CSS osnove

U ranijim lekcijama smo se bavili isključivo HTML jezikom i njegovim svojstvima. Videli smo kako se unosi sadržaj na stranice, a sada ćemo se upoznati sa CSS-om, njegovim osnovnim pravilima i stilizacijom elemenata.

Šta je CSS

CSS (*Cascading Style Sheets*) je jezik koji se koristi za opis prezentacione semantike dokumenta pisanog u opisnom jeziku (markup language). Jednostavnije rečeno, opisuje, tj. uređuje izgled i formatiranja bilo kog elementa na stranici. On nam omogućuje da, kreirajući **CSS opise, stilizujemo elemente HTML** i drugih dokumenata.

CSS je nastao dosta kasnije nego što je to slučaj sa HTML-om. U početku, HTML stranice nisu bile vizuelno stilizovane. Potom smo koristili HTML tagove za stilizaciju, ali je sve to bilo previše komplikovano i nepraktično. Zato je osmišljen CSS jezik koji ostavlja HTML strukturi da prikazuje sadržaj, a CSS taj sadržaj stilizuje. U početku, mnogi autori su kombinovali HTML i CSS stilizaciju, ali danas, kada je CSS potpuno sazreo, možemo **u potpunosti poštovati pravilo razdvojenosti sadržaja (HTML) i stilizacije (CSS).**

CSS nam omogućuje da kreiramo opise pomoću kojih ćemo stilizovati određene elemente. Na primer, CSS-om možemo definisati da pozadina strane bude plava i da sav tekst na njoj bude u fontu Arial i to u crvenoj boji. Takođe, možemo postaviti da svi paragrafi imaju određene margine i tako dalje.

U CSS-u ne pišemo tagove niti postavljamo bilo kakav HTML sadržaj. On služi isključivo za uređivanje i stilizaciju sadržaja u HTML i drugim dokumentima.

CSS je odvojen od HTML jezika i njegovih tagova. Ukoliko imamo kreiranu HTML stranicu, nije potrebno da prepravljamo taj kôd kako bi uneli CSS kôd. Njega unosimo ili u posebnom fajlu (sa ekstenzijom *.css*) ili u head delu HTML dokumenta. Kasnije ćemo se vratiti detaljno na to kako i gde se unosi CSS.

Kako funkcioniše CSS

Osnova razumevanja funkcionisanja CSS jezika je da zamislimo nevidljiv okvir (box) oko svakog elementa na strani. CSS-om kreiramo opise pomoću kojih stilizujemo taj okvir, ali i same elemente unutar njega.

CSS sintaksa se sastoji iz **CSS opisa (CSS Rule).** Sami CSS opisi su obavezno sastavljeni iz dva dela. Razlikujemo **selektor (selector) i deklaraciju (declaration)**.

Selektor ukazuje na koji element (ili više njih, odvajamo iz zarezom) se odnosi taj CSS opis.

Deklaracija postavlja stilizaciju za element na koji se odnosi CSS opis.

Kada kreiramo CSS opis, postavljamo selektor, zatim unosimo razmak i postavljamo vitičaste zagrade u kojima pišemo deklaracije. Kao na sledećem primeru:

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

Primer kôda iznad predstavlja jedan jednostavan CSS opis.

Deklaracije se takođe sastoje iz dva dela, a to su **svojstvo (property)** i **vrednost (value).** Pišemo svojstvo, postavljamo dve tačke (:) i zatim vrednost. U istom opisu možemo pisati više deklaracija, samo ih odvajamo znakom tačka-zarez (;). Pogledajmo primer:

Ukoliko želimo da postavimo da paragrafi naše stranice budu u fontu Tahoma i da imaju veličinu 12px, pisaćemo:

```
p {
      font-family:Tahoma;
      font-size:12px;
}
```

U ovom primeru postavili smo selektor **p** koji označava da se taj opis odnosi na sve paragrafe u dokumentu (na <p> tagove u HTML-u). Postavili smo vitičaste zagrade i u njima pisali dve deklaracije koje se odnose na izbor fonta (font-family) i izbor veličine teksta (font-size). Primećujete da između svojstva i vrednosti u deklaraciji stoji uvek znak **:** (bez razmaka), a da posle deklaracije uvek stoji znak **;** (koji označava kraj vrednosti)**.**

