# 1

# Upoznavanje s CSS-om

Veb stranica se sastoji od *markiranja* ili *označavanja* (engl. *markup*) – segmenata HTML ili XHTML koda koji opisuju značenje sadržaja na stranici – i *kaskadnih opisa stilova* (engl. *Cascading Style Sheets*, skraćeno *CSS*), koji saopštavaju čitaču veba kako će se taj sadržaj prikazivati u čitačima i drugim korisničkim agentima. CSS daje čitaču veba sve vrste informacija o izgledu – od rasporeda elemenata do boja naslova na stranici.

U ovom poglavlju, koje je organizovano drugačije od ostalih, upoznaću vas sa osnovama sintakse CSS-a i pokazati kako da je primenite na veb stranicama. Ako ste već koristili CSS, možete preskočiti ovo poglavlje i preći na konkretne primere u poglavlju 2.

Ova knjiga nije udžbenik nego zbirka zadataka s rešenjima koja će vam pomoći da izvršavate zadatke u CSS-u. Ako nemate ni osnovna znanja o HTML-u i CSS-u, preporučujem da, zajedno sa ovom knjigom, koristite i knjigu *Build Your Own Website the Right Way Using HTML & CSS* (SitePoint, treće izdanje) autora Iana Lloyda.[1] Ukoliko ste već koristili HTML i CSS, upotrebite ovo poglavlje kao podsetnik dok rešavate zadatke iz narednih poglavlja.

## Definisanje stilova pomoću CSS-a

Osnovna namena CSS-a jeste da veb dizajnerima omogući definisanje *deklaracija stilova* (engl. *style declarations*) – tj. detalja o formatiranju, kao što su fontovi, veličine elemenata i boje – a zatim da te stilove primene na izabrane delove HTML stranica pomoću *selektora* (engl. *selectors*): referenci na element ili grupu elemenata za koje je stil predviđen.

[1] http://www.sitepoint.com/books/html3/

Pokazaćemo kako se to radi na jednostavnom primeru. Razmotrimo naredni HTML dokument:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>A Simple Page</title>
</head>
<body>
  <h1>First Title</h1>
  <p>A paragraph of interesting content.</p>
  <h2>Second Title</h2>
  <p>A paragraph of interesting content.</p>
  <h2>Third title</h2>
  <p>A paragraph of interesting content.</p>
</body>
</html>
```

Ovaj dokument sadrži tri podebljana naslova definisana pomoću *oznaka* (engl. *tags*) `h1` i `h2`. Ukoliko se dokument ne formatira pomoću CSS-a, za naslove će se koristiti interni opis stila definisan u *čitaču veba* (engl. *Web browser*); naslov `h1` će se prikazati krupnim fontom, dok će se za naslov `h2` upotrebiti manji font ali ipak krupniji od fonta u tekstu pasusa. Dokument koji koristi ove podrazumevane stilove biće *čitljiv*, ali ne i atraktivan. Izgled navedenih elemenata možemo izmeniti pomoću jednostavnih kaskadnih opisa stilova:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>A Simple Page</title>
  <style>
    h1, h2 {
      font-family: "Times New Roman", Times, serif;
      color: #3366cc;
    }
  </style>
</head>
<body>
  <h1>First Title</h1>
  <p>A paragraph of interesting content.</p>
  <h2>Second Title</h2>
  <p>A paragraph of interesting content.</p>
  <h2>Third title</h2>
  <p>A paragraph of interesting content.</p>
</body>
</html>
```

Sva magija se odvija između oznaka `style` u elementu `head` dokumenta, gde možete zadati svetloplavu boju i beserifni font, i primeniti ih na sve elemente **`h1`** i **`h2`** na stranici. Ne zamarajte se sintaksom, uskoro ću je detaljno objasniti. Nema potrebe da išta dodajemo u HTML kôd – izmene u definiciji stila na vrhu stranice odraziće se na sva tri naslova, kao i na svaki naslov koji će se kasnije dodati stranici.

**HTML ili XHTML?**

U ovoj knjizi primeri će se predstavljati HTML5 dokumentima u kojima će se koristiti XML sintaksa, jer se meni tako više dopada. Međutim, svi primeri će raditi i u XHTML ili HTML4 dokumentima.

## Umetnuti stilovi

CSS stilovi se najlakše dodaju veb stranicama pomoću *umetnutih* ili *rednih stilova* (engl. *inline styles*). Umetnuti stil se primenjuje na HTML element pomoću atributa `style`, kao u sledećem odlomku koda:

```
<p style="font-family: "Times New Roman", Times, serif;
  color: #3366cc;">
  Amazingly few discotheques provide jukeboxes.
</p>
```

Upotreba umetnutih stilova ne zahteva selektore – primenjuju se deklaracije stilova na nadređeni element. U gornjem primeru to je oznaka `p`.

