viewport-a
vmin	Predstavlja 1% od manje dimenzije viewport-a
vmax	Predstavlja 1% od veće dimenzije viewport-a

Mrežasta organizacija sadržaja

U prilagodljivom dizajnu jedini prihvatljiv način organizacije sadržaja je zasnovan na primeni relativnih mera, pre svega u odnosu na širininu ekrana. Pošto je sadržaj najčešće organizovan po kolonama, dakle širine kolona treba da budu definisane procentima - % odnosno em -jedinicama.
Laka organizacija sadržaja po kolonama je iskorišćena za projektovanje unapred pripremljenih okvira (eng. framework) za dizajn prilagodljivih stranica. Koristeći unapred pripremljene okvire postiže se jednostavnije i brže dizajniranje prilagođenih stranica. Obično se okviri zasnivaju na primeni stilova koji unapred definišu neku mrežastu strukturu (eng. grid). Mrežasta struktura je

organizacija sadržaja definisana kolonama odnosno redovima u koje se smeštaju elementi stranice.
Na slici 6.4. prikazana je jedna mrežasta struktura od 12 kolona iste širine. Elementi koji se postavljaju na stranicu, postavljaju se definišući širinu koja se zasniva na broju kolona. Svih 12 kolona zajedno čini 100% širine.



Slika 6.3. Mrežasta organizacija sadržaja

Svojstvo box-sizing

Veći broj kolona u šablonu je neophodan kako bi se dobio što bolji dizajn odnosno sa više detalja, ali je neophodna potpuna kontrola cele širine prozora. Zato svi HTML elementi treba da imaju podešeno svostvo `box-sizing`.

```
* {
    box-sizing: border-box;
}
```

Ovim svojstvom se osigurava da su `pedding` i `border` uključeni u ukupnu širinu i visinu elemenata. Na ovaj način se značajno olakšava definisanje sadržaja čija ukupna širina mora biti 100%. Inače, podrazumevana vrednost ovog svojstva tzv. standardni box model je: `box-sizing: content-box`, a tada se padding i border se dodaju na širinu/visinu.
U konkretnom slučaju, sa slike 6.3., iskazano u procentima širina jedne kolone je: 100% / 12 = 8.33%. Formiraju se klase za svih 12 kolona istog prefiksa, `class="col-"` .
Napomena: Ovako formirano ime klase omogućava CSS pristup pomoću atributa koji ciljaju imena svojstva (*=, ^=, $=).

U kodu koji sledi naveden je CSS kod za 12 klasa različitih širina, kao i zajedničko definisanje svojstava: float, padding, border.

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

[class*="col-"] {
    float: left;
    padding: 15px;
    border: 1px solid red;
}
```

Svaki red u prikazu koristi jedan ili više odeljaka koje su širine određene kao broj kolona. Ukupna suma širina svih odeljaka jednog reda ne može biti veća od 12 kolona. Na primer:

```
<div class="col-3">
    <ul>
        <li>NRT</li>
        <li>RT</li>
        <li>EPO</li>
        <li>...</li>
    </ul>
</div>
<div class="col-9">
    <h1>Najbolji studijski programi</h1>
    <p>Na osnovu nepoznatih kriterijuma...</p>
    <p>Ipak NRT je naj.</p>
</div>
```

Važne osobine ovog dizajna su:

- Sve kolone u redu imaju postavljeno svojstvo `float` na vrednost `left`.
- Da bi se omogućio prelazak u novi red ukida se plutanje nakon svakog reda, na primer pomoću stila:

```
.row::after {
    content: "";
    clear: both;
    display: block;
}
```

Primer realizacije meni kontrole

Nakon ovih svojstava mogu se dodati posebne klase za sređivanje izgleda, menija na primer:

CSS	HTML

`.menu ul {`
`  list-style-type: none;`
`  margin: 0;`
`  padding: 0;`
`}`
`.menu li {`
`  padding: 8px;`
`  margin-bottom: 7px;`
`  background-color :#33b5e5;`
`  color: #ffffff;`
`}`
`.menu li:hover {`
`  background-color: #0099cc;`
`}`
	`<div class="col-3 menu">`
	`  <ul>`
	`  <li>NRT</li>`
	`  <li>RT</li>`
	`  <li>EPO</li>`
	`  <li>...</li>`
	`  </ul>`
	`</div>`

Slika 6.4. Primer jedne stranice u mrežastoj organizaciji

Medija upiti

Medija upiti su nova CSS tehnika uvedena sa standardom CSS3 i ujedno predstavlja jednu od osnovnih karakteristika prilagodljivog dizajna. Medija upiti su logički izrazi. Omogućavaju primenu različitih stilova za različite medije odnosno karakteristike medija. Za prilagodljivi dizajn, gotovo uvek, reč je o mediju koji je ekran, a osnovna karakteristika medija je širina ekrana. Upiti se zasnivaju na korišćenju `@media` pravila. Sintaksa je:

`@media "mediji and/or uslovZaOsobinuMedija"`.

Na primer:

```
<style>
   @media screen and (max-width: 500px) {
        body {
           background-color: yellow;
        }
   }
</style>
```

predstavlja stil koji se primenjuje pod uslovom da je tip medija ekran i da je maksimalna širina 500px.
Postoji 10 različitih tipova medija. Mi ćemo u nastavku isključivo koristiti screen koji je najčešći u praktičnoj upotrebi. Neki tipovi medija su:

`all` – svi mediji
`print` – stranični medij u režimu prikaza štampača
`speech` – snimanje glasa
`tv` – televizija,...
Tip medija se može koristiti pri definisanju stila ili pri povezivanju sa spoljnim fajlovima, na primer:

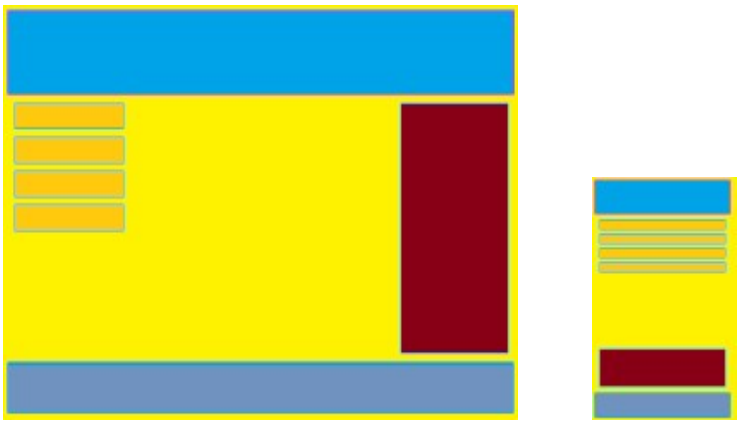
```
<link href="stilovi.css" media="screen">

<style media="print">
```

Osobine medija predstavljaju karakteristiku koja se testira u medija upitu. Postoji 13 različitih osobina koje se mogu testirati. Neke od njih su:
`aspect-ratio` – odnos širine i visine uređaja
`color` – broj bitova o boji komponente boje
`resolution` – gustina piksela
`orientation` – portrait ili landscape režim
`height` – visina oblasti prikaza (po standardu treba izbegavati device-height – visina prikazane površi)
`widht` – širina oblasti prikaza (po standardu treba izbegavati device-width – širina prikazane površi)
. . .
Svim osobinama može da se doda prefiks `min-` ili `max-` . Na primer:

```
@media (min-width: 350px) { . . . }
@media (min-width: 350px and max-width) { . . . }
```

Tačke preloma

Tačke preloma predstavljaju definisane širine ekrana koje predstavljaju granicu za primenu različitih stilova. Na primer, skaliranje kolona u prikazu može biti dobro samo do određene širine, ali za slučaj prikaza na mobilnim telefonima u *portrait* modu skaliranje svakako nije dobro. Za takve prikaze bolje je imati drugačiju organizaciju sadržaja, na primer u jednoj koloni.



Slika 6.5. Dve strukture sadržaja za jednu tačku preloma

U narednom primeru dat je primer CSS koda za jednu tačku preloma (prekida) u slučaju mrežaste strukture.

```
/* Desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

```
@media only screen and
(max-width: 768px) {
    /* Mobilni */
    [class*="col-"] {
        width: 100%;
    }
}
```

Dizajn za mobilne uređaje izvodi se prilagođavanjem postojećeg, desktop, dizajna postavljanjem celokupnog sadržaja u jednu kolonu.

Prvi dizajn za mobilne uređaje – (eng. *Mobile First*)

Označava dizajniranje stranice prvo za mobilne uređaje, a primenom tačke preloma vrši se prilagođavanje za desktop ili neki drugi uređaj.

```
/* Mobilni */
[class*="col-"] {
    width: 100%;
}
```

```
@media only screen and (min-width:
768px) {
    /* Desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    ....
}
```

Više tačaka preloma

Ako se prilagodljivi dizajn projektuje ne samo za mobilni i desktop već i za tablet uređaje onda se može postaviti dodatna, druga, tačka preloma.

```
/*Mobilni: */
[class*="col-"] {
    width: 100%;
}
@media only screen and (min-width: 600px) {
    /* Tablet: */
    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
    .col-m-4 {width: 33.33%;}
    ....
}
@media only screen and (min-width: 768px) {
    /* Desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    ....
}
```

Odvojene kolekcije stilova

Dizajneri često posežu za rešenjima u kojima posebno definišu stilove za posebne prikaze. Mogući nedostatak je ponavljanje istih stilova što se može donekle prevazići izdvajanjem zajedničkih stilova u poseban dokument.
Na primer, u fajlovima *small.css* i *large.css* mogu biti stilovi za određene ekrane.

```
<link rel="stylesheet" media="screen and (min-width:200px) and (max-width: 500px)" href="small.css">

<link rel="stylesheet" media="screen and (min-width:501px) and (max-width: 1100px)" href="large.css">
```

Napomena: Zbog nekih starijih čitača po nekada se koristi i `only screen`, ali i kao naglašavanje da se ne koristi neki drugi prikaz osim `screen` prikaz, na primer *print*.

Orijentacija ekrana

Pomoću medija upita može da se koristi tekuća orijentacija ekrana i da se dizajn prilagodi orijentaciji. Moguće orijentacije su:

- Portrait,
- Landscape.