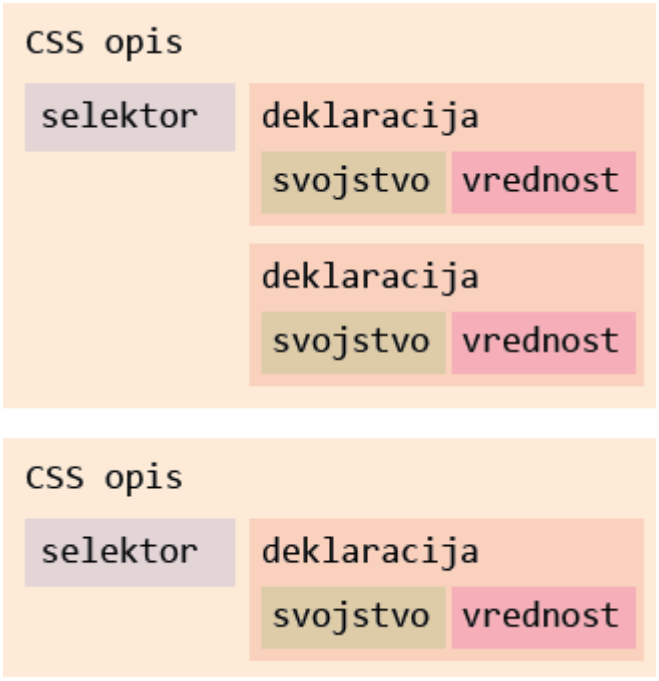
Važno je pomenuti da novi redovi i razmaci u CSS jeziku ne igraju preterano značajnu ulogu, ali treba poštovati neku strukturu radi preglednosti kôda. Uobičajeno je da se pišu selektor i početna zagrada u jednom redu. Zatim, na sledećim redovima sve deklaracije (po jedna po redu) i na kraju zatvaramo zagradu u novom redu, kao na primeru iznad.

CSS selektori razlikuju velika i mala slova tako da moramo voditi računa o tome.

Neke vrednosti mogu imati više vrednosti (odvajamo zarezom), ili mogu imati više grupa vrednosti (odvajamo razmakom) ili mogu sadržati string (tekst) i tada pišemo vrednost pod znacima navoda. Može doći i do kombinacije tih načina pisanja. Pogledajmo primere za ove slučajeve:

```
p {
      font-family:”Times New Roman”, Arial;
      margin:10px 0 0 0;
}
```

Pre nego što krenemo dalje, dobro zapamtite ove termine koje smo pominjali do sada u ovoj lekciji jer ćemo se uvek vraćati na njih. Dakle, da ponovimo: **CSS jezik se sastoji iz opisa. Svaki opis je sastavljen iz selektora i deklaracija. Deklaracija se sastoji iz svojstva i vrednosti.**



CSS opisi

Način pisanja CSS opisa

Pomenuli smo da je CSS odvojen od HTML koda, ali to znači da ga postavljamo... Gde? Postoje tri varijante, od kojih se dve aktivno koriste:

Eksterni CSS

Kao što mu i samo ime kaže, eksterni CSS se nalazi kompletno van HTML dokumenta u posebnom fajlu.

Najprostija varijanta je kada imamo jedan HTML fajl i pored njega jedan CSS fajl. Dovoljno je da u head delu HTML fajla unesemo određeni tag, koji će povezati eksterni CSS fajl. Sa druge strane, u samom CSS fajlu pišemo samo opise. Ne sme se pojaviti nikakav deo HTML jezika u CSS fajlovima. CSS fajlovi, za razliku od HTML-a, sastoje se samo od CSS opisa.