Umetnuti stilovi imaju ozbiljan nedostatak: ne mogu se koristiti više puta. Na primer, da bismo prethodni stil primenili na drugi element `p`, morali bismo ponovo da ga upišemo u atributu `style` tog elementa. Ukoliko bismo kasnije taj stil želeli da izmenimo, morali bismo da pronađemo i izmenimo svaku HTML oznaku u koju je taj stil kopiran. Uz to, umetnuti stilovi se definišu zajedno s kodom stranice, što otežava čitljivost i ažuriranje koda.

## Ugrađeni stilovi

Drugi način primene CSS stilova na veb stranice jeste pomoću elementa `style`, kao u prvom primeru koji smo razmotrili. Pomoću *ugrađenih stilova* (engl. *embedded styles*) možete deklarisati proizvoljan broj CSS stilova između početnih i završnih oznaka style na sledeći način:

```
<style>
  ⋮ Ovde dolaze CSS stilovi…
</style>
```

Oznake `style` se postavljaju u okviru elementa `head`. Iako je prikladna i jednostavna, oznaka `style` ima jedan veliki nedostatak: da biste na veb lokaciji koristili određeni skup stilova, morate ponoviti njihove definicije u okviru elementa `style` na vrhu svake stranice na veb lokaciji.

Mnogo je praktičnije da ove definicije postavite u običnu tekstualnu datoteku s kojom ćete kasnije povezati svoje dokumente. Ovu spoljnu datoteku nazivamo *spoljni opis stila* (engl. *external stylesheet*).

## Spoljni opisi stilova

*Spoljni opis stila* je datoteka (obično sa oznakom tipa **.css**) koja sadrži CSS stilove veb lokacije. Na taj način opisi stilova se čuvaju nezavisno od bilo koje veb stranice. S jednom **.css** datotekom može se povezati više stranica, a svaka izmena definicija stilova u toj datoteci odraziće se na sve stranice s kojima je povezana. Tako ostvarujemo prethodno pomenuti cilj – pravljenje definicija stilova za čitavu veb lokaciju.

Da biste dokument povezali sa spoljnim opisom stila (na primer, s datotekom **styles.css**), dovoljno je da postavite element `link` u zaglavlje dokumenta (element `head`):

```
<link rel="stylesheet" href="styles.css" />
```

Sećate li se našeg prvog primera, u kome su tri naslova delila isto *pravilo stila* (engl. *style rule*)? Sačuvajmo to pravilo u spoljnom opisu stila pod nazivom **styles.css**, i povežimo ga s veb stranicom na sledeći način:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>A Simple Page</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <h1>First Title</h1>
  <p>A paragraph of interesting content.</p>
  <h2>Second Title</h2>
  <p>A paragraph of interesting content.</p>
  <h2>Third title</h2>
  <p>A paragraph of interesting content.</p>
</body>
</html>
```

Vrednost atributa `rel` mora biti opis stila (`stylesheet`). Atribut `href` sadrži lokaciju i ime datoteke opisa stila (`styles.css`).

### Nije vaš tip

Često ćete videti da veza ka opisu stila izgleda kao: `<link rel="stylesheet" type="text/css" href="styles.css" />`. Ovde izostavljamo atribut `type`, jer koristimo HTML5 koji ga ne zahteva.

Povezana datoteka **styles.css** sadrži sledeću definiciju stila:

```
h1, h2 {
  font-family: "Times New Roman", Times, serif;
  color: #3366cc;
}
```

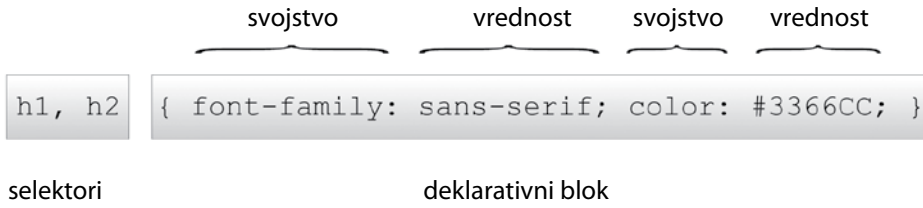
Datoteku **styles.css** možete koristiti na svakoj stranici, baš kao i datoteke sa slikama. Zahvaljujući tome ne morate ponovo upisivati definicije stilova, a naslovi će se sigurno prikazivati ujednačeno na čitavoj veb lokaciji.

## Sintaksa CSS-a

*Opis stila* je skup *definicija stilova*. Svaka CSS definicija stila, ili pravilo, ima dve glavne komponente:

- Listu od jednog ili više *selektora* razdvojenih zarezima, koja definiše element ili elemente na koje će se stil primeniti.
- *Deklarativni blok* (engl. *declaration block*), odvojen vitičastim zagradama {...}, koji određuje šta se datim stilom zapravo postiže.

Deklarativni blok sadrži jednu ili više *deklaracija stila*, koje definišu vrednosti određenog *svojstva* (engl. *property*). Deklaracije se razdvajaju tačkom i zarezom (;). Deklaracija svojstva se sastoji od imena svojstva i njegove vrednosti razdvojenih dvotačkom (:). Ove elemente možete videti na slici 1.1.




*Slika 1.1. Komponente CSS pravila: lista selektora i deklarativni blok.*

Rešenja u knjizi pretežno će se oslanjati na različita svojstva i vrednosti koje ta svojstva mogu imati. Na slici 1.1 se vidi da se pravilo stila može napisati u jednom redu. Neki autori uvlače svako pravilo stila da bi kôd bio pregledniji, a drugi ih pišu u jednom redu da bi uštedeli prostor. U nastavku će biti prikazana oba načina pisanja pravila stilova:

```
h1, h2 {
  font-family: "Times New Roman", Times, serif;
  color: #3366cc;
}

h1, h2 {
  font-family: "Times New Roman", Times, serif; color: #3366cc;
}
```

Način pisanja nije bitan i zavisi isključivo od vas.

## Šta su CSS selektori i kako se koriste

Selektore koristimo da bismo na stranici izabrali deo koda koji formatiramo. Ovo može biti jednostavno (biranje određenog HTML elementa po imenu) ili složeno (biranje elementa koji se nalazi na određenoj poziciji ili u određenom stanju). U narednom primeru, `h1` i `h2` su selektori, što znači da pravilo treba primeniti na sve elemente `h1 i h2`:

```
h1, h2 {
  font-family: Times, "Times New Roman", serif;
  color: #3366CC;
}
```

Primere CSS selektora viđaćemo i dalje u ovoj knjizi, i zato treba što pre da se upoznate s različitim tipovima selektora i načinima njihovog rada. U tome će vam pomoći primeri za osnovne tipove selektora, dati u nastavku.

## Selektori tipa

Osnovni oblik selektora je *selektor tipa* (engl. *type selector*) s kojim smo se već sreli. Prilikom imenovanja određenog HTML elementa, možete primeniti pravilo stila na taj element kad god se pojavi u dokumentu. Selektori tipa često se koriste da bi se definisali osnovni stilovi koji će se pojavljivati na veb lokaciji. Na primer, pomoću narednog pravila stila možemo zadati podrazumevani font `h1` za veb lokaciju:

```
h1 {
  font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif;
  font-size: 1.2em;
  color: #000000;
}
```

Definisali smo font, veličinu i boju za sve elemente `h1` u dokumentu.

## Selektori klase

Dodela stilova elementima je dobra praksa, ali šta ako želite da dodelite različite stilove identičnim elementima koji se pojavljuju na različitim mestima u dokumentu? Ovde na scenu stupaju *CSS klase*.

Razmotrimo naredni still, koji sve naslove `h2` u dokumentu boji u plavo:

```
h2 {
  color: #0000ff;
}
```

Divota! Ali šta će se desiti ako na stranici postoji i *izdvojen odeljak teksta*, tj. *bočna traka* (engl. *sidebar*) na plavoj pozadini? Ako bi tekst ovog odeljka takođe bio plav, ne bi se video. Potrebno je da definišete klasu za izdvojeni odeljak i da potom toj klasi dodelite CSS stil.

Prvo izmenite HTML kôd kako biste dodali klasu naslovu:

```
<h2 class="sidebar">This text will be white, as specified by the
  CSS style definition below.</h2>
```

Sada napišite stil za ovu klasu:

```
h2 {
  color: #0000ff;
}

.sidebar {
  color: #ffffff;
}
```

U drugom pravilu se pomoću selektora klase zadaje da se stil primenjuje na sve elemente čija je vrednost klase `.sidebar`. Tačka (`.`) pokazuje da imenujemo klasu, a ne HTML element.

Klasu možete dodati proizvoljnom broju elemenata u dokumentu.

## ID selektori

Suprotno selektorima klase, *ID selektori* služe za biranje određenog elementa umesto grupe elemenata. Da biste upotrebili ID selektor, prvo pridružite atribut `id` elementu koji stilizujete. Važno je da ID bude jedinstven u čitavom HTML dokumentu:

```
<p id="tagline">This paragraph is uniquely identified by the ID
  "tagline".</p>
```

Ovaj element referenciramo pomoću njegovog ID selektora tako što ispred atributa ID navodimo znak tarabe (`#`). Na primer, naredno pravilo nalaže da se prethodni pasus prikaže belom bojom:

```
#tagline {
  color: #ffffff;
}
```

## Kombinatori

Narednu grupu selektora koju ćemo obraditi čine *kombinatori* (engl. *combinators*). Kombinatori upućuju na znak koji se postavlja između dva jednostavna selektora. Tako se dobija moćniji selektor kojim se još preciznije može izdvojiti odgovarajući deo dokumenta.

### Selektori potomaka

*Selektor potomaka* (engl. *descendant selector*) odgovara elementu koji je potomak zadatog elementa. U ovom selektoru se kao kombinator koristi razmak.

Pretpostavimo da na veb lokaciji imate element `h2` koji treba da se prikaže u plavoj boji; međutim, unutar izdvojenog odeljka teksta na veb lokaciji postoje elementi `h2` koje hoćete da prikažete u beloj boji kako bi se videli na tamnoj pozadini. Kao što smo ranije pomenuli, mogli biste dodati klasu svim ovim naslovima, ali je mnogo jednostavnije da ih definišete pomoću CSS-a. U ovakvim slučajevima koriste se selektori potomaka.

Evo novog selektora:

```
.sidebar h2 {
  color: #ffffff;
}
```

A evo i ažuriranog HTML koda:

```
<div class="sidebar">
  <h2>A heading in white</h2>
  <h2>Another heading in white</h2>
</div>
```

Kao što vidite, selektor potomaka se sastoji od liste selektora (razdvojenih razmacima) koja odgovara elementu stranice (ili grupi elemenata) „od spolja ka unutra“. U ovom slučaju, pošto stranica sadrži element `div` klase `sidebar`, selektor potomaka `.sidebar h2` odnosi se na sve elemente `h2` unutar tog elementa `div`.

Ako koristite selektore potomaka, ne morate pristupati svom HTML kodu kako biste direktno pridruživali klase svim elementima. Možete koristiti glavne strukturne oblasti na stranici – koje se identifikuju pomoću klasa ili identifikatora tamo gde se to traži – i elemente stilova unutar njih.

## Selektori dece

Za razliku od selektora potomaka – koji odgovaraju svim elementima potomcima roditeljskog elementa, uključujući elemente koji *nisu direktni* potomci – *selektor dece* (engl. *child selector*) odgovara svim elementima koji su direktni potomci zadatog elementa. U ovom selektoru se kao kombinator koristi znak `>`.

Razmotrimo naredni segment koda:

```
<div class="sidebar">
  <p>This paragraph will be displayed in white.</p>
  <p>So will this one.</p>
  <div class="tagline">
    <p>If we use a descendant selector, this will be white too.
      But if we use a child selector, it will be blue.</p>
  </div>
</div>
```

U ovom primeru, selektor potomaka koji smo videli u prethodnom odeljku, `.sidebar p`, odgovarao bi svim pasusima ugnježđenim u element `div` sa klasom `sidebar`, kao i u one unutar elementa `div` sa klasom `tagline`. Ako želite samo da stilizujete one pasuse koji su direktni potomci elementa `sidebar div`, upotrebićete selektor dece. Selektor dece koristi znak `>` da bi zadao direktnog potomka.

Evo novog selektora, kojim se u belo boji tekst onih pasusa koji su direktno u elementu `sidebar div` (ali ne i u elementu `tagline div`):

```
p {
  color: #0000FF;
}
```

```
.sidebar>p {
  color: #ffffff;
}
```

### Selektori susednih elemenata

*Selektor susednog elementa* (engl. *adjacent selector*) odgovara elementu samo ako je on susedan drugom zadatom elementu. Kombinator ovog selektora je znak plus (+).

Pretpostavimo da imamo HTML kôd:

```
<h2>This is a title</h2>

<p>This paragraph will be displayed in white.</p>

<p>This paragraph will be displayed in black.</p>
```

Upotrebimo naredni selektor:

```
p {
  color: #000000;
}

h2+p {
  color: #FFFFFF;
}
```

Samo će prvi pasus biti obojen u belo. Drugi element `p` nije susedan elementu `h2`, pa će njegov tekst biti prikazan crnom bojom, što smo zadali u prvom pravilu za elemente `p`.

## Selektori pseudoklasa

*Selektor pseudoklasa* (engl. *pseudo-class selector*) deluje kao da element ima klasu koja se primenjuje u skladu sa stanjem tog elementa. Selektori pseudoklasa na početku imaju dvotačku i obično se navode iza selektora tipa elementa, bez dodatnih razmaka.

Moj cilj je da vas upoznam sa sintaksom i terminologijom vezanom za ove selektore, kako biste razumeli njihovo delovanje kada ih budemo sretali u nastavku ove knjige. Zato u ovom poglavlju neću detaljno obrađivati sve selektore, a kompletan spisak sa objašnjenjima možete pronaći na mreži (SitePoint CSS Reference).[2]

### Selektori veza

Većina nas se prvi put sretne sa pseudoklasama tokom njihove primene na veze. *Veza* (engl. *link*) ima različita stanja. Ona može biti *neposećena* (engl. *unvisited*) ili *posećena* (engl. *visited*), *pokazana mišem* (engl. *hovered over*) ili *pritisnuta* (engl. *clicked*). Svako stanje veze možemo definisati pomoću CSS-a:

```
a:link {
  color: #0000ff;
}
```

[2] http://reference.sitepoint.com/css/selectorref

```
a:visited {
  color: #ff00ff;
}

a:hover {
  color: #00ccff;
}

a:active {
  color: #ff0000;
}
```

Prva definicija zadaje boju neposećene veze. Ako je veza posećena, primenjuje se drugo pravilo. Ukoliko je na vezu postavljen pokazivač miša, koristi se definicija `:hover`, a ako je pritisnuta ili aktivirana na drugi način, definicija `:active`. Selektori pseudoklasa `:hover` i `:active` nazivaju se i *dinamičke pseudoklase* (engl. *dynamic pseudo-classes*), jer deluju samo kad korisnik radi sa elementom. Da bi se one aktivirale, nešto se mora prethodno desiti.

Redosled definicija u dokumentu je važan. Definicija `a:active` mora se navesti poslednja kako bi redefinisala prethodne definicije. O tome ćemo detaljnije govoriti u nastavku poglavlja kada budemo obrađivali kaskadu.

## Selektori first-child

Selektor pseudoklase `first-child` odnosi se na element koji predstavlja prvo dete roditeljskog elementa. I ovaj selektor ćete lakše savladati pomoću primera.

Dokument sadrži skup pasusa. Oni se nalaze unutar elementa `div` sa klasom `article`. Pomoću CSS-a i selektora potomaka možemo definisati da font u svim ovim pasusima bude krupniji i podebljan:

```
.article p {
  font-size: 1.5em;
  font-weight: bold;
}
```

Da biste prvi pasus prikazali krupnijim i podebljanim fontom – kao u uvodnim pasusima članaka – upotrebite `first-child`:

```
.article p:first-child {
  font-size: 1.5em;
  font-weight: bold;
}
```

Čitač veba će primeniti CSS samo ako pasus predstavlja prvi element `p` unutar elementa sa klasom `article`. Zato je selektor pseudoklase prvog deteta koristan kada hoćete da ulepšate tekst (na primer, da prvi pasus nekog teksta ili prva pojava naslova budu malo drugačiji od ostatka).

### Selektori last-child

Kao što smo pomoću selektora `first-child` adresirali prvu instancu nekog elementa u kontejneru, pomoću selektora `last-child` odredićemo poslednju instancu elementa. Sledi CSS koji postavlja ivicu ispod svake stavke u listi:

```
.navigation li {
  border-bottom: 1px solid #999999;
}
```

Da se ivica ne bi prikazala ispod poslednje stavke, upotrebite naredni CSS kôd:

```
.navigation li {
  border-bottom: 1px solid #999999;
}

.navigation li:last-child {
  border-bottom: none;
}
```

### Selektori nth-child

Selektor pseudoklase `nth-child` bira više elemenata, zavisno od njihove pozicije u stablu dokumenta. Sledi jednostavan primer u kome ćemo segmentirati redove kako bi tabela postala preglednija.

Naredna CSS deklaracija dodeljuje boju pozadine samo ćeliji u neparnom redu tabele:

```
tr:nth-child(odd) td {
  background-color: #f0e9c5;
}
```

Osim ključnih reči za *parne* (engl. *even*) i *neparne* (engl. *odd*) brojeve, možete upotrebiti i izraz s množiocem:

```
tr:nth-child(2n+1) td {
  background-color: #f0e9c5;
}
```

U nastavku knjige detaljnije ćemo govoriti o selektoru `nth-child` i objasniti kako se množioci mogu koristiti za biranje odgovarajućih delova tabele s podacima.

### Selektori only-child

Selektor pseudoklase `only-child` bira element samo ako je on jedino dete roditeljskog elementa. Na primer, pretpostavimo da u HTML kodu postoje dve liste koje stoje jedna iza druge, i da prva sadrži dve stavke a druga jednu:

```
<ul>
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ul>

<ul>
  <li>A single item list – not really a list at all!</li>
</ul>
```

Naredna CSS deklaracija referisaće samo stavku druge liste jer je element `li` jedino dete roditeljskog elementa `ul`:

```
li:only-child {
  list-style-type: none;
}
```

## Selektori pseudoelemenata

*Pseudoelementi* (engl. *pseudo-elements*) deluju kao da ste umetnuli nov segment HTML koda na stranicu a zatim ga stilizovali. Pseudoelementi se u CSS3 specifikaciji označavaju duplom dvotačkom, na primer, `p::first-letter`.

Kada je reč o pseudoelementima koji su postojali u CSS2 (na primer, `::first-letter`, `::first-line`, `::before` i `::after`), od proizvođača čitača veba se traži da održavaju podršku za sintaksu s jednom dvotačkom koju su ovi selektori ranije koristili. Ako koristite prethodne selektore, u vreme pisanja ove knjige jedna dvotačka trebalo bi da je dobila bolju podršku čitača veba pa savetujem da je primenjujete. Izuzetak je `::selection`, koja je dodata u CSS3 specifikaciji.

### Selektori first-letter

Selektor pseudoelementa `first-letter` deluje kao da ste prvo slovo sadržaja unutar roditeljskog elementa postavili između oznaka `span` a zatim ga stilizovali. Na primer, ako smo u kodu koristili oznaku `span`, on može izgledati ovako:

```
<div class="wrapper">
<p><span class="firstletter">T</span>his is some text within a div
  with a class of wrapper.</p>
</div>
```

U CSS kodu će stajati:

```
.wrapper .firstletter {
  font-size: 200%;
  font-weight: bold;
}
```

Možemo i ukloniti `span` iz koda i na isti način izabrati prvo slovo pomoću selektora pseudoelementa prvog slova:

```
.wrapper:first-letter {
  font-size: 200%;
  font-weight: bold;
}
```

### Selektori first-line

Isto kao što selektor prvog slova bira prvo slovo unutar kontejnera, selektor `first-line` bira prvi red:

```
.wrapper:first-line {
  font-size: 200%;
  font-weight: bold;
}
```

Selektor `first-line` ima mnogo više mogućnosti od postavljanja prvog reda teksta između oznaka `span` i stilizovanja. Kada sadržaj stavljate u oznake `span`, ne možete znati da li će se dužina prvog reda promeniti (na primer, zbog korisnički podešene veličine teksta ili izmena koje će u tekstu praviti sistem za održavanje sadržaja). Selektor pseudoklasa `first-line` uvek će formatirati prvi red teksta onako kako se prikazuje u čitaču veba.

### Pseudoelement before

Pseudoelement `before` se koristi zajedno sa svojstvom `content` da bi se definisalo mesto prikazivanja generisanog sadržaja. *Generisani sadržaj* (engl. *generated content*) jeste sadržaj koji se u dokumentu vizuelizuje iz CSS-a. Ovo može biti korisno iz više razloga, i o tome ćemo govoriti u nastavku knjige. Za sada ćemo prikazati HTML kôd za jednostavan primer:

```
<div class="article">
  <p>Hello World!</p>
</div>
```

A CSS kôd je:

```
.article:before {
  content: "Start here";
}
```

Ovo će u čitaču veba ispisati „Start here“ unutar elementa `div` koji se otvara, odnosno, pre prvog elementa `p`.

### Pseudoelement after

Pseudoelement `after` radi na isti način kao i `before`, ali on vizuelizuje sadržaj na kraju roditeljskog elementa, tj. pre elementa `div` koji se zatvara u našem prethodnom primeru HTML koda:

```
.article:after {
  content: "End here";
}
```

Ako se upotrebi isti kôd kao i u prethodnom primeru za pseudoelement `before`, prethodni CSS će kao rezultat dati tekst „End here“ pre elementa `div` koji se zatvara, i nakon zatvaranja oznake `p`.

## Selektori atributa

*Selektori atributa* (engl. *attribute selectors*) omogućavaju da izaberete element na osnovu atributa. Primer atributa HTML elementa naći ćemo ako pogledamo element `a` koji pravi vezu. Atributi naredne veze su `href i title`:

```
<a href="http://google.com" title="Visit Google">Google</a>
```

Pomoću selektora atributa možemo proveriti vrednost atributa i prikazati CSS koji se zasniva na njoj. Jednostavan primer za to je polje `form input`; ono ima atribut `type` koji objašnjava o kojoj vrsti polja je reč. Valjane vrednosti za atribut `type` jesu `text`, `radio`

i `checkbox`. Ako pokušamo da stilizujemo `checkbox` na isti način kao polje `text input`, dobićemo vrlo čudan rezultat pa možemo primeniti selektor atributa da bismo napravili definiciju samo za polja `input` s tipom `text`. Na primer, uzmimo polje `form`:

```
<input type="text" name="name" id="fName" />
```

CSS za definisanje ovog polja glasi:

```
form input[type="text"] {
  background-color: #ffffff;
  color: #333333;
}
```

U poglavlju 6 naći ćete više primera za korišćenje selektora atributa.

## Šta se radi sa starijim čitačima veba?

Pretpostavljate da svi čitači veba ne podržavaju CSS na isti način. Neki korisnici imaju stare verzije čitača. Primeri u ovoj knjizi trebalo bi da rade na opisani način u aktuelnim verzijama glavnih čitača veba, ali većina će raditi i u prethodnim verzijama. Ako je neka funkcija nedostupna starijim verzijama, čitač veba (na primer, Internet Explorer) skrenuće vam na to pažnju.

U poglavlju 7 izložiću više načina za usklađivanje starijih čitača veba s poslednjom verzijom CSS-a. Jedan način je primena JavaScripta za uvođenje podrške za CSS3 selektore u starije verzije Internet Explorera. Recimo, ako znate da će projekat na kome radite imati mnogo korisnika sa starijim verzijama Internet Explorera, obratite pažnju na ovo poglavlje kako biste pažljivo isplanirali svoju strategiju.

### Proširenja drugih proizvođača

U nastavku knjige pronaći ćete primere za to kako se čitači veba snalaze s verzijom CSS3. Specifikacija CSS3 se razlikuje od ranijih po tome što je modularna. Ona se rastavlja na module koji se mogu dovršavati u različito vreme – u W3C terminologiji ovo se naziva *W3C preporuka* (engl. *W3C Recommendation*). Modul se može nalaziti u sledećim stanjima:

1. Working Draft (Radni nacrt): zajednica objavljuje modul radi pregleda
2. Candidate Recommendation (Preporuka za kandidaturu): prikupljaju se informacije o iskustvima vezanim za primenu modula
3. Proposed Recommendation (Predložena preporuka): modul se šalje Savetodavnom komitetu W3C (engl. W3C Advisory Committee) na konačno odobrenje
4. W3C Recommendation (W3C preporuka): W3C odobrava model i on može da se koristi

Dok modul prelazi iz jednog stanja u drugo, proizvođači čitača veba često počinju da primenjuju modul u prvom stanju (Working Draft). To je korisno jer pomaže da se prikupi što više informacija o ponašanju specifikacije tokom rada; međutim, detalji primene se vremenom mogu promeniti u odnosu na početne uslove.

Na primer, ako ste koristili svojstvo verzije CSS3 koje se naknadno promenilo, veb lokacija napravljena pre godinu dana mogla bi odjednom poprimiti čudan izgled u novom i unapređenom čitaču veba.

Da bi se izbegao ovaj problem, proizvođači čitača u ranim fazama implementacije često koriste prefiks prozvođača kako bi napravili specifičnu implementaciju. Na primer, da bismo napravili zaobljene uglove, koristimo svojstvo `border-radius` na sledeći način:

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

Međutim, da bi zaobljeni uglovi mogli da se primene u starijim verzijama čitača Firefox i Safari, treba da umetnete prefikse proizvođača:

```
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
```

Kada modul dospe u fazu u kojoj se verovatno više neće menjati, čitači veba počinju da podržavaju stvarno svojstvo uporedo sa sopstvenim. Neki čitači nemaju posebne verzije proizvođača i primenjuju samo verziju iz specifikacije.

U ovoj knjizi ćete naći brojne primere ovakvih svojstava s prefiksom i naučićete da ih koristite.

## Kako čitač veba bira stilove koje će primeniti?

Kako čitač veba zna šta hoćemo? Kada se na jedan element može primeniti više pravila, čitač koristi *kaskadu* (engl. *cascade*) kako bi se opredelio za odgovarajuće svojstvo stila.

Razumevanje kaskada je važno za rad sa CSS-om. Tokom razvoja CSS-a, probleme često izaziva to što se neki stil nehotično primeni na određeni element. U ovom poglavlju već smo prikazali primere u kojima smo pisali opšte pravilo stila za elemente pasusa, a zatim određenije pravilo za jedan ili više pasusa. Oba pravila stila odnosila su se na pasuse, *ali pri radu s pasusima specifično pravilo nadjačava opšte pravilo*.

Postoje četiri faktora koje čitači veba koriste za donošenje odluke: težina, poreklo, specifičnost i redosled primene.

*Težinu* (engl. *weight*) neke deklaracije stila određuje korišćenje ključne reči `!important`. Kada se ova reč prikaže nakon vrednosti svojstva, ta vrednost ne može biti redefinisana istim svojstvom u drugom pravilu stila, osim u posebnim okolnostima. Korišćenje ključne reči `!important` u opisima stilova negativno utiče na njihovo održavanje, a ionako se retko poziva. Iz ovih razloga treba je izbegavati, što ćemo i mi činiti u ovoj knjizi. Ako želite da saznate više o ovoj temi, potražite je na veb lokaciji SitePoint CSS Reference.[3]

Postoje tri moguća *porekla* (engl. *origins*) opisa stilova: čitač veba, autor i korisnik. U ovoj knjizi ćemo se usredsrediti na takozvane *autorske opise stilova* (engl. *author stylesheets*), odnosno, opise stilova koje pravi dizajner veb stranice – a to ste vi! Pomenuli smo interni opis stila čitača veba, koji definiše podrazumevane stilove za sve elemente, ali stilovi u

[3] http://reference.sitepoint.com/css/importantdeclarations

autorskim opisima stilova uvek će redefinisati stilove u podrazumevanim opisima stilova čitača. Jedino moguće drugo poreklo za opise stilova jesu korisnički opisi stilova – namenski stilovi koje su pisali korisnici čitača veba – a čak i njih redefinišu autorski opisi stilova (osim u posebnim slučajevima). Ako želite da saznate više o ovoj temi, posetite poseban odeljak na lokaciji SitePoint CSS Reference.

Dva dela kaskade koja će najviše uticati na vaš svakodnevni rad sa CSS-om jesu specifičnost i redosled primene. Pravilo *specifičnosti* (engl. *specificity*) omogućava da pravilo stila sa selektorom najvišeg nivoa specifičnosti redefiniše ostale selektore s nižim nivoima specifičnosti.

Da bismo ovaj problem ilustrovali primerom, razmotrite jednostavan odlomak HTML koda:

```
<div id="content">
  <p class="message">
    This is an important message.
  </p>
</div>
```

Razmotrite sada naredna pravila stila koja će se primeniti na ovaj HTML:

```
p { color: #000000; }
.message { color: #CCCCCC; }
p.message { color: #0000FF; }
#content p.message { color: #FF0000; }
```

Sva četiri selektora odgovaraju elementu pasusa u primeru HTML-a i zadaju boju teksta. Koja će se boja primeniti na pasus? Ako je vaš odgovor `#FF0000` ili crvena, u pravu ste. Tip selektora `p` (svaki element `p`) ima najniži nivo specifičnosti, a sledi `.message` (svaki element s klasom `message`). Selektor `p.message` (svaki element `p` s klasom `message`) nakon toga ima viši nivo specifičnosti. Najviši je selektor `#content p.message` (svaki element `p` sa klasom `message` koji predstavlja dete elementa sa sadržajem `id`).

Duži selektori nemaju nužno viši nivo specifičnosti. Selektor ID će, na primer, uvek imati višu specifičnost od elemente `type` ili klase `selector`. Što su selektori složeniji, stvar postaje zapetljanija ali se nadam da će primeri u ovoj knjizi biti dovoljno jednostavni za razumevanje. Ako hoćete da znate tačnu formulu za merenje specifičnosti, posetite veb lokaciju SitePoint CSS Reference, koja sadrži odgovore na sva vaša pitanja.[4]

Ako se dva ili više stilova i dalje mogu primeniti na neki element, upotrebiće se pravilo *redosleda primene* (engl. *source order*). Primenjuje se poslednje pravilo koje se deklariše. Ovo važi i u slučajevima kada deklarišete više pravila stilova jednim selektorom (na primer, `.message` u vašem opisu stilova). Ovo će biti druga pojava pravila koje će se primeniti na element. Kao što ćemo videti u narednim poglavljima, ova osobina je vrlo korisna.

[4] http://reference.sitepoint.com/css/specificity

## Da li će korišćenje CSS-ovih radnih okvira olakšati učenje CSS-a?

Otkako sam napisala prethodno izdanje ove knjige, dizajneri pri radu sve više koriste CSS-ove radne okvire (engl. *CSS frameworks*).

Lično smatram da su radni okviri korisni ali ne mogu zameniti učenje CSS-a. Kada savladate CSS i počnete da ga primenjujete u projektima, pojaviće se konkretni zahtevi i problemi koje ćete moći da rešite primenom dostupnih alatki i radnih okvira. Ako vam oni pomognu da uspešno savladate probleme – odlično! Nema ničeg lošeg u tome da u svoj rad ugradite i rad drugih ljudi. Međutim, ukoliko ne poznajete dobro CSS, korišćenje radnih okvira će vas samo još više zbunjivati – sve će postati složenije i teže ćete savladati osnovne pojmove CSS-a.

## Pravi izbor

Ovo poglavlje je trebalo da vas uvede u svet CSS-a i njegovu primenu na osnovnom nivou. Pozabavili smo se čak i konceptom kaskade, koji ponekad može delovati zbunjujuće. Ako nemate iskustvo u radu sa CSS-om ali razumete pojmove o kojima se govorilo u ovom poglavlju, verovatno ćete bez većih problema koristiti primere iz ove knjige.

Primeri u prvim poglavljima su jednostavniji od onih u kasnijim poglavljima. Zavisno od nivoa znanja CSS-a, rešavajte ih onim redom koji vam odgovara. Nadam se da ćete uživati u radu sa CSS-om.