Na primer, ako se u zavisnosti od orijentacije menja pozadinska boja.

```
@media only screen and (orientation: landscape) {
    body {
         background-color: yellow;
    }
}
```

Slike u prilagodljivom dizajnu

Da bi slika u prilagodljivom dizajnu bila prilagođena širini viewporta neophodno je koristiti relativno podešavanje širine.

```
img {
    width: 100%;
}
```

Međutim, realan slučaj je da se koristi slika ne preterano velike rezolucije, recimo širine 480px što bi u slučaju desktop varijante prikaza dovelo do širenja slike do dimenzija prikaza pa time istovremeno i do narušavanja kvaliteta slike u prikazu. Zato se umesto toga koristi

```
img {
    max-width: 100%;
}
```

Prilagođavanje pozadinske slike

Pozadinske slike mogu takođe biti prilagodljive i skalirajuće. Postoji nekoliko različitih metoda za dobijanje efekta prilagodljivosti:

- Ako je svojstvo `background-size` postavljeno na vrednost `contain`, pozadinska slika će biti skalirana i prilagodiće se površini koja je sadrži. Slika će sačuvati proporcionalnost između širine i visine slike (*„aspect ratio“*):

- Ako se svojstvo `background-size` postavi na vrednosti `"100% 100%"`, pozadinska slika će biti razvučeni preko cele klijentske oblasti:

- Ako se svojstvo `background-size` postavi na vrednost "`cover`", pozadinska slika će biti skalirana preko cele površi kontejnera. Vrednost "cover" čuva proporcionalnost širine i visine slike tako da se zbog toga deo slike obično iseca:

Različite slike za različite uređaje

Kao što je velika slika dobra za veliki ekran, mala je za mali. Da bi se izbeglo nepotrebno učitavanje velike slike za male ekrane, može se definisati potpuno druga tj. već obrađena slika manje veličine.

```
body {
    background-image: url('pejzaz1_mobile.jpg');
}

@media only screen and (min-width: 400px) {
   body {
     background-image: url('pejzaz1_desktop.jpg');
   }
}
```

Video

Video element ima svojstvo koje može da se iskoristi tako da se veličina prilagođava širini uređaja podešavanjem na 100% širine ekrana, sa eventualnim ograničenjem koristeći max-width svojstvo.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width,
                initial-scale=1.0">
    <style>
       video {
           max-width: 100%;
       }
    </style>
</head>
```

```
<body>
    <video width="600" controls>
        <source src="mo1.mp4" type="video/mp4">
    </video>
</body>

</html>
```

Uvod u fleksibilni model okvira

Fleksibilni okviri predstavljaju novi način prikaza uveden uz CSS3. Upotreba fleksibilnog koncepta garantuje predvidivo prikazivanje elemenata u slučaju različitih uređaja za prikaz. Za neke primene ovo je napredniji model prikaza koji daje bolje karakteristike i veće mogućnosti.
Fleksibilni model okvira sastoji se od:

- kontejnera i
- stavki.

Kontejner se deklariše postavljanjem svojstva `display` na vrednost `flex` (za blok element) odnosno `inline-flex` (za inline elemente). Unutar jednog kontejnera smešta se jedna ili više stavki.
Sve što je van **fleksibilnog kontejnera**, ali takođe i ono što je unutar **fleksibilnih stavki**, prikazuje se na uobičajen način.
Fleksibilni kontejner definiše kako se prikazuju stavke u njemu. Stavke se pozicioniraju duž jedne linije. Podrazumevano u jednom kontejneru postoji samo jedna `flex` linija. Takođe, podrazumevano pozicioniranje ide od leve ka desnoj strani.
Važno:

- Ovo svojstvo još uvek nije potpuno identično za sve čitače pa se preporučuje upotreba prefiksa specifičnih za pojedine čitače:
  - `-moz`, Mozilla Browsers (Firefox)
  - `-webkit`, Safari, Chrome)
  - `-o`, Opera
  - `-ms`, `<!--[if IE]>…<![endif]-->`, Internet Explorer

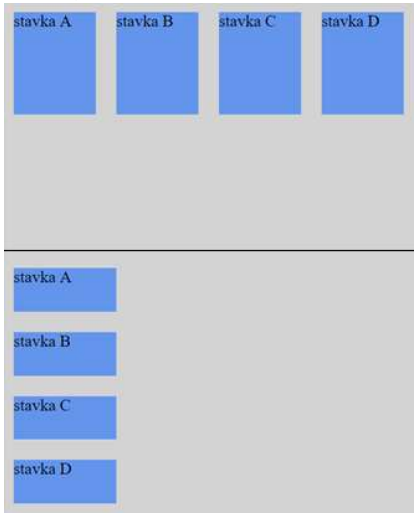
- Definisanje flex linije se vrši postavljanjem svojstva `directon` , na primer:

```
.flex-container {
     ...
     display:flex;
     flex-direction:row;
 }
```

postavlja pozicioniranje s leva na desno i odozgo na dole što je definisano svojstvom `flex-direction:row`.

- Ostale moguće vrednosti za pozicioniranje su:
- `row-reverse` – suprotno od podrazumevanog tj. s desna na levo,
- `column` – stavke se prikazuju po kolonama tj. vertikalno,
- `column-reverse` – po kolonama, ali u suprotnom redosledu.

Na slici 6.6. prikazan je jedan raspored odeljaka horizontalno i za iste odeljke vertikalno.



Slika 6.6. Fleksibilni raspored po horizontali i vertikali

Horizontalna raspodela praznog prostora

U fleksibilnom rasporedu stavki, kada stavke ne ispune ukupan unutrašnji prostor važno je upravljati viškom belog prostora. Za to se koristi svojstvo justify-content. Moguće vrednosti su:

- `flex-start` – Podrazumevano. Stavke su pozicionirane od početka kontejnera,

- `flex-end` – Stavke su pozicionirane na kraju kontejnera,
- `center` – Stavke su pozicionirane oko centra kontejnera,
- `space-between` – Stavke su pozicionirane sa prostorom između linija,
- `space-around` – Stavke su pozicionirane sa prostorom ispred, između i posle linija.

Slika 6.7. Rapodela belog prostora pomoću `center` odnosno `space-between`

Vertikalna raspodela praznog prostora

U slučaju kada stavke nisu popunile ukupan raspoloživ prostor po vertikali, vertikalna raspodela praznog prostora se definiše pomoću svojstva `align-items`. Moguće vrednosti su:

- `stretch` – Podrazumevana vrednost. Stavke se razvlače i tako popunjavaju ukupan kontejner.
- `flex-start` – Pozirionirane od vrha.
- `flex-end` – Pozicionirane na dno kontejnera.
- `center` – Stavke su pozicionirane vertikalno na sredini.
- `baseline` – Stavke su pozicionirane na baznoj liniji kontejnera.

Prelom sadržaja

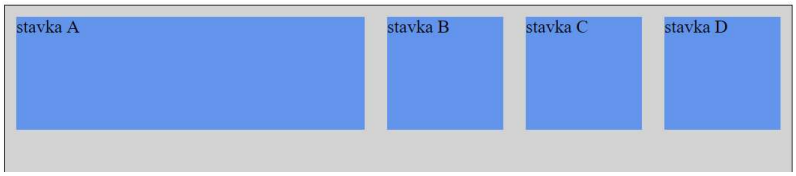
Svojstvo `flex-wrap` definiše da li se stavke u kontejneru prelamaju u slučaju da nema dovoljno praznog prostora u jednoj liniji. Moguće vrednosti su sledeće:

- `nowrap` – podrazumevano,
- `wrap` – prelom postoji ako sadržaj ne može da stane,

- `wrap-reverse` – prelom ali u obrnutom redosledu.

Dužina stavke

Svojstvom `flex` specificira se dužina jedne stavke, relativno u odnosu na ostatak preostalih stavki u istom kontejneru. U narednom primeru prvi div dobija 3 od ukupno 6 vrednosti (50%), drugi, treći i četvrti dobija širinu 1 od 6. Ova vrednost definiše i proporciju sa kojom se stavke povećavaju.




```
#prvi1 {
    flex:3;
}
#prvi2 {
    flex:1;
}
```

```
#prvi3 {
    flex:1;
}
#prvi4 {
    flex:1;
}
```

Slika 6.8. Definisanje dužine stavki

Fleksibilna širina stavki i redosled

Kao što se definiše širina stavki u fleksibilnom okviru, odnosno način proširivanja, isto tako se može definisati i način tj. srazmernost pri skupljanju sadržaja. Ta vrednost se može zadati kao druga vrednost, a kao treća se zadaje početna veličina u odnosu na koju se vrši računanje. Na primer: flex: 1 3 150px;
Redosled se može posebno definisati svojstvom `order`. Niža vrednost ovog svojstva znači prvenstvo u prikazu. Ako dve stavke imaju istu vrednost onda se primenjuje podrazumevani redosled.

```
#prvi1 {
    flex: 1 3 150px;
}
#prvi2 {
    flex: 1 1 150px;
```

```
}
#prvi3 {
    flex: 1 1 150px;
}
#prvi4 {
    flex: 1 1 150px;
}
```

Pitanja za proveru znanja

1. Šta je to prilagodljivi dizajn i šta je važno za postizanje dobrog „korisničkog iskustva“?
2. Kojim tehnika se postiže prilagođenje?
3. Koje relativne jedinice mere postoje? Objasniti ih.
4. Šta je to mrežasta struktura? Napisati stil za jednu.
5. Kada se primenjuju medija upiti i šta se postiže njihovom primenom?
6. Objasniti tačke preloma.
7. Objasniti fleksibilni model okvira. Navesti primere organizacije sadržaja primenom ovog modela

Deo 7: Odabrana svojstva CSS3

- Nova vizuelna svojstva
- Transformacije
- Animacije
- Tranzicije

U okviru ovog poglavlja obrađen je jedan skup novih CSS svojstava uveden sa standardom CSS3 koji prikazuje osobine novog standarda sa novim vizuelnim efektima. Radi se o vizuelnoj transformaciji objekata, animacijama i tranzicijama. Prikazani efekti mogu da posluže za prikaz objekata u 3D okruženju, ali i za posebnu namenu uključujući i korisničko zadovoljstvo u radu.

Nova vizuelna svojstva

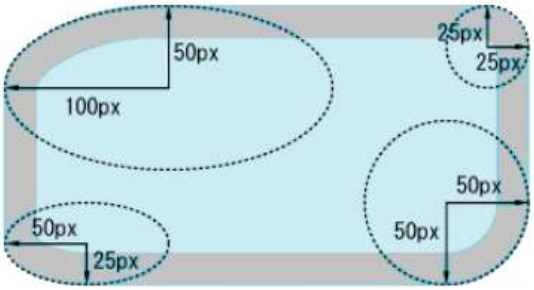
Zaobljeni uglovi

Jedna od novih mogućnosti koju donosi CSS3 je mogućnost preciznog i lakog definisanja zaobljenosti uglova. Zaobljeni uglovi HTML elementa mogu se definisati posebno za svaki ugao ili istovremeno za više uglova. Takođe, svaki ugao može biti zaobljen simetrično (kružno) ili asimetrično (elipsoidno).

U opštem slučaju, za definisanje zaobljenosti jednog ugla potrebno je definisati dve vrednosti koje određuju poluprečnike elipse. Definisane vrednosti se odvajaju kosom crtom: prva vrednost je horizontalni poluprečnik, a druga vertikalni poluprečnik. Ako je zaobljenje kružno onda je dovoljna jedna vrednost po uglu. Vrednost može biti zadata kao % ukupne dužine elementa ili koristeći apsolutnu vrednost. Sintaksa je:

`border-radius: 1-4 dužine|% / 1-4 dužine|%`

Za primenu ovog stila poželjno je koristiti i prefiks specifičan za određene čitače. Na ovom mestu nećemo se baviti njihovim sintaksama, ali ćemo u nekim primerima navesti kako bi se uočila razlika. Na slici 7.1. prikazan je opis zaobljenja uglova sa istaknutim poluprečnicima koji definišu zaobljenje.



Slika 7.1. Zaobljenje uglova elementa

Na sledećoj slici prikazan je izgled i CSS kod četiri primera sa različitim zaobljenjima:


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


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


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

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

Slika 7.2. Primeri zaobljavanja uglova

Senčenje

Senka objekta ostavlja utisak treće dimenzije odnosno dubine. Senke se koriste, ne samo zbog estetskog efekta, već i kao jedna opcija da se obezbedi jasnoća vidljivosti granica objekta na podlozi (npr. tekst u titlovima). Osim senki koje mogu biti na tekstu, CSS3 omogućava i senke na elementima.
Svojstvo kojim se definiše senka je `box-shadow` i ovo svojstvo se može primeniti na većini HTML elemenata (`div`, `p`, `img`, `span`,...).
Sintaksa za senku je:

`box-shadow: #rrggbb xof yof rza`

gde je,

- `#rrggbb` ili `rgb(r,g,b)` – boja senke,
- `xof`, `yof` – pomeraj senke u odnosu na objekat,
- `rza` – rastojanje zamagljenja, ako postoji.