Sam tag za povezivanje CSS opisa u head delu strane je **<link>.** Poput nekih drugih tagova (kao što je meta, npr.), nema sadržaj i samozatvarajući je, a sve vrednosti nosi u svojim atributima nad tagom. Ti atributi u slučaju <link> taga su:

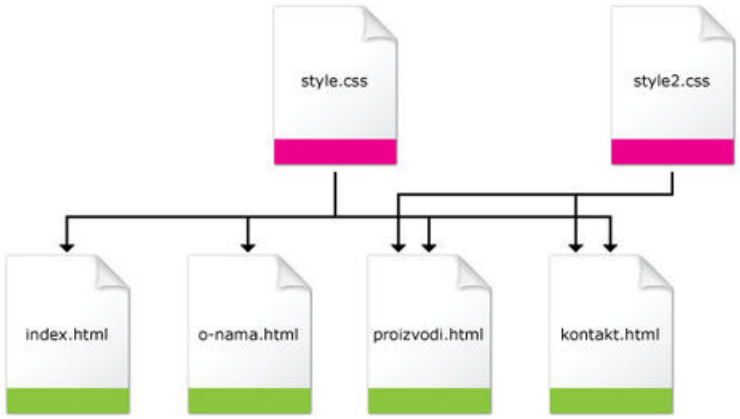
- **href** – definiše putanju do eksternog CSS fajla. Može biti relativna ili apsolutna. Preporuka je postavljati relativne putanje i voditi računa da svi fajlovi sajta uvek budu zajedno i u istim odnosima. Ovaj isti atribut smo susretali kod img taga, u sličnoj funkciji (putanja do slike).
- **type** – definiše tip dokumenta ka kome se linkuje pošto se tag <link> može koristiti i za neke druge svrhe, mada se u poslednje vreme koristi samo za CSS. Postavljamo: *"text/css"*

- **rel** – definiše odnos HTML fajla i linkovanog, eksternog fajla. Postavljamo: *„stylesheet"*

Evo primera za eksterno povezani CSS fajl, tačnije, početak HTML strane može izgledati ovako:

```
<!DOCTYPE html>
<html>
<head>
<title>Eksterni CSS</title>
<link href="css/stilizacija.css" type="text/css" rel="stylesheet" />
</head>
<body>
...
```

Podsećamo, ovaj tag se **uvek** postavlja u okviru **head dela** HTML fajla i praktično je uvek isti, samo se razlikuje putanja do fajla.



HTML stranice i eksterni CSS fajlovi

U jednom HTML dokumentu može postojati više povezanih CSS fajlova[1]. U tom slučaju postavićemo po jedan <link> tag za svaki eksterni CSS fajl. Podsećam, jedan CSS fajl može skladištiti neograničen broj opisa, ali većina autora razdvaja grupe CSS opisa u različitim fajlovima radi lakše organizacije.

Sa druge strane, jedan CSS fajl može biti pozvan, učitan od strane više različitih HTML fajlova. Tako možemo stilizovati iste elemente na različitim stranama istim fajlom.

[1] U starijim verzijama IE može se učitati do 20 različitih CSS fajlova u jednom dokumentu. U novijim verzijama nema ograničenja.

Interni CSS

Nasuprot eksternom, interni CSS se nalazi direktno u HTML dokumentu, tačnije u head delu. Umesto da postavljamo <link> tag kao malopre, možemo postaviti <style> tag i u okviru njega direktno upisati CSS opise. Pogledajmo primer:

```
<!DOCTYPE html>
<html>
<head>
<title>Interni CSS</title>
<style type="text/css">
      p {
            font-family:Tahoma;
            font-size:12px;
      }
</style>
</head>
<body>
...
```

Na ovaj način praktično smo *ugradili* ceo CSS fajl u strukturu HTML stranice. Van <style> taga nalazi se HTML strana i važe njena pravila. Sa druge strane, unutar njega se nalaze CSS opisi i važe pravila CSS jezika.

