Sadržaj

CSS3

1. Uvod u CSS

CSS je tehnologija koja omogućava razdvajanje strukture stranice od njenog izgleda, što predstavlja osnovni princip ove tehnologije. To konkretno znači da informacije koje želimo da prikažemo na stranici treba da budu sadržane u HTML fajlu, a opis izgleda stranice i načina predstavljanja informacija treba da se nalazi u drugom fajlu – css fajlu. Odrednica CSS (*Cascading Style Sheets*) u prevodu znači “Kaskadne Liste Stilova”.

Neke od mogućnost CSS-a su:

- Pozicionira sadržaj stranice
- Definiše tip i veličinu fonta
- Možemo da definišemo izgled linkova na način kako nama odgovara (ne moraju da budu podvučeni i u plavoj boji, već u stilu koji mi zadamo).
- Mogućnost redefinisanja postojećih tagova u HTML-u. Npr. ako želimo da se tag koji inicijalno prikazuje tekst podebljanim slovima, prikaže u crvenoj boji sa veličinom fonta od 16 piksela, to sa CSS-om vrlo lako možemo da uradimo.
- Definišemo izgled stranice/a na jednom mestu kako bismo izbegli ponavljanje koda na stranici.
- Jednostavno i brzo menjamo izgled stranice čak i nakon što smo je kreirali.

2. CSS sintaksa

CSS fajl se sastoji od niza instrukcija koje nazivamo pravilima. Jedno pravilo se sastoji od tri elementa: selektora (selector), svojstva (property), vrednost svojstva (value) :

selektor { nazivSvojstva: vrednostSvojstva; } ,

gde selektor označava element nad kojim primenjujemo stil, svojstvo (osobina) može da bude *color, padding, margin, background* itd, vrednost predstavlja vrednost svojstva. Primer:

```
p { color: #FFFFFF; }  ,
```

ovo pravilo kaže da će tekst svih paragrafa biti bele boje.

Za jedan selektor može da se definiše i više svojstava sa vrednostima. To činimo tako što u okviru vitičastih zagrada navodimo parove „svojstvo:vrednost“ jedan za drugim. Svaki par mora da se završi znakom „ ; “ (tačka-zarez).

```
p { color: #FFFFFF;  font-size:18px;  font-family:'Helvetica'; }
```

Skup svih svojstava i njihovih vrednosti naziva se deklaracija. Vitičaste zagrade označavaju početak i kraj deklaracije.

Evo nekoliko pravila koje treba poštovati pri pisanju lista stilova kako ne bi došlo do neočekivanih efekata u radu sa stilovima:

1. Svako pravilo mora imati selektor i deklaraciju. Deklaracija dolazi odmah iza selektora i oivičena je vitičastim zagradama.
2. Deklaraciju čine jedan ili više svojstava odvojenih “ ; ” znakom.
3. Svako svojstvo ima ime iza koga je dvotačka a zatim ide vrednost za to svojstvo. Postoji veliki broj različitih vrednosti za svojstva ali svako svojstvo može primiti samo vrednost koja mu je po specifikaciji propisana.
4. Nekada svojstvo može dobiti više vrednosti kao što je to slučaj sa *font-family* svojstvom. Višestruke vrednosti moraju biti odvojene zarezom i praznim prostorom.
5. Vrednost koja se sastoji od više reči treba da se nalazi pod navodnicima.
6. Između vrednosti svojstva i jedinice u kojoj se vrednost navodi ne sme da postoji prazan prostor.

```
Primer:  font-size: 1em; - Ispravno

         font-size: 1 em; - Neispravno
```

7. Kao i kod HTML-a, dodatni prazan prostor se ignoriše pa se može koristiti samo da bi se olakšalo čitanje koda.
8. Selektori se mogu grupisati tako što će se navesti jedan za drugim, odvojeni zarezom. U primeru su grupisani svi elementi za naslove u HTML-u i svakom od njih dodeljena je osobina da su slova ispisana zelenom bojom.

```
h1, h2, h3 { color: green;}
```

3. Načini primene stilova

Stilovi se mogu primeniti na jednu HTML stranicu na tri načina:

- *Inline* - direktno u određenom HTML tagu koji se želi formatirati. (interno)
- Unutar stranice u *head* delu HTML koda u okviru <style> taga. (interno)

- Eksterno - povezivanjem HTML stranice i eksternog fajla sa ekstenzijom *.css* (u kome su definisana pravila)

3.1. Inline stil

Stil se unosi direktno u otvarajući tag elementa. Za ovakvu internu primenu stila koristi se atribut *style* kojeg mogu posedovati svi HTML elementi. Primer:

```
<h1 style="color: #FF00FF; padding:20px;"> Primenjen je inline stil.</h1>
```

Ovakav način zadavanja stila se sve ređe koristi. Razlog tome je dosta ponovljenog koda na stranici, jer ako npr. Imamo više paragrafa, a za svaki paragraf hoćemo da važi isti stil, moramo unutar svakog <p> taga da unesemo isti kod za taj stil.