```
width: 200px;
height: 100px;
background-color: lightblue;
box-shadow: 10px 10px 5px #999;
```

```
width: 200px;
height: 100px;
background-color: lightblue;  box-
shadow: 5px -5px 5px
```

```
width: 200px;
height: 100px;
background-color: lightblue;
box-shadow:
inset rgb(0,50,100) -5px -5px 15px,
inset white 5px 5px 15px
```

Slika 7.3. Primeri realizacije tri različite senke

Kao što se vidi, moguće je istovremeno zadati više stilova za senke razdvojenih zarezom i tako dobiti novi efekat kao u trećem primeru.

Osim senčenja elemenata može se osenčiti i tekst. Pravila za podešavanje senke su identična, razlika je u sintaksi:

```
text-shadow: #FF0000 2px 2px 8px;
```

Gornji primer dao bi sledeći efekat:

Slika 7.4. Senčenje teksta

Gradient

Gradijent ili tonski prelaz je način da se oboji objekat bojom koja se postepeno menja iz jedne u drugu. Korišćenjem ovog svojstva, može se napraviti gradijent od najmanje dve boje. Broj boja, pravac i način promene definiše se primenom stilova.
Da bi se proces postavljanja gradijenta olakšao, često se koristi pomoć nekog *online* generatora, na kojima se odmah vidi efekat izbora parametara kao što su orijentacija, boje, položaji.

Nekoliko *online* generatora tonskih prelaza je dato na sledećim linkovima:

- http://www.colorzilla.com/gradient-editor/,
- http://css3gen.com/gradient-generator/ .

Postoje dve vrste gradijenta:

- linearni i
- radijalni.

Sintaksa za definisanje boje pomoću gradijenta je:

```
background:linear-gradient(pravac, clr1, clr2, ...);
```

Navedeni primer se odnosi na pozadinsku boju. Pravac promene tj. gradijenta se definiše argumentom `pravac`. Pravac se može definisati pomoću ključnih reči ili koristeći ugao. Argumenti clr1, clr2 su dve boje za koje se definiše prelaz u definisanom pravcu. Na primer

```
background: linear-gradient(to bottom, green, blue);
```

predstavlja promenu od zelene ka plavoj počev od vrha prema dnu. Ili ako želimo da obezbedimo istu promenu, ali pod uglom od 70 stepeni, CSS kod bi bio:

```
background: linear-gradient(70deg, green, blue);
```

Ukoliko želimo da dobijemo prelaz po svim pravcima počev od sredine elementa koristi se radijalni gradijent.

```
radial-gradient(green, blue, rgba(200,200,200,0.2));
```

U prethodnom primeru generiše se prelaz od zelene u centru elementa, preko plave, do svetlo sive sa izvesnom dozom prozirnosti koja je zadata kao četvrti argument funkcije `rgba`.

Postavljanje teksta po kolonama

CSS3 nudi jednostavan način za prikaz sadržaja u više kolona. Ovo je tipično u slučaju prikaza teksta na široj površi. Na ovaj način se dobija na čitljivosti. Za web čitače Chrome, Opera i Firefox potrebno je koristiti prefikse. Slede svojstva i njihov opis:

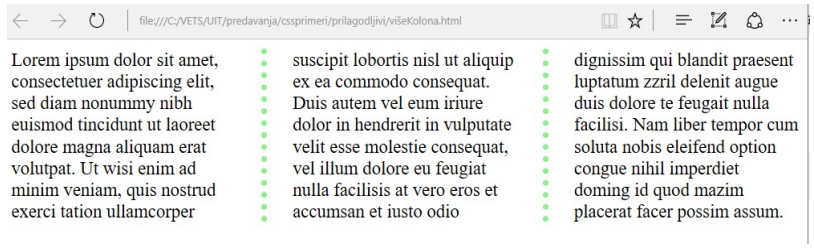
- `column-count` – broj kolona
- `column-gap` – prostor između kolona

Takođe se dodatno može definisati stil između kolona:

- `column-rule-style` – stilovi između kolona
- `column-rule-width` – širina korišćenog stila
- `column-rule-color` – boja korišćenog stila

Pogledajte naredni primer:

```
<style>
   .triKolone {
      -webkit-column-count: 3; /* Chr, Saf, Op */
      -moz-column-count: 3; /* Firefox */
      column-count: 3;

      -webkit-column-gap: 50px; /* Chr, Saf, Op */
      -moz-column-gap: 50px; /* Firefox */
      column-gap: 50px;

      -webkit-column-rule-style: dotted; /* Chr, Saf, Op */
      -moz-column-rule-style: dotted; /* Firefox */
      column-rule-style: dotted;

      -webkit-column-rule-width: 5px; /* Chr, Saf, Op */
      -moz-column-rule-width: 5px; /* Firefox */
      column-rule-width: 5px;

      -webkit-column-rule-color: lightgreen; /* Chr, Saf */
      -moz-column-rule-color: lightgreen; /* Firefox */
      column-rule-color: lightgreen;
}
</style>
```



Slika 7.5. Organizacija teksta po kolonama

Transformacije

Transformacija je vizuelni efekat koji utiče na element da menja određene osobine kao što su: oblik, veličina ili pozicija. CSS3 transformacije omogućuju pomeranje, skaliranje, rotaciju i rastezanje elemenata.
CSS koristi vizualni model opisan koordinatnim sistemom u kom je element pozicioniran. Pozicije i veličine elemenata izražavaju se u pikselima počevši od gornjeg levog ugla. Koordinatni sistem se može menjati pomoću svojstva `transform`. Atribut je podržan uz primenu specifičnih prefiksa.
Svojstvo `transform` omogućava 2D ili 3D transformacije na nekom elementu. Sintaksa je:

`transform: funkcijaTransformacije(arg);`

Na primer:

```
#myDIV {
    border: solid 1px;
    width: 200px;
    height: 200px;
    background: red;
    transform:skewX(-30deg);
}
```

Gde je `skewX` jedna funkcija transformacije koja vrši iskošavanje. U ovom primeru, za -30 stepeni oko X ose. Ostale funkcije su:

- `matrix`(x,x,x,x,x,x), `matrix3d`(x,x,x,x,x,x,x,x,x,x,x,x)
  Opšti oblik transformacije u vidu matričnih operacija.
- `translate`(x,y), `translate3d`(x,y,z), `translateX`(x), `translateY`(y), `translateZ`(z)
  Vrši pomeranje objekta po nekoj osi. Pomeranje može da se izvodi posebno po svakoj osi, ili istovremeno po više osa.

```
.translation {
    -webkit-transform:translateX(120px);
    transform:translateX(120px);
}
```

- `scale`(x,y), `scale3d`(x,y,z), `scaleX`(x), `scaleY`(y), `scaleZ`(z)
  Skaliranje objekta po jednoj ili vše osa.
- `rotate`(ugao), `rotate3d`(x,y,z, ugao), `rotateX`(ugao), `rotateY`(ugao), `rotateZ`(ugao)
  Rotiranje objekta u bilo kom smeru oko bilo koje ose. Na primer kod klase za rotaciju za 45 stepeni:

```
.rotate {
    -webkit-transform:rotate(45deg);
    transform:rotate(45deg);
}
```

- `skew`(x- ugao,y- ugao), `skewX`(ugao), `skewX`(ugao)
  Vrši iskošenje objekta u pravcu jedne ili obe ose. U primeru je urađeno iskošenje slike za 45 stepeni.

```
.skew {
    -webkit-transform:skew(45deg);
    transform:skew(45deg);
}
```

- `perspective`(n) – Određuje pogled za 3D transformisani element.

U narednom primeru objedinjeno je nekoliko transformacija za uporedni prikaz.

CSS

```
div {
    width: 100px;
    height: 100px;
    position:absolute;
    left:50px;
}
.normal{
    border: solid 2px blue;
    background:white;
}
.transform{
    border: solid 2px red;
```

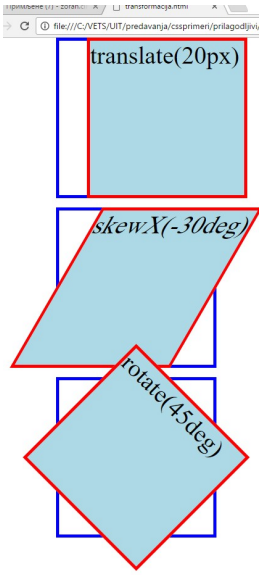
```
.red1{
    top:0px;
}
 .red2{
    top:110px;
 }
 .red3{
    top:220px;
 }
 .translate{

transform:translate(20px);
```

```
    background:lightblue;
}
```

```
}
.skew {
    transform:skewX(-30deg);
}
.rotate {
    transform:rotate(45deg);
}
```

HTML

```
<div class="normal red1">
</div>
<div class="transform red1
translate">
    translate(20px)
</div>
<div class="normal red2">
</div>
<div class="transform red2 skew">
    skewX(-30deg)
</div>
<div class="normal red3">
</div>
<div class="transform red3 rotate">
    rotate(45deg)
</div>
```



Slika 7.6. Primeri translacije, iskošenja i rotacije

Animacije

Animacija predstavlja promenu određenog stila u definisanom vremenskom intervalu. Stil se menja od početnog ka krajnjem na način koji se definiše svojstvima animacije. U toku animacije, može se menjati više CSS svojstava koja zajedno čine jedan stil, a sve te

promene se mogu ponavljati proizvoljan broj puta tokom jedne animacije.
Stilovi, koji se u pri opisivanju animacije definišu u određenim vremenskim trenucima, nazivaju se ključnim – *eng. „Keyframes“.*
Da bi se koristile animacije potrebno je definisati ključne frejmove za animaciju. Sintaksa je:

```
@keyframes naziv {vremenskaOznaka {cssStil;}}
```

gde je:
`naziv` – naziv animacije,
`vremenskaOznaka` – pozicija frejma u vremenu animacije iskazana u procentima 0-100% ili ključnim rečima `from` (0%) odnosno `to` (100%),
`cssStil` – CSS stil u definisanom trenutku.

U narednom primeru data je animacija pozadinske boje jednog odeljka i to tako da se vrši promena tri boje, od bele, preko plave do crvene.