Interni CSS je ugrađen u HTML stranicu

Eksterni vs interni CSS

Pitanje je da li koristiti eksterni ili interni CSS. Skoro uvek je praktičnije koristiti eksterni CSS. Evo samo nekih od razloga:

- Možemo koristiti iste CSS opise za više različitih stranica, umesto da ih bespotrebno ponavljamo.
- Pošto nema ponavljanja kôda, stranice su „lakše" i brže se učitavaju (CSS se učitava samo na prvoj).
- Razdvajamo sadržaj od stilizacije.
- Možemo menjati CSS opise na jednom mestu, što će odmah uticati na sve povezane stranice.
- Lakše je snalaženje u CSS kôdu koji je odvojen od HTML-a.

Inline CSS

Pomenuli smo da postoje tri varijante pisanja CSS kôda. Pored eksternog i internog CSS-a, postoji i **inline** način pisanja. Inline se razlikuje po tome što je selektor izostavljen i sam CSS postavlja se direktno u HTML tagu, poput atributa.

Ovaj način pisanja je dozvoljen **samo u xHTML transitional i HTML4** dokumentima i predstavlja zaostatak iz nekih ranijih vremena.

Kod inline CSS-a postavljamo (na HTML tagu koji želimo da stilizujemo) atribut *style* u kome pod znacima navoda pišemo svojstva i vrednosti, kao i ranije.

Zbog specifičnosti ove varijante, opis koji unesemo se odnosi samo na taj jedan element. Zato je pisanje inline stilova veoma nepraktično, a još je teža kasnija izmena.

Pogledajmo primer:

```
...
<p>Prvi paragraf</p>
<p style="font-family:Tahoma; font-size:12px;">Moj paragraf</p>
<p>Treći paragraf</p>
...
```

Ovako smo dobili stilizaciju nad samo jednim (drugim) paragrafom teksta. Ako bismo želeli da svi budu stilizovani, morali bismo i njima da dodamo atribut style i ova svojstva i vrednosti. Lakše je postaviti eksterni/interni CSS i jednim opisom stilizovati paragrafe.

Možda se u ovom trenutku pitate kako ću stilizovati samo jedan od paragrafa, ako je to ono što želim, tj. želim da se jedan razlikuje. U takvom i sličnim slučajevima ipak nećemo koristiti inline silizaciju, već nešto složenije selektore od jednostavnog **p**, kao što ćemo videti u nastavku.

Tipovi CSS selektora

Postoje razni tipovi selektora i pomoću njih možemo *gađati* skoro svaki element na stranici. Na nama je da odredimo koji ćemo način koristiti u zavisnosti od situacije. I ono što je zanimljivo jeste da ne postoji jedan pravi način, već barem nekoliko njih.

*U prethodnom pasusu sam koristio reč **gađati**, kada govorimo o selektorima. Ovo ćete često čuti u žargonu, odnosno u govoru CSS stručnjaka. Iako bi pravilnije bilo reći nešto poput - **ovaj opis stilizuje element koji je određen selektorom x**, češće se koristi - **ovaj opis gađa element x**. Ovo se ustalilo kao nepisano pravilo jer potiče od toga da selektori izdvajaju samo određene elemente u moru ostalih i gađaju samo njih.*

Pogledajmo tabelu u kojoj su predstavljeni tipovi selektora:

Tip selektora	Primer	Objašnjenje
Univerzalni	*	Gađa sve elemente na stranici
Opšti (tipski)	Body, p, h2	Gađa <body>, kao i sve <p>i <h2> elemente na stranici
ID[†]	#primer	Gađa element sa označenom ID vrednošću.

CLASS[‡]	.primer	Gađa element sa označenom CLASS vrednošću.
Nasledni (descendant)	h1 a	Gađa sve <a> elemente koji se nalaze unutar <h1>, čak iako ima elemenata između njih.
Dete (child)	h1>a	Gađa sve <a> elemente koji se nalaze unutar <h1>, ali samo one koji su direktno u njemu (nema elemenata između).
Srodni (sibling)	h1~p	Gađa sve <p> elemente koji slede posle <h1> elementa
Neposredni srodni (direct sibling)	h1+p	Gađa samo jedan <p> element koji je direktno posle <h1>

[†] *Podsećam, ID vrednost se u ID atributu HTML taga postavlja samo po imenu, dok se u CSS opisu piše sa prefiksom #*
[‡] *Podsećam, CLASS vrednost se u CLASS atributu HTML taga postavlja samo po imenu, dok se u CSS opisu piše sa prefiksom.*

Dodatna napomena za ID i CLASS

ID i CLASS selektore možemo kombinovati sa drugim selektorima tako što ćemo ih pisati zajedno. Na primer:

```
h1#primer {…}
```

bi gađao samo taj <h1> tag koji poseduje ID vrednost *"primer"*. Istovetno je i za CLASS, stim što se umesto # piše . (tačka).