```
div {
    width: 500px;
    height: 50px;
    background-color: red;
    animation-name: srb;
    animation-duration: 5s;
}

@keyframes srb {
    from {background-color: white;}
    50% {background-color: blue;}
    to {background-color: red;}
}
```

Animacija traje 5 sekundi. Na startu boja pozadine biće bela, za 50% vremena animacije prelazi u plavu, i za kraj, posle 5s biće crvena.

Dodatni opis animacije izvodi se svojstvima:

- `animation-delay` – vrednost ovog svojstva definiše odlaganje starta animacije u sekundama,

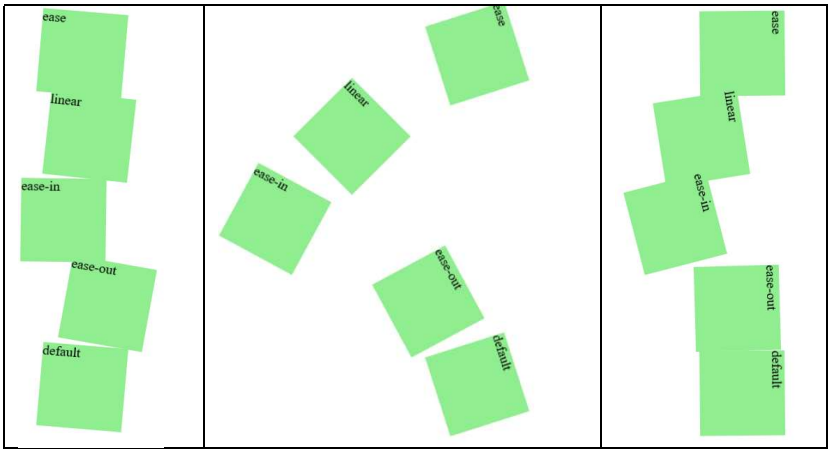
- `animation-iteration-count` – definiše broj ponavljanja animacije, ako je vrednost `infinite` onda se animacija neprekidno ponavlja,
- `animation-direction` –ukoliko ima vrednost `reverse` definiše suprotan tok animacije. Podrazumevana vrednost je `normal`. Moguća je naizmenična promena ove dve vrednosti pomoću `alternate` odnosno `alternate-reverse`. Ako je vrednost `alternate` onda se ponavlja *normal* pa *reverse*.
- `animation-timing-function` - vrednost ovog svojstva definiše brzinu i krivu promena po vremenu. Moguće vrednosti su:
  - `ease` – lagani start, sredina je brza i lagani kraj, ovo je podrazumevano svojstvo,
  - `linear` – ista brzina od početka do kraja,
  - `ease-in` – spori start,
  - `ease-out` – spori kraj,
  - `ease-in-out` – spori start i kraj,
  - `cubic-bezier`(n,n,n,n) – definisanje sopstvene krive animacije u vidu *cubic-bezier* funkcije.

CSS:

```
div {
    width: 100px;
    height: 100px;
    background-color: lightgreen;
    position: relative;
    animation: anim 5s infinite;
}
#div1 {animation-timing-function: ease;}
#div2 {animation-timing-function: linear;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
@keyframes anim {
    from {left: 0px;}
    to {left: 500px;
    transform:rotate(90deg);
}
```

HTML

```
<div id="div1">ease</div>
<div id="div2">linear</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">default</div>
```



Slika 7.7. Pozicija objekta u toku animacije

Tranzicije

Efekat tranzicija omogućava promenu jednog ili više CSS svojstava od početne do krajnje vrednosti, slično animacijama, ali bez ključnih frejmova, a efekat se izaziva pri promeni svojstva.
Dakle, da bi se kreirao efekat jedne tranzicije potrebno je najpre definisati:

1. CSS svojstva kojim se želi postići efekat promena u tranziciji,
2. Trajanje tranzicije. Ukoliko trajanje nije definisano, podrazumevana vrednost je 0.

Tranzicija se skraćeno definiše svojstvom `transition` koje se sastoji od sledećih elemenata:

- `transition-property` - naziv svojstva za tranziciju,

- `transition-duration` - vreme tranzicije u sekundama ili milisekundama,
- `transition-timing-function` - naziv funkcije promene brzine tranzicije, potpuno isto kao kod animacija,
- `transition-delay` - vreme za koje startuje tranzicija.

U sledećem primeru, nad odeljkom određenih dimenzija, izvodi se tranzicija pošto se miš nađe iznad njega. Tranzicija se definiše tako da odeljak postaje vidljiv u potpunosti za 2 sek i zelene pozadinske boje.

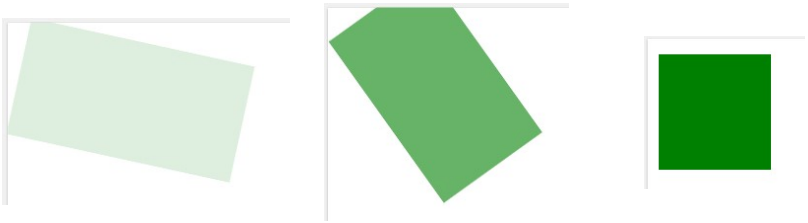
```
div {
    width: 200px;
    height: 100px;
    background: transparent;
    transition: background 2s;
}
```

```
div:hover {
    background: green;
}
```

Napomena: osim generalnog svojstva `transition` preporučuje se upotreba prefiksa za različite čitače.
Elementi tranzicije se zapisuju razdvojeni prazninom, a ukoliko se primenjuje više tranzicija onda se više njih razdvajaju zapetom. Na primer: `transition: background 2s, width 3s;` Takođe se može uključiti i neka od transformacija, na primer:

```
div {
  width:200px;
  height: 100px;
  background:transparent;
  transition:background 2s,
             width 3s,
             transform 2s;
}
```

```
div:hover {
  background: green;
  width:100px;
  transform:
         rotate(90deg);
}
```



Slika 7.8. Tranzicija uz rotaciju, promenu boje i širine

Pitanja za proveru znanja

1. Koje svojstvo se koristi za definisanje zaobljenih ivica?
2. Definiši stil kojim se iscrtava krug prečnika 100px.
3. Koje svojstvo definiše senke objekata? Napravi 3 primera senki.
4. Navedi vrste gradijenta i stil koji se koristi za definisanje.
5. Šta se postiže primenom stila: radial-gradient(red, blue, rgba(200,200,0,0.3));
6. Kako se definiše prelom teksta u više kolona?
7. Koje vrste vizuelnih transformacija znate? Za svaku napišite jedan primer.
8. Kako se definišu animacije preko stilova?
9. Svojstvo animacije animation-timing-function utiče na….?
10. Kako definišete tranzicije?
11. Navedite razliku između animacije i tranzicije. Ilustrujte to primerima.

Deo 8: Uvod u JavaScript

- Osnovna sintakasa
- Primena u dizajnu stranice
- Funkcije
- Promenljive i tipovi
- Objekti
- Nizovi
- Konverzije
- Otkrivanje grešaka
- Objekti okruženja

Ovo poglavlje je posvećeno osnovama programskog jezika JavaScript. Za razumevanje ovog poglavlja neophodno je elementarno poznavanje nekog programskog jezika, ali i poznavanje gradiva iz prethodnih odeljaka knjige. Najpre se postavljaju osnove sintakse i uvode elementarni primeri kao i način njihovog pisanja. Zatim se navode primeri tipične upotrebe sa naglaskom na praktičnu primenu.

Osnovna sintaksa

JavaScript (JS) je danas osnovni programski jezik za HTML i za Web. Ogromna popularnost Web tehnologija dovela je do primene ovog jezika i u druge svrhe. Najpopularniji je programski jezik.

Sintaksa jezika JavaScript-a je slična sintaksi C jezika, tako da zajedno sa C#, C++, JAVA-om, PHP-om i nekim drugim jezicima spada u C-olike jezike.

Klijentski je orijentisan, izvršava se na strani klijenta tj. u web čitačima. Jednostavan je. JavaScript kod se piše unutar `body` i/ili `head` sekcije i koristeći odgovarajući tag `script`. JavaScript kod se piše u vidu odgovarajućih funkcija koje se pozivaju, na primer:

```
<script>
  function myFunction() {
    document.getElementById("div1").innerHTML = "Zdravo!";
   }
</script>
```

Napomena: Svi primeri u knjizi su kompletni i testirani, pa ponekada sadrže delove koda koji će biti naknadno objašnjeni. U tom slučaju treba obratiti pažnju samo na kod koji je već objašnjen.

Zbog brzine učitavanja stranice programeri nekada ubacuju script tag na dno stranice pre `</body>` taga.
Kod se može pisati u posebnom fajlu koji se naknadno uključuje u neku `script` sekciju.

HTML:

```
<!DOCTYPE html>
<html>
<body>
    <script src="ScriptFunkcije.js"></script>
</body>
</html>
```

Sadržaj fajla ScriptFunkcije.js

```
function myFunction() {
  document.getElementById("p1").innerHTML = "Promena u JS";
}
```

Zasebni dokumenti, fajlovi, koji sadrže JavaScript kod imaju ekstenziju **`.js`**.

Primena u dizajnu stranice

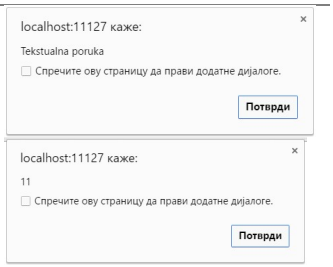
Podaci se mogu prikazati pomoću JavaScript-a na više načina:

a) **window.alert() -** koristeći pomoćni prozor za poruke,
b) **document.write() –** upisujući u HTML izlazni tok,
c) **innerHTML –** ubacujući HTML kod unutar nekog elementa,
d) **console.log() –** upisujući u konzolu čitača.

Primeri:

`window.alert()`

```
<!DOCTYPE html>
<html>
<body>
    <h1>Alerti</h1>
    <script>
        alert("Tekstualna poruka");
        window.alert(5 + 6);
    </script>
</body>
</html>
```

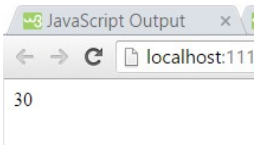


`Document.write()` – ubacivanje u postojći tok

```
<!DOCTYPE html>
<html>
<body>
    <h1>h1 naslov</h1>
    <script>
        document.write("<h2>ubačeni
element</h2>");
    </script>
    <p>Sledi ubačen rezultat
izraza</p>
    <script>
         document.write(44 - 14);
    </script>
</body>
</html>
```

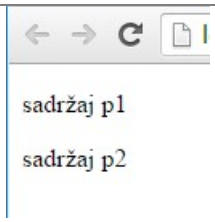
`Document.write()` – prepisivanje preko postojećeg toka

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function myFunction() {
            document.write(44 - 14)
        }
    </script>
</head>
<body>
    <h1>Naslov</h1>
    <p>paragraf</p>
    <button
onclick="myFunction()">Test</button>
</body>
</html>
```



`innerHTML`

```
<!DOCTYPE html>
<html>
<body>
    <div id="div1"></div>
    <p id="p2"></p>
    <script>

document.getElementById("div1").inner
HTML =
        "<p id='p1'>Ubaceni
paragraf</p>";

document.getElementById("p2").innerHT
ML =
        "sadržaj p2";

document.getElementById("p1").innerHT
ML =
        "sadržaj p1"; // testirajte
posle komentarisanja ove linije
```

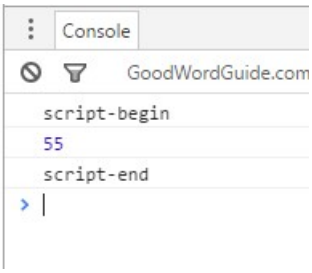


```
    </script>
</body>
</html>
```

console.log()

```
<!DOCTYPE html>
<html>
<body>
    <div id="div1"></div>
    <script>
        console.log("script-begin");
        console.log(77 - 22);
        console.log("script-end");
    </script>
</body>
</html>
```

F12 –



Napomena: Prepisivanje preko postojećeg toka događa se prateći definisan postupak upotrebe objekta document prilikom učitavanja stranice. Postupak je sledeći:

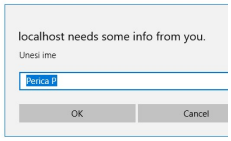
1. Prozor dokumenta se otvara za pisanje, slično upisu u fajl.
2. Čitač upisuje sadržaj u dokument. U ovom koraku se kreiraju objekti, html se priprema za prikaz
3. Prozor zatvara dokument, kao pri završetku upisivanja u fajl.