Složeni selektori

Možemo i kombinovati tipove selektora i onda se situacija donekle komplikuje. Tako dobijamo *složene* selektore. Pogledajmo ovaj primer:

```
body div.mojaKlasa h1+p {
      font-style:italic;
}
```

Možete li zaključiti koji element će gađati ovaj CSS opis, tj. njegov selektor? Sa analizom počnite sa desne strane i možemo zaključiti sledeće: ovaj opis će gađati <p> element koji se nalazi neposredno posle <h1> elementa, a koji je pri tom unutar elementa koji poseduje CLASS vrednost „mojaKlasa", ukoliko se on nalazi unutar <body> elementa.
Verovatno deluje zbunjujuće, ali nemojte brinuti. Ovakvi ili komplikovaniji selektori su retki, ali i kada naiđete na njih uvek ih možete razložiti ako poštujete pravila. Uglavnom su selektori jednostavniji. I treba težiti ka jednostavnosti.

Pogledajmo pravila koja važe za selektore i uopšte CSS opise:

Slaganje CSS stilova (Cascade theory)

Videli smo da se CSS može postaviti na tri načina (poseban fajl, head sekcija HTML dokumenta ili inline na samom tagu) tako da može doći do poklapanja istih CSS vrednosti. Takođe, u okviru samih CSS fajlova ili CSS dela u okviru stranice, može doći do poklapanja. U takvim slučajevima postoje određena pravila koja se automatski primenjuju i određuju koji će CSS opis biti primenjen.

U slučaju da se CSS opisi poklapaju, browser će poštovati pravilo **bliskosti** (engl. *proximity rule*) koje određuje da će biti primenjen CSS opis koji je **najbliži** ciljnom elementu. U tom kontekstu je sledeća struktura i redosled:

1. Default vrednosti browsera
2. Eksterni CSS fajl
3. Interni CSS (u head delu dokumenta)
4. Inline CSS stil (u okviru samog taga)

To praktično znači da, ukoliko postoje sve četiri varijante stilizacije, browser će iskoristiti četvrtu.

Pogledajmo na primeru. Ukoliko smo u CSS fajlu postavili:

```
p {
    color:#F00; /*crvena boja*/
}
```

u head delu strane:

```
p {
    color: #3F0; /*zelena boja*/
}
```

a na samom tagu inline plavu boju:

```
<p style="color:#00F;">Ovo je neki tekst.</p>
```

Logično je da istovremeno tekst ne može biti crn (default), crven, zelen i plav, već će browser postaviti plavu boju, jer je njen CSS opis **najbliži** elementu.

Bitno je napomenuti i da ukoliko se nalaze dva ili više CSS opisa koja se odnose na isti ciljni elemenat, a sama svojstva se ne sukobljavaju, biće primenjena sva svojstva.

Pogledaćemo dva primera.

Ukoliko imamo CSS opis:

```
p {
    color:#F00;
}
```

ali i drugi opis:

```
p {
    font-family:Arial;
}
```

Iako se odnose na isti element, paragraf će biti formatiran u crvenoj boji, ali i u fontu Arial jer se dopunjuju.

Pravila nasleđivanja (Inheritance theory)

Sa druge strane, ukoliko imamo:

```
p {
    color:#F00;
}
```

ali i

```
p {
    font-family:Arial;
    color:#3F0;
}
```

Uzimaju se **sva svojstva koja se ne sukobljavaju, plus jedno od sukobljenih**. Paragraf će svakako biti formatiran u fontu Arial, ali formatiranje boje (crvena ili zelena) će zavisiti od pravila sukobljavanja koje smo pomenuli ranije.

Pravila potonjih elemenata (Descendant theory)

Ciljni elementi preko kojih dodeljujemo svojstva mogu kompleksniji i specifičniji u zavisnosti od elemenata iznad njih. Na primer:

```
h1 {
 color:#F90;
}
```

je stilizovao sve h1 elemente. Dok bi sledeći kod:

```
div h1 {
 color:#F90;
}
```

stilizovao samo h1 elemente koji se nalaze u okviru div elementa. Ukoliko npr. postavimo h1 direktno u body, ne bi bio pogođen, niti stilizovan ovim opisom jer nije u okviru nijednog diva.

Pravilo specifičnosti (Specificity theory)

Dalje, možemo još preciznije definisati odnos elementa kojeg ciljamo stilizacijom. Na pređašnjem primeru:

```
div h1 {
 color:#F90;
}
```

ukoliko napišemo:

```
div.mojaKlasa h1 {
 color:#F90;
}
```

stilizovaćemo sve h1 naslove koji se nalaze u div elementu sa klasom mojaKlasa.

Ukratko, ako postavimo razmak, ciljamo na elemenat ispod; ako nema razmaka, ciljamo na element u istom nivou (elemenat koji ispunjava oba uslova).

U ovom trenutku ste primili mnogo novih informacija sa kojima se niste susretali. Kada budete prošli kroz primere i malo provežbali, vratite se na ovu lekciju kako biste utvrdili ove detalje jer predstavljaju osnovu CSS jezika.

Dostupna svojstva u CSS-u

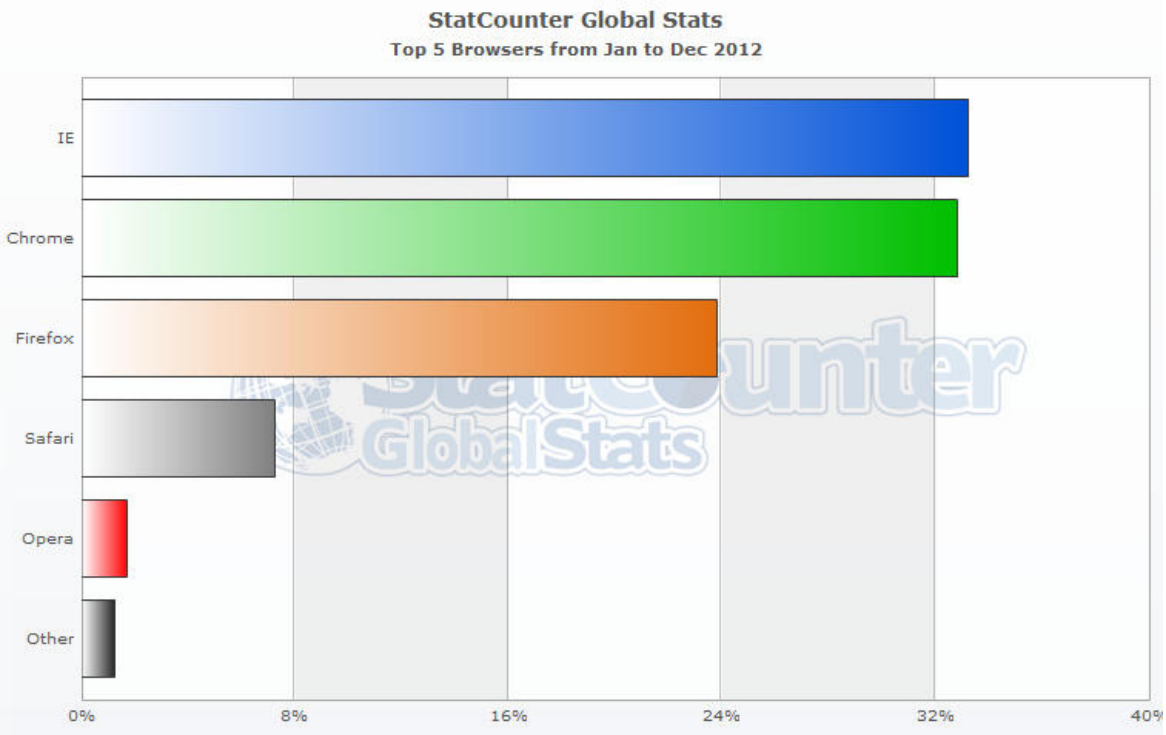
Kao što već znamo, pored selektora koji je jedna strana CSS opisa, moramo znati i koja svojstva i vrednosti možemo koristiti.