Pošto je završeno učitavanje stranice i dokument zatvoren, pri ponovnom korišćenju istog objekta funkciji dolazi do otvaranja novog praznog dokumenta u koji se vrši novi upis.

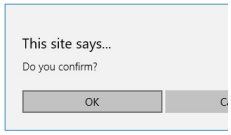
Upotreba ugrađenih prozora „prompt“ i „confirm“

Osim ugrađenog prozora „alert“ u upotrebi su: „prompt“ i „confirm“. Prozor „prompt“ omogućava unos podataka sa dva dugmeta za potvrdu ili otkazivanje unosa, dok se „confirm“ koristi za prikaz poruka takođe sa mogućnošću potvrde ili otkazivanja. Na primer:

```
function myFunction() {
   var ime = prompt("Unesi ime", "Perica P");
   if (ime != null){
     document.getElementById("test").innerHTML =
     "Zdravo " + ime + "!";
      }
   }
}
```




```
function myFunction() {
  var confirmResult = confirm('Do you confirm?');
  document.getElementById("test").innerHTML =
  confirmResult;
}
```



Funkcije

JavaScript funkcija je blok JS naredbi tj. kod koji se poziva kao jedna celina. To se izvodi navođenjem naziva funkcije. Definisanje funkcije odnosno poziv postojeće funkcije je definisano sintaksom programskog jezika. JS koristi ključnu reč **function** za definisanje funkcije, a poziv se izvodi navođenjem imena funkcije.

Na primer, neka funkcija se može izvršavati samo pošto se klikne na neko dugme. Pošto trenutak pritiska na dugme nije unapred predvidiv, definiše se događaj odnosno funkcija za taj događaj. Konkretno događaj pritiska (tj. klik) na dugme se naziva **click**.

```
<!DOCTYPE html>
<html>
<head>
  <script>
     function myFunction() {
       document.getElementById("p1").innerHTML = "Tekst
nakon klika.";
     }
  </script>
</head>

<body>
  <p id="p1">Početni tekst</p>
```

Definisanje funkcije

Upotreba funkcije

```
  <button onclick="myFunction()">Promeni tekst</button>
</body>
</html>
```

Ukoliko funkcija vraća vrednost, koristi se ključna reč `return`, na primer:

```
function addFunction(x, y){ return x + y; };
var c = addFunction(5, 10);
```

Promenljive i tipovi

Promenljive se koriste da bi se neka vrednost čuvala odnosno koristila u JS kodu. Promenljiva je definisana za određene podatke odnosno definisana kao određeni tip. Tipovi promenljivih, ali i svih podataka u JavaScripta su:

- String – niz karaktera, tekst. Zapisuje se između jednostrukih ili dvostrukih znakova navoda. Moguće je koristiti apostrof u stringu sa navodnicima i obratno. Specijalni znakovi su:
  `\'` – apostrof,
  `\"` – navodnici,
  `\\` - obrnuta kosa crta,
  `\n` - novi red,
  `\r` - povratak na početak reda,
  `\t` – tab,
  `\b` – backspace,
  `\f` - nova strana.

Neke funkcije i svojstva za stringove su:

- `length` - dužina stringa,
- `charAt()` - vraća znak sa određene pozicije u stringu, počev od nule,
- `concat()` - spaja dva ili više stringova, npr:
  str2 = str0.concat(" ",str1);
  str = "Zdravo" + " " + "svete!";
  var str = " Zdravo ".concat(" ","svete!");

- `indexOf()` - vraća indeks, tj. poziciju, prvog pojavljivanja nekog teksta u stringu ili -1 ako ne postoji,
- `lastIndexOf()` - vraća indeks, tj. poziciju, poslednjeg pojavljivanja zadatog teksta u stringu,
- `match()` - vraća niz pogodaka za pretragu stringa korišćenjem regularnog izraza ili null ako nema pogodaka,
- `replace()` - zamenjuje prvo pojavljivanje traženog teksta u stringu drugim tekstom,
- `search()` - kao i indexOf(), pretražuje pojavljivanje zadatog stringa i vraća njegovu poziciju, ali ima mogućnost korišćenja regularnih izraza,
- `slice()` - vraća deo stringa između zadatih pozicija,
- `substr()` - slično kao slice() samo što je drugi parametar dužina vraćenog stringa i on ne može biti negativan,
- `split()` - deli string na osnovu zadatog stringa i pravi niz podstringova.

- Number – tip za brojeve. Zapisuje se u formatu pokretnog zareza sa dvostrukom preciznošću - 64 bita. Ovo je jedini tip za brojeve. Brojevi se mogu zapisivati sa ili bez decimala kao i u eksponencijalnom zapisu: tmp = 2.68; tmp = 365; tmp = 1212e4; x = 1.11e-5; Što se tiče preciznosti celi brojevi su zapisani na 15 cifara, decimalni do 17 decimala. Heksadecimalne konstante se pišu sa 0x na početku - 0xbeca. Generalno, za prikaz u binarnom, oktalnom ili heksadecimalnom brojnom sistemu može se koristiti funkcija toString() čiji argument je osnova brojnog sistema:

var a = Number(15).toString();	15
var b = Number(15).toString(2);	1111
var num = 14;	
var c = num.toString(8);	16
var d = num.toString(16);	e

Slične metode su:

- `toExponential(brojDecimala)`; - vraća string od zaokruženog broja na određeni broj decimala i u eksponencijalnom zapisu,

- `toFixed()` - vraća string od zaokruženog broja i predstavljenog sa određenim brojem decimalnih mesta,
- `toPrecision()` - vraća string na osnovu broja cifara i bez decimalne tačke.
  Specifične vrednosti su:
- `Infinity` odnosno –Infinity, predstavljaju beskonačne vrednosti a takođe su tipa Number.
- `NaN` (eng. Not a Number) – vrednost koja znači da promenljiva tipa Number nije broj. Treba zapamtiti i funkciju: Number.isNaN(), koja provera da li je promenljiva broj ili ne. Neke metode vezane za brojeve su:
- `Number()` - Vraća broj ili NaN,
- `parseFloat()` - Parsira argument i vraća broj u pokretnom zarezu ili NaN,
  - `parseInt()` - Parsira argument i vraća ceo broj ili NaN.

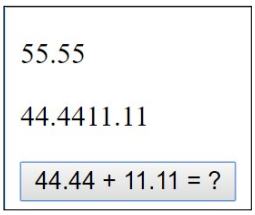
- Boolean – vraća jednu od dve vrednosti: `true` ili `false`. Ovaj tip se vraća kao rezultat logičkih izraza, na primer: (10 > 11), (a < b), (x == y), ... Svaka realna vrednost promenljive može se pretvoriti u logičku vrednost pomoću Boolean izraza, na primer: Boolean("Hello"); Boolean("false"); Boolean(-15);.... Za sve ove izraze logička vrednost je true.
- Array – tip objekata za niz. Ovom tipu smo posvetili posebno poglavlje. Radi se o nizu vrednosti koje čuva jedna promenljiva, na primer: var smer = ["NRT", "RT", "EPO"];
- Object – u jeziku JavaScript sve promenljive su objekti osim prostih tipova. Prosti tipovi su: string, number, boolean, null i undefined. Čak se string, boolean i number mogu kreirati i koristiti kao objekti. Jedan primer definisanog objekta je:
  var osoba={ime:"Perica", prezime:"P", starost:50};
- Null – predstavlja vrednost objekta koji nije inicijalizova tj. nepostojeću vrednost. Na primer, ako promenljiva osoba postavljena kao objekat u nekom trenutku dobije vrednost null, ona ostaje objekat i dalje. Na primer: var osoba={ime:"Perica", prezime:"P", starost:50}; osoba=null; osoba ostaje objekat sa vrednošću null. Takođe, ako se izvrši inicijalizacija promenljive var osoba=null; promenljiva je implicitno tipa object sa vrednošću null.

- Undefined – ova vrednost se u JavaScriptu dodeljuje promenljivoj koja nije inicijalizovana (ili joj je eksplicitno dodeljena vrednost undefined) , na primer: var osoba; osoba dobija vrednost undefined implicitno. Ovako definisana promenljiva ima vrednost i tip undefined.

Deklarisanje promenljivih

Sve promenljive se deklarišu pomoću ključne reči **var**. Promenljive imaju i svoju stringovsku reprezentaciju. To znači da, ukoliko promenljiva nije tipa string, već na primer broj, ta promenljiva se može konvertovati u string. Konverzija se može desiti i bez eksplicitnog zahteva (implicitna konverzija). Na primer:

```
<!DOCTYPE html>
<html>
<head>
 <script>
  function myFunction() {
    var x = 44.44;
    var y = 11.11;
    document.getElementById("p1").innerHTML = x + y;
    document.getElementById("p2").innerHTML = "44.44" + y;
  }
  </script>
</head>
<body>
  <p id="p1">44.44 + 11.11 = </p>
  <p id="p2">"44.44" + 11.11 = </p>
  <button onclick="myFunction()">44.44 + 11.11 = ?
  </button>
</body>
</html>
```



Tip promenljive možemo prikazati primenom metode **typeof(.)**

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

```
<head>
  <script>
    function myFunction() {
      var x = 44.44;
      var y = new Date(2016,5,21);
      var z = true;
      var o = document.getElementById("p1");
      document.getElementById("p1").innerHTML = "typeof
      (44.44), typeof (new Date(2016,5,21)), typeof(true),
typeof (<p1>)";
      document.getElementById("p2").innerHTML = typeof (x)
       + ", " + typeof (y) + ", " + typeof (z) + ", "
       + typeof (o);
    }
  </script>
</head>
<body>
  <p id="p1">...</p><p id="p2">...</p>
  <button onclick="myFunction()">typeof() = ?</button>
</body>
</html>
```

typeof(44.44), typeof (new Date(2016,5,21)), typeof(true), typeof ()

number, object, boolean, object

typeof() = ?