Uz ovu jedinicu je dat pdf dokument sa najčešće korišćenim svojstvima i kratkim objašnjenjem (fajl *CSS_podsetnik.pdf*). Preporučujem da taj dokument čuvate pri ruci (možete ga i odštampati) kako bi Vam služio kao referenca za buduće radove. Normalno je da nećete pamtiti napamet sva moguća svojstva jer ih, sa jedne strane, ima previše, a i sa druge, nepotrebno je. Ionako su uvek pri ruci na internetu i/ili u literaturi[2]. Ono što je bitno jeste znati kako ih primeniti i gde.

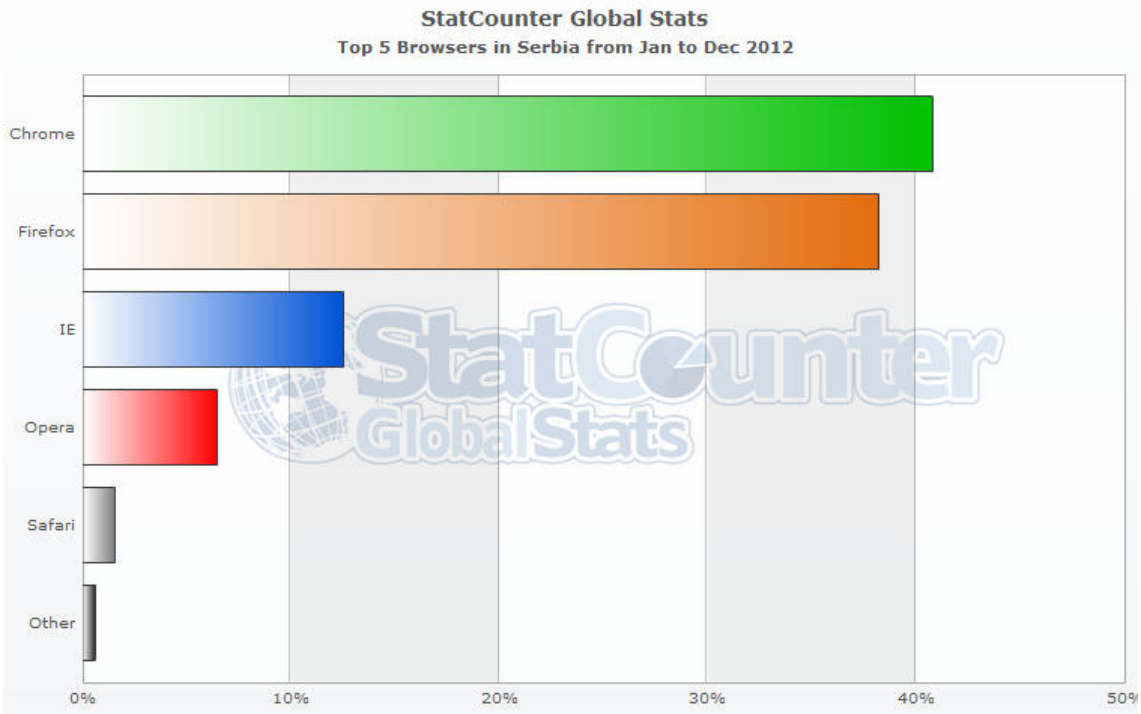
Prikaz u različitim browserima

Ukoliko pitate bilo kod iskusnog web developera, dizajnera ili uopšte osobu koja ima iskustva sa HTML i CSS-om, da li se CSS ponaša potpuno isto u različitim browserima, dobićete odričan odgovor. Kao i kod HTML-a, CSS je tekstualni dokument koji treba parsirati i prikazati, a to sve zavisi od mnogih detalja, a najviše od samog browsera. Već postaje čuvena priča o MS Internet Exploreru i njegovom (ne)poštovanju opšteprihvaćenih standarda. IE i dalje kasni drastično za ostalim modernim browserima (Chrome, Opera, Firefox). Ono što mi, kao kreatori weba, moramo raditi je to da naše sajtove testiramo na različitim browserima, kako bismo bili sigurni da se sve prikazuje normalno. Nije dovoljno napraviti sajt koji se idealno prikazuje u npr. Chrome pretraživaču, a zanemariti to što je sve pogrešno u Internet Exploreru. Iako lično možda ne koristite IE, veliki broj ljudi ga i dalje koristi. Pogledajte globalnu statistiku i statistiku za Srbiju u vezi sa korišćenjem browera na sledećim grafikonima:

[2] Više o dostupnim CSS svojstvima i vrednostima možete naći na internetu na adresama: http://reference.sitepoint.com/css/propertyref ili https://developer.mozilla.org/en-US/docs/CSS/CSS_Reference. Preporučena štampana literatura je data u pregledu kursa.



Globalna statistika upotrebe različitih browsera za 2012.
Izvor slike i podataka - http://gs.statcounter.com



Statistika upotrebe različitih browsera u Srbiji za 2012.
Izvor slike i podataka - http://gs.statcounter.com