Provera tipa objekta se izvodi primenom metode **instaceof**. Na primer:

```
<!DOCTYPE html>
<html>
<head>
  <script>
  function myFunction() {
    var tmp = 11;
    document.getElementById("p1").innerHTML =
        "" + tmp + " instanceof Number = " +
        (tmp instanceof Number);
    var tmp = "viser";
    document.getElementById("p2").innerHTML =
        "" + tmp + " instanceof String = " +
        (tmp instanceof String);
    var tmp = true;
```

11 instanceof Number = false

viser instanceof String = false

true instanceof Boolean = false

22.6.2016. instanceof String = false

instanceof

```
    document.getElementById("p3").innerHTML =
        "" + tmp + " instanceof Boolean = " +
        (tmp instanceof Boolean);
    var tmp = new Date(2016, 5, 22);
    document.getElementById("p4").innerHTML =
        "" + tmp.toLocaleDateString() + "
instanceof String = " + (tmp instanceof
String);
    }
    </script>
</head>
<body>
    <p id="p1">...</p>
    <p id="p2">...</p>
    <p id="p3">...</p>
    <p id="p4">...</p>
    <button
onclick="myFunction()">instanceof</button>
</body>
</html>
```

Obratite pažnju na dobijeni rezultat.
Promenljive tipa Number, String i Boolean koje su u primeru korišćene kao primitivne mogu biti definisane i kao objekti. Promenite definicije promenljivih u kodu u objekte, pa proverite ishod primera, na primer:

```
var tmp = new Number(11);
var tmp = new String("asdf");
var tmp = new Boolean(true);
```

Oblast važenja (dostupnosti)

Oblast važenja promenljive je deo koda u kome važi definisana promenljiva odnosno deo koda u kome se može pristupiti istoj promenljivoj. Van oblasti važenja, promenljiva može biti skladištena na neki način, ali joj nije moguće pristupiti po imenu. U JavaScript-u postoje dva osnova tipa promenljive, odnosno oblasti važenja: **lokalni** i **globalni**.

Globalna promenljiva je dostupna bilo gde u programu. Ona se definiše van funkcija. Promenljiva prestaje da važi po zatvaranju stranice.
Lokalna promenljiva je dostupna samo unutar funkcije u kojoj je definisana. Lokalna oblast važenja naziva se još i oblast važenja funkcije.

```
// Globalna var.
var pocetak = "Globalna ";

// Lokalna var. in fja1
function fja1() {
    var pocetak = "Lokalna ";
}

pocetak += "promenljiva. Nastavak zapocetog teksta";

document.write(pocetak); // Globalna promenljiva. Nastavak
zapocetog teksta
```

Ovde treba obratiti pažnju da postoji bitna razlika u odnosu na neke druge programske jezike u kojima lokalna oblast važenja nastupa sa blokom vitičastih zagrada. U JavaScriptu cela funkcija predstavlja jednu oblast važenja pa promenljiva deklarisana bilo gde u funkciji ima oblast važenja u celoj funkciji. Da bi se doprinelo jasnoći koda obično se promenljive deklarišu na početku funkcije.

```
function fja1() {
    if (true)
    {
        var pocetak = "pocetak";
    }
    var novipocetak = "Novi " + pocetak;
    return novipocetak;
}
document.write(fja1());
```

Funkcije mogu sadržati ne samo promenljive već i druge funkcije – ugnježdene funkcije. Oblast važenja ugnježdenih funkcija je takođe ugnježdena. Na primer:

```
function myFunction() {
    var povrsina =
document.getElementById("povrsinaZida").value;
    var dimX = document.getElementById("dimXcm").value;
    var dimY = document.getElementById("dimYcm").value;
    alert("Potrebno je kupiti: " + brojPločica(dimX, dimY,
povrsina));
```

```
}
function brojPločica(plocica_dimXcm, plocica_dimYcm,
povrsinaZida) {
    return povrsinaZida * 1.05 /
povrsinaPlocice(plocica_dimXcm, plocica_dimYcm);
    function povrsinaPlocice(plocica_dimXcm,
plocica_dimYcm) {
        return plocica_dimXcm * plocica_dimYcm * 0.0001;
    }
}
```

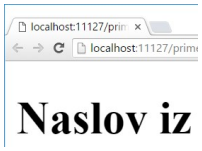
Promene na elementima i atributima

JS poseduje metodu **`getElementById()`** koja omogućava preuzimanje kontrole nad određenim elementom HTML dokumenta i rad sa njim. Element se identifikuje pomoću atributa **`id`**. Ono što ova metoda vraća je objekat sa pratećim svojstvima, pomoću koji se može izvesi nekoliko tipičnih operacija:

1. promena sadržaja
2. promena atributa
3. promena važećeg stila
4. provera ispravnosti sadržaja (validacija)

Promena sadržaja

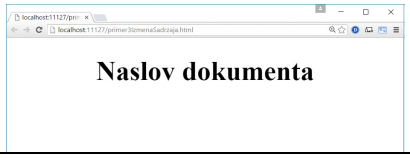
```
<!DOCTYPE html>
<html>
<body>
  <h1 id="naslov">Naslov dokumenta</h1>
  <script>
    document.getElementById('naslov')
            .innerHTML = 'Naslov iz JS!';
  </script>
</body>
</html>
```



Promena atributa

Pošto se element identifikuje, atributu se pristupa imenom atributa koji sledi nakon tačke. Dakle, ime atributa je svojstvo vraćenog objekta.

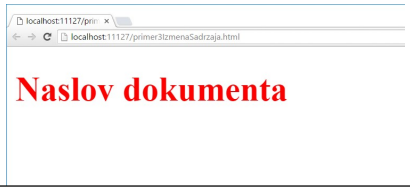
```
<!DOCTYPE html>
<html>
<body>
  <h1 id="naslov">Naslov dokumenta</h1>
  <script>
    document.getElementById('naslov').align = 'center';
  </script>
</body>
</html>
```



`Promena tj. definisanje stila`

Koristi se ključna reč `style` nakon koje sledi stil koji se primenjuje. Pristupanje stilu nekog elementa slično je atributu tj. radi se o svojstvu objekta elementa, a pristup određenom stilu se vrši kao svojstvo objekta stila.

```
<!DOCTYPE html>
<html>
<body>
    <h1 id="naslov">Naslov dokumenta</h1>
    <script>
    document.getElementById('naslov').style.color
        = 'red';
    </script>
</body>
</html>
```

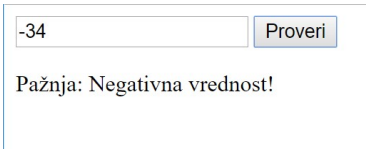


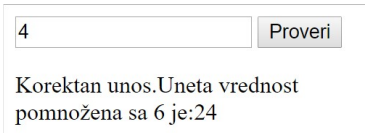
`Validacija podataka`

Validacija podataka znači proveru ispravnosti unetih podataka. Validacija se odnosi na HTML elemente koji omogućavaju unos podataka i primenjuju se na HTML formama. JavaScript ima jaku

podršku za validaciju podataka. Na ovom mestu upoznaćemo se sa osnovnim pristupom.
Vrednost elementa za unos podataka dobija se preko svojstva `value`. Isto svojstvo može se koristiti i pri izmeni vrednosti.

```
<!DOCTYPE html>
<html>
<body>
  <input id="arg1" placeholder="unesi vrednost" />
  <button type="button" onclick="proveriArg1()">
  Proveri</button> <br/>
  <p id="poruka">...</p>
  <script>
   function proveriArg1() {
     console.log("provera");
     var ctrPoruka = document.getElementById("poruka");
     var temp = document.getElementById('arg1').value;
      if (isNaN(temp))
        ctrPoruka.innerHTML = "Greška: Nije broj!";
      else if (temp < 0)
        ctrPoruka.innerHTML = "Pažnja: Negativna
vrednost!";
      else
        ctrPoruka.innerHTML = "Korektan unos." +
        "Uneta vrednost pomnožena sa 6 je:" +
                                        temp * 6;
    }
  </script>
</body>
</html>
```





Funkcije kao vrednosti

Funkcija se može samostalno koristiti i kao promenljiva i tako prosleđivati drugim funkcijama ili svojstvima nekog objekta, na primer:

```
function knjiga(naslov, autor, fjacene) {
    this.naslov = naslov;
    this.autor = autor;
    this.racunajCenu = fjacene;
}
function knjigaCena(osnovica)
{
    this.cena = osnovica*0.95;
}
var primerak1 = new knjiga("Pesme", "Ršumović",
knjigaCena);

primerak1.racunajCenu(300);
```

Funkcija može biti imenovana ili anonimna.

```
var x = function (osnovica) {   // anonimna
    this.cena = osnovica * 0.95;
}
ili
var x = (osnovica) => this.cena = osnovica * 0.95; //
anonimna

var x1 = function cenaOdOsnovice_Imenovana(osnovica) {
    this.cena = osnovica * 0.95;
}
x(100);
console.log(cena);
x1(200);
console.log(cena);
```

Anonimna funkcija nema ime, ali se u praksi često koristi. Na primer, može se pridružiti nekoj promenljivoj koja ima osobine promenljive,

ali istovremeno se koristi kao funkcija, kao argument za drugu funkciju ili samo za dobijanje vrednosti koju funkcija vraća.
Funkcija se može pisati tako da se njeno izvršavanje izvede odmah.

```
(function () {
    console.log("dobijena cena je: " + cena);
})();
```

ili

```
var cena = 999.9;
( () => console.log("dobijena cena je: " + cena) )();
```

Deklaracija `let` odnosno `var`

Razlika promenljivih koje su deklarisane ključnom reči `var` u odnosu na one koje su deklarisane sa `let` je u oblasti važenja.

Promenljiva deklarisana sa var ima oblast važenja u celoj funkciji, a dostupna je u kodu i pre same deklaracije.

Promenljive deklarisane sa let nisu dostupne pre nego što se deklarišu a oblast važenja je blok koda označen vitičastim zagradama.

Oba su globalna ako su van bilo kog bloka.

```
function allyIlliterate() {
    //vLet NIJE vidljiv
    for (let vLet = 0; vLet < 5; vLet++) {
        //vLet JE vidljiv
    }
    //vLet is *not* visible out here
}
function byE40() {
    //vGlb JE vidljiv
    for (var vGlb = 0; vGlb < 5; vGlb++) {
        //vGlb JE vidljiv
    }
    //vGlb JE vidljiv
}
```

Objekti

Ponovimo još jednom. U JavaScript jeziku sve promenljive su objekti osim prostih tipova. Prosti tipovi su: string, number, boolean, null i undefined. Čak se string, boolean i number mogu kreirati i koristiti kao objekti: `var tmp=new String();`

Jedan objekat može da sadrži više promenljivih koje same mogu biti prostog tipa ili objekti. Promenljive u objektu imaju svoj naziv, a odgovarajuća vrednost se pridružuje koristeći naziv. Kaže da objekti predstavljaju kolekciju imenovanih vrednosti ili parova `name:value`. Ovi parovi se nazivaju još i svojstvima objekata. Osim svojstava objekti mogu da sadrže i metode tj. pripadajuće funkcije. Na primer:

```
<!DOCTYPE html>
<html>
<body>
    podaci o osobi:
    <p id="osoba1"></p>
  <script>
    var osoba = {
        ime: "Perica",
        prezime : "P",
        id: 666,
        // this!!!
        toString : function() {
           return this.prezime + " " + this.ime;
        }
    };
    document.getElementById("osoba1").innerHTML =
osoba.toString();
  </script>
</body>
</html>
```

Zapazite da se pristup elementima objekta iz metode u objektu obavlja pomoću ključne reči `this`. Ključna reč `this` označava tekući (nadređeni) objekat. Ako se ključna reč `this` koristi u objektu označava taj objekat, ako se koristi u funkciji označava nadređeni

objekat funkcije. U prethodnom primeru objekat `this` je iskorišćen za pristup svojstvima objekta iz jedne funkcije istog objekta.