Pre početka rada na sajtu, morate odrediti (ili se dogovoriti sa naručiocem posla) koje ćete browsere podržati, da bi znali na koje funkcije možemo računati, jer jednostavno neke starije verzije ne podržavaju novije funkcije. Na primer, ako vlasnik sajta zahteva da se sajt može normalno otvoriti i u Internet Exploreru 6, slobodno zaboravite na sve novije funkcije, kao i na HTML5, CSS3 i tako dalje.

Pitanje u jedinici

CSS jezik
- Koristi sve tagove iz HTML jezika
- Koristi neke tagove iz HTML jzika
- **Ne koristi HTML tagove, već CSS opise**

Pitanje ostaje kako ispraviti situaciju ako se u jednom browseru nešto prikazuje normalno, a u drugom ne. Krenućemo sa proverom po sledećim koracima:

- **Da li postoji neka greška u našem HTML i/ili CSS kôdu**. Ukoliko browser naiđe na nezatvoren tag, ukrštene tagove, loše napisan CSS opis i tako dalje, on će pokušati da pretpostavi šta bi trebalo da stoji i tu nastaju problemi. U većini slučajeva je ovde problem, a ne u samim browserima.
- **Da li browser u kome se javlja problem podržava to što želimo da postignemo?** Ovaj detalj smo već pomenuli u ranijem pasusu. Ne podržavaju svi browseri sve. Na primer, IE verzije 6 ne podržava transparentne PNG fajlove. U svim ostalima će se takav PNG prikazati normalno, osim u IE6. Ili možemo zameniti te PNG slike drugačijima, ili zanemariti IE verzije 6, ali to je već stvar planiranja.
- Ukoliko smo prošli prethodne dve stavke i naš problem nije vezan za njih, onda smo naišli na CSS ***browser bug***, odnosno ***browser quirk.*** U tom slučaju, najbolje je potražiti odgovor i rešenje na internetu, jer skoro sigurno je neko pre Vas imao isti ili barem sličan problem i rešenje podelio sa drugima na netu.

Najvažnije iz lekcije

- CSS je opisni jezik
- CSS se koristi za stilizaciju HTML sadržaja
- HTML služi za prikaz sadržaja, a CSS za njegovu stilizaciju.
- CSS jezik je sastoji iz opisa. Svaki opis je sastavljen iz selektora i deklaracija. Deklaracija se sastoji iz svojstva i vrednosti.
- Možemo postaviti eksterni, interni ili inline CSS.
- Postoje različiti tipovi selektora i načina „gađanja" elemenata.
- CSS stilovi se slažu po određenim pravilima.
- Ne prikazuju svi browseri istovetno naše stranice.