Pristup do vrednosti jednog svojstva vrši se pomoću imena svojstva: `obj[imeSvojstva]` ili `obj.imeSvojstva`. Pristup do svih svojstava može da se izvrši pomoću petlje `for..in`:
`for(var svojstvo in objekat){….}`.

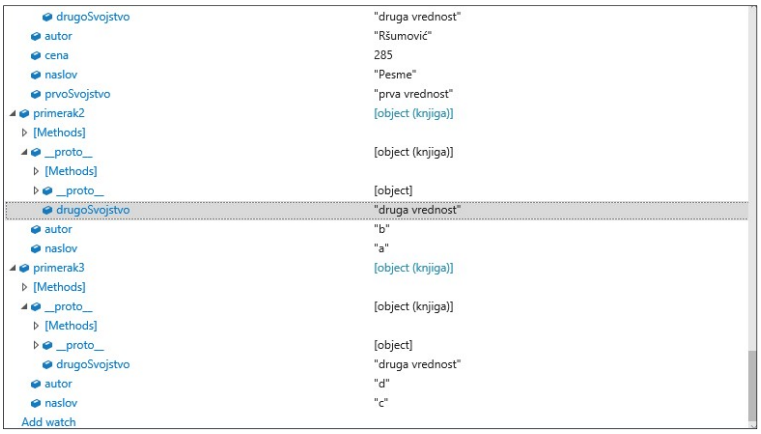
Prototip - objekat delegiranja ponašanja

Svaki JavaScript objekat ima posebno svojstvo koje se naziva prototip. Prototip je sam objekat. JavaScript objekti dobijaju svojstva i metode od svojih prototipova. Na vrhu lanca prototipova se nalazi `Object.prototype` od koga svi nasleđuju.
Pri kreiranju jednog objekta, koristeći rezervisanu reč `new`, formira se odgovarajući prototip, na primer:

```
function knjiga(naslov, autor) {
    this.naslov = naslov;
    this.autor = autor;
}
var primerak1 = new knjiga("Pesme", "Ršumović");
```

Svojstvo `prototype` omogućava dodavanje novog svojstva već postojećem tipu.

```
var primerak1 = new knjiga("Pesme", "Ršumović");
primerak1.prvoSvojstvo = "prva vrednost";
var primerak2 = new knjiga("a", "b");
knjiga.prototype.drugoSvojstvo = "druga vrednost";
// nakon promene prototipa promenjena su i svojstva tipa
„knjiga“
// to vazi za primerak3, ali i za primerak2 i primerak1
// Menjaju se samo korisnički prototipovi, a ne ugrađeni.
var primerak3 = new knjiga("c", "d");
var tmp3 = primerak3.drugoSvojstvo;
```



Slika 8.1. Prikaz vrednosti iz „debuger-a“

Funkcija eval

JavaScript može da obradi zadati string kao sopstveni kod. Ovo se obavlja pomoću funkcije eval. Na primer:

```
function testEval() {
    var a = 12, b = 4;
    console.log(eval("a - b"));
    console.log(eval("a * b"));
}
```

Nizovi

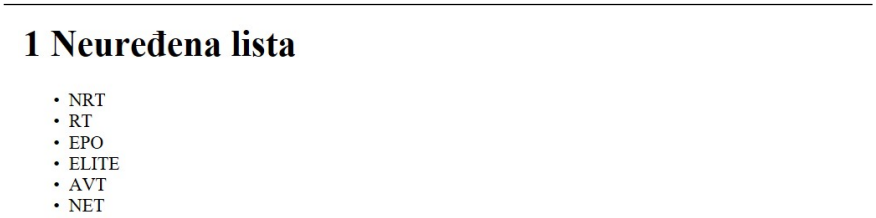
Array predstavlja tip objekta koji je niz u JavaScript-u. Niz se definiše pomoću uglastih zagrada. Koristeći niz, umesto da se definiše za svaku vrednost jedna promenljiva, može se definisati jedna promenljiva koja je niz, a koja omogućava rad sa više vrednosti. Ovaj tip podataka ima svoje osobine tj. svojstva i pripadajuće funkcije koje se mogu koristiti. Neke od njih su:

`constructor`	Vraća funkciju koja kreira Array - konstruktor
`length`	Broj elemenata niza
`prototype`	Dozvoljava dodavanje svojstava i metoda već formiranom tipu objekta

concat()	Spajanje dva ili više nizova
find()	Vraća vrednost prvog elementa u nizu koji zadovoljava uslov
findIndex()	Vraća indeks prvog elementa u nizu koji zadovoljava uslov
indexOf()	Pretražuje niz za neki element i vraća njegovu poziciju
sort()	Sortira elemente niza

Na primer:

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

    <h1>1 Neuređena lista</h1>
    <p id="Smerovi1"></p>

    <script>
        var smer = viserSmerovi(false);

        html1 = "<ul>";
        for (i = 0; i < smer.length; i++) {
            html1 += "<li>" + smer[i] + "</li>";
        }
        html1 += "</ul>";
        document.getElementById("Smerovi1").innerHTML = html1;

        function viserSmerovi(sort)
        {
            smer = ["NRT","RT","EPO","ELITE","AVT","NET"];
            if(sort)
            {
                smer = smer.sort();
            }
            return smer;
        }
    </script>

</body>
</html>
```



1 Neuređena lista

- NRT
- RT
- EPO
- ELITE
- AVT
- NET

Primer niza sa prekidom i preskokom. Takođe primer sadrži i petlju po svojstvima jednog objekta odnosno upotrebu operatora for...in.

```
<!DOCTYPE html>
<head>
<style>
div{
width:300px;
height:300px;
background-color:lightblue;
}
ul, ol{
margin:0;
}
</style>
</head>
<html>
<body>
<a href="#"
id="link1">testiranje rada sa
nizom sa preskokom i
prekidom</a>
<div id="rez1">for petlja kroz niz</div>
<div id="rez2">for in petlja</div>
<script>
var niz = ["NRT", "RT", true, 54, 666, 77];
var mojObjekat = {
ime: "Perica",
prezime: "P",
jmbg: 1231231231231,
datum: new Date(2017,12,31)
};
var link1 = document.getElementById("link1");
var rez1 = document.getElementById("rez1");
var rez2 = document.getElementById("rez2");
link1.addEventListener("click", function () {
var htmlRez = "<ul>";
```

testiranje rada sa nizom sa preskokom i prekidom

- NRT
- RT
- 54

1. ime : Perica
2. prezime : P
3. jmbg : 1231231231231
4. datum : Wed Jan 31 2018 00:00:00 GMT+0100 (Central Europe Standard Time)

```
for (var i = 0; i < niz.length; i++)
{
if (i == 2) continue;
if (i == 4) break;
debugger;
htmlRez += "<li>" + niz[i] + "</li>";
}
htmlRez += "</ul>";
rez1.innerHTML = htmlRez;

var htmlRez = "<ol>";
for (var s in mojObjekat) {
debugger;
htmlRez += "<li>" + s + " : " + mojObjekat[s] + "</li>";
}
htmlRez += "</ol>";
rez2.innerHTML = htmlRez;
});
</script>
</body>
</html>
```

Konverzije

Eksplicitna konverzija se obavlja primenom metoda za konverziju:

- `Number()` – konverzija u broj,
- `String()` – konverzija u string,
- `Boolean()` – konverzija u Boolean.

Tabela konverzije:

Vrednost/val	Number(val)	String(val)	Boolean(val)
false	0	"false"	false
true	1	"true"	true
0	0	"0"	false
1	1	"1"	true
"0"	0	"0"	true
"1"	1	"1"	true
NaN	NaN	"NaN"	false
Infinity	Infinity	"Infinity"	true
-Infinity	-Infinity	"-Infinity"	true

""	**0**	""	**false**
"20"	20	"20"	true
"twenty"	NaN	"twenty"	true
[ ]	**0**	""	true
[20]	**20**	"20"	true
[10,20]	NaN	"10,20"	true
["twenty"]	NaN	"twenty"	true
["ten","twenty"]	NaN	"ten,twenty"	true
function(){}	NaN	"function(){}"	true
{ }	NaN	"[object Object]"	true
null	**0**	"null"	false
undefined	NaN	"undefined"	false

Otkrivanje grešaka

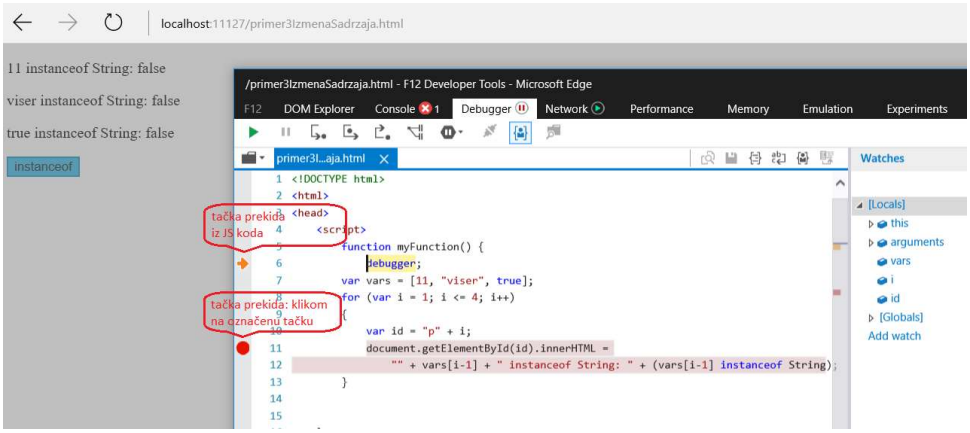
Otkrivanje grešaka u kodu naziva se i debagovanje. Novi web čitači imaju ugrađene mehanizme za debagovanje koji uključuju zaustavljanje izvršavanja na određenoj tački koda kao i izvršavanje liniju po liniju koda. Ovi mehanizmi se nazivaju još i debageri. Debageri mogu biti uključeni ili isključeni u toku izvršavanja koda. Kada su uključeni vrši se prikazivanje svih informacija o toku izvršavanja naročito vezanih za pojavu grešaka.

Pomoću debagera može se postaviti jedna ili više tačaka prekida (eng. *break point*) u kojima se program zaustavlja izvršavanje i u kojima se mogu pratiti stanja promenljivih. Rad pomoću debagera se ostvaruje prebacivanjem web čitača u poseban režim prikaza namenjen razvoju, a obično se može startovati skraćenicom sa tastature „F12“ .

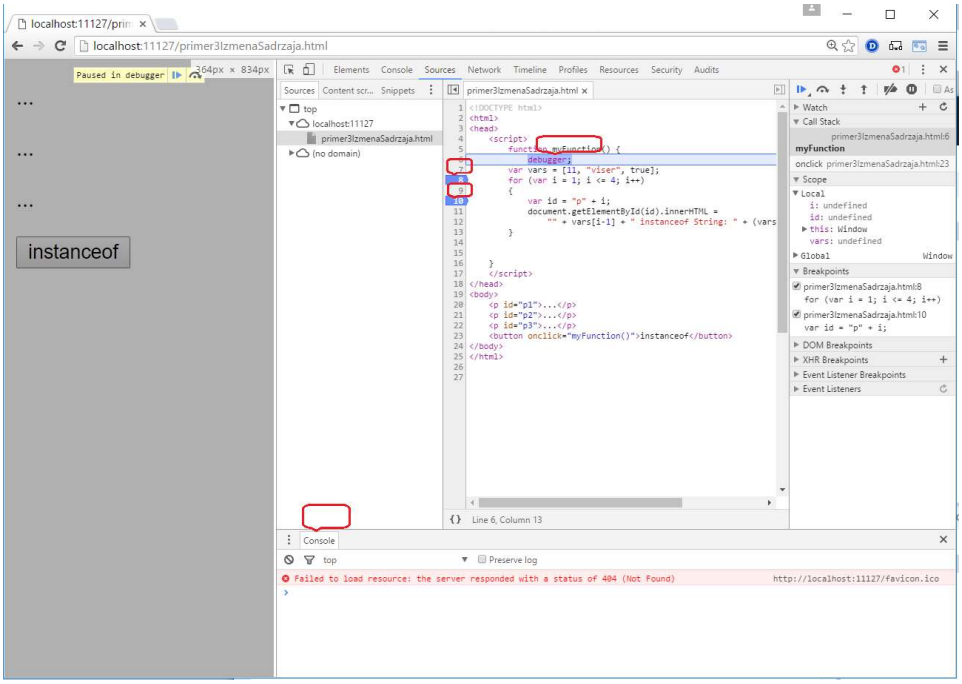
Jedan od načina za ispis posebnih poruka u prozor **`console`** u toku debagovanja izvodi se primenom metode: **`console.log()`**.

Postavljanje tačaka prekida se izvodi u JS kodu pomoću komande **`debugger`**. Takođe, tačke prekida se mogu postaviti i direktno u prozoru debagera. Obično je dovoljan klik na liniju gde inače stoji

oznaka za postavljenu tačku prekida ili nekom skraćenicom sa tastature.



Slika 8.2. Postavljanje tačaka prekida (eng. Break points)



Slika 8.3. Praćenje tekućih vrednosti i stanja

Objekti okruženja

Kada se JavaScript kod izvršava kod može pristupiti određenom sadržaju tj. izvršava se u određenom okruženju. Okruženje u kom se kod izvršava je okruženje nekog web čitača i obuhvata pristup sadržaju tekuće stranice, sadržajima otvorenih stranica ako ih ima više i na kraju pristup web čitaču.
Objekat window predstavlja jedan od objekata JavaScript-a koji su deo web čitača. Zajedno sa objektima screen, location, history čini BOM (eng. Browser Object Model) model. Neke važne metode i svojstva objekta window su:

- window.open() - otvara novi prozor,
- window.close() - zatvara tekući prozor,
- window.moveTo() - pomera tekući prozor,
- window.resizeTo() - menja veličinu tekućem prozoru,
- window.screen - predstavlja korisnikov ekran. Može se pisati bez prefiksa window, a neka njegova svojstva su:
  - screen.width - širina ekrana u pikselima,
  - screen.height - visina ekrana u pikselima,
  - screen.availWidth - širina ekrana u pikselima raspoloživa za kontrolu prikaza, npr. bez linije sa alatkama (taskbar),
  - screen.availHeight - visina ekrana u pikselima umanjena za delove interfejsa (npr. taskbar).
- window.print() – Metod koji obezbeđuje štampanje sadržaja tekućeg prozora.

Osim objekta window koji predstavlja prvi i najviši objekat u web čitaču važni objekti su:

- document – obuhvata tekući prikaz u web čitaču. Pristup ovom objektu znači pristup do svih elemenata tekućeg prikaza sa mogućnošću da se upravlja sa istim. Više detalja o metodama ovog objekta biće u poglavlju o DOM modelu. Neke metode su:
  - document.body – pristup body elementu,
  - document.forms - vraća sve <form> elemente,
  - document.head - vraća <head> element,

  - document.images - vraća sve <image> elemente.
- history – sadrži listu svih sajtova u istoriji web čitača. Omogućava kretanje unapred ili unazad po toj listi. Objekat history se može pisati bez prefiksa window (window.history isto je što i history). Zbog zaštite privatnosti korisnika postoje ograničenja kako JavaScript pristupa ovom objektu. Osnovne metode:
  - history.back() - Isto kao klik na dugme back/nazad u pregledaču,
  - history.forward() - Isto kao klik na dugme forward/napred u pregledaču.
- location – čuva informacije o poziciji dokumenta koji se prikazuje. Lokacija obuhvata URL adresu kao i protokol, domen, putanju do fajla i port. Objekat location se može pisati bez prefiksa window. Osnovna svojstva su:
  - location.href - vraća href (URL) tekuće stranice,
  - location.hostname - ime domena,
  - location.pathname - Vraća putanju (path) i ime datoteke tekuće stranice,
  - location.protocol - web protokol koji se koristi (http:// ili https://).

DOM model

DOM model je skraćenica od engleskog naziva *Document Object Model*. U ovom modelu HTML dokument sa svim svojim sadržajima je predstavljen u obliku čvorova. Čvorovi su elementi, atributi, sadržaji, komentari. Čvorovi su hijerarhijski organizovani.

document

Korenski element je jedini element koji nema roditeljski element u hijerarhijskoj strukturi. U ovom modelu to je **`document`** objekat.

Objekat **`document`** sadrži svojstva i metode za pristup svim ostalim čvorovima preko JavaScript-a. Izuzetak: Dokument se prikazuje u prozoru čitača, pa se jedino sam objekat prozora čitača, **`window`**, može koristiti iznad dokumenta.

Do sada smo često koristili metodu **`getElementById()`** koja pripada dokumentu. U tabeli su navedene još neke često korišćene metode odnosno svojstva.

Svojstvo / Metod	Opis
`document.activeElement`	tekući element koji je u fokusu/aktivan,
`document.body`	**`body`** element,
`document.cookie`	parovi name/value svih kolačića u dokumentu,
`document.createAttribute()`	kreira jedan atribut,
`document.createComment()`	kreira komentar,
`document.createElement()`	kreira element,
`document.forms`	vraća kolekciju svih <form> elemenata,
`document.getElementById()`	element definisanog ID,
`document.getElementsByClassName()`	lista čvorova elemenata definisane klase,
`document.getElementsByName()`	vraća listu čvorova definisanog imena,
`document.getElementsByTagName()`	vraća listu čvorova definisanog tag naziva,
`document.hasFocus()`	vraća da li je dokument u fokusu tj. aktivan,
`document.head`	vraća element **`<head>`**,
`document.images`	vraća kolekciju svih **`<img>`** elemenata,
`document.links`	vraća kolekciju svih **`<a>`** i **`<area>`** elemenata koji imaju **`href`**,
`document.open()`	otvara HTML izlazni tok za prikupljanje sadržaja od **`document.write()`**,
`document.querySelector()`	vraća prvi element koji odgovara specifičnom **`CSS`** selektoru,

document.scripts	vraća kolekciju **<script>** elemenata,
document.title	vraća naslov,
document.URL	vraća puni **URL** tekućeg HTML dokumenta,
document.write()	piše HTML izraze ili JavaScript kod u dokument,
document.writeln()	isto kao write() samo dodaje novi red na kraju.

Pitanja za proveru znanja

1. Kako se JavaScript kod pridružuje HTML/CSS kodu?
2. Navedi načine pomoću kojih JavaScript utiče na prikaz HTML stranice? Napišite primer za svaki.
3. Šta su, i kako se koriste, „prompt“ i „confirm“ prozori?
4. Navedi dva primera JavaScript funkcije.
5. Šta su to promenljive i koji tipovi promenljivih u JavaScript-u postoje?
6. Navedi po dve ugrađene funkcije za rad sa promenljivima koje su tipa String odnosno Number.
7. Kako se definišu promenljive i koja je oblast njihovog važenja?
8. Kako se, primenom JavaScript-a, može promeniti/dodati HTML sadržaj elementa?
9. Kako se, primenom JavaScript-a, može promeniti/dodati vrednost atributa nekog HTML elementa?
10. Kako se, primenom JavaScript-a, može promeniti/dodati stil za neki element?

11. Da li se funkcija može koristiti kao vrednost tj. neka promenljiva?
12. Koja dva načina za deklarisanje promenljivih postoje i koje su razlike tako deklarisanih promenljivih?
13. Šta su objekti u JavaScript-u?
14. Kako se definišu objekti? Navedi primer.
15. Šta se označava ključnom reči *this*?
16. Šta su to prototipovi objekata?
17. Kako se deklarišu i koriste nizovi u JavaScript-u?
18. Koje vrste konverzija podataka postoje i kako se obavljaju?
19. Koje objekte okruženja poznajete? Navedite po 2 funkcije za svaki objekat.
20. Šta je document objekat? Navedi 5 funkcija ovog objekta.

Literatura

1. Laura Lemay, Rafe Colburn, Jennifer Kyrnin, “HTML5, CSS3 i JavaScript za razvoj veb strana“, Mikro knjiga, 2016.
2. J. D. Gauchat, „HTML5, CSS3 i JavaScript: Integrisane tehnologije za izradu veb strana“, Mikro knjiga, 2014
3. Christopher Schmitt, Kyle Simpson, “HTML5 Cookbook“, O’Reilly Media, 2012.
4. Faithe Wempen, “HTML5 Step by Step“, O’Reilly Media, 2011.
5. Lyza Danger Gardner, Jason Grigsby, “Head First Mobile Web“, O’Reilly Media, 2012.
6. Benjamin LaGrone, “HTML5 and CSS3 Responsive Web Design Cookbook“, Packt 2013.
7. Craig Sharkie, Andrew Fisher, “Jump Start Responsive Web Design“, SitePoint, 2013.
8. Ben Frain, “Responsive Web Design with HTML5 and CSS3“, Packt, 2012.
9. Jennifer Niederst Robbins, “Learning Web Design“, O’Reilly Media, 2012.
10. Janine C. Warner, “Uradi sam: Web strane za neupućene“, Mikro knjiga, 2009.
11. S.Janićijević, D.Pelemiš, „“Web dizajn: HTML & JavaScript“, Prometej Novi Sad,2006.
12. Molly Holzschlag, “Skok u HTML i CSS“, Kompjuter biblioteka, 2006.
13. B. Nikolić, Programiranje Internet aplikacija, Beograd, 2008.
14. http://html.net/tutorials/html/
15. http://www.w3schools.com/
16. www.w3.org
17. http://meyerweb.com
18. http://css-tricks.com/

Indeks pojmova

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Indeks slika

Indeks tabela