3.2. Interno unutar head dela stranice

Stil se piše u head delu HTML stranice unutar tagova <style></style>. Primer:

```
<head>
	<title>Moja prva HTML stranica</title>
	<style>
		h2 {
		color: #FF00FF;
		padding: 20px;
		}
	</style>
</head>
```

Ovakav način dodavanja stila otklanja prethodni nedostatak inline stilova, jer stil koji se navede u head delu važiće za sve <h2> elemente na stranici. Međutim, i ovakav stil ima svojih nedostataka:

- Šta ako imamo više stranica i na svim želimo isto pravilo za sve <h2> elemente? To je nemoguće postići na ovaj način. Morali bismo u svaku stranicu da unesemo isti stil, a to je dosta ponavljajućeg koda i utrošenog vremena.
- Narušava se pravilo odvajanja sadržaja stranice od definicije izgleda.

3.3. CSS kao eksterni fajl

Osnovni princip CSS filozofije glasi: „Odvojiti sadržaj od izgleda stranice“. Eksternim načinom primene stila, ovaj princip je zadovoljen! Naime, sva CSS pravila se definišu u jednom posebnom *.css* dokumentu. Nikakvi stilovi nisu potrebni u html kodu. Sve što je potrebno je povezivanje *.css* fajla sa *html* dokumentom. To se radi pomoći <*link*> taga u head delu *html* dokumenta.

```
<link rel="stylesheet" type="text/css" href="stilovi.css" />,
```

gde *rel* atribut definiše vezu između trenutnog i eksternog dokumenta, *type* govori o tipu eksternog dokumenta, *stilovi.css* je naziv dokumenta u kojem se čuvaju stilovi i koji se nalazi u istom folderu u kojem je i html dokument.

Ovaj način pridruživanja stilova stranici je strogo preporučen s obzirom na to da omogućava veliku fleksibilnost prilikom promene stila stranice i primene jednog istog stila na više različitih stranica. Znači, ako se u eksternom CSS fajlu napiše da pozadina stranice bude bela, ona će biti bela na bilo kojoj stranici koja je sa ovim fajlom prethodno bila povezana, i ako nakon toga u istom CSS fajlu promenimo boju pozadine u crnu ona će istovremeno postati crna u svim tim stranicama makar ih bilo i 1000. Ovakva dinamičnost može se postići i kada su u pitanju fontovi, boje, linkovi, paragrafi, liste, tabele, hederi, pozicije, dimenzije...

4. Prioritet stilova

Šta će se desiti ako za jedan html element definišemo dva stila – jedan interni i jedan eksterni (pri čemu su selektori tagovi elementa)? Ispoštovaće se pravilo prioriteta, koje govori da interni stil ima veći prioritet, naročito inline stil. Npr. Za <p> element je definisan eksterni stil:

```
p {
   color:#FF00FF;
   font-size:20px;
}
```

I interni inline stil

```
<p style=“color: green;“> Moj paragraf</p>.
```

U ovom slučaju pravila se sabiraju, a krajnji rezultat je stil:

```
p {
    color:green;
    font-size:20px;
}
```

Kako su navedene dve boje, biće primenjena ona iz inline stila, a font-size svojstvo se preuzima iz eksternog fajla.

Uopšteno pravilo glasi:

Kada za jedan element postoji više definisanih stilova onda će biti primenjeni stilovi po sledećem prioritetu,

1. Inline stil
2. Ugrađena lista stilova unutar zaglavlja stranice
3. Eksterna lista stilova

pri čemu prvi način definisanja ima najveći prioritet jer je najbliži elementu.

Takođe, ukoliko koristimo više eksternih css fajlova i sve povezemo sa html dokumentom, prioritet će imati stil koji je poslednji u nizu <link> tagova.

5. Selektori

Kao što smo već rekli, selektori se koriste za selekciju HTML elemenata pomoću tipa, id-a, klase, atributa itd.. Opisaćemo tri vrste selektora: *type*, *class*, *id* selektor.

5.1. Type selektor

Type selektor se odnosi na svaki tag na stranici čiji naziv odgovara nazivu selektora. Npr. *type* selektor „p“ se odnosi na svaki <p> tag koji postoji na stranici, „body“ selektor se odnosi na <body> tag,...

5.2. Class selektor

Šta da uradimo kada jedno pravilo želimo da primenimo na samo tri <p> elementa, a ne na sve koji se nađu na stranici? Rešenje se nalazi u korišćenju *class* atributa. Ovaj

atribut se dodaje odabranim <p> elementima, sa istom vrednošću za sva tri elementa. Npr. Izgled <p> taga za sva tri elementa je sledeći:

```
<p class="crveno">...</p>,
```

CSS kod je:

```
.crveno{
    color:red;
}
```

Dakle, samo elementi sa vrednošću klase „crveno" imaće tekst obojen u crveno. Takva klasa može da se doda i elementima nekog drugog tipa.

U css-u element se poziva preko klase tako što se stavi tačka, pa zatim vrednost *class* atributa (.crveno).

Naziv klase ne sme početi brojem.

U slučaju da elementi različitog tipa imaju iste vrednosti *class* atributa i samim tim neke zajedničke stilove, kada želimo da dodamo ili promenimo svojstvo na samo jednom tipu elementa, tada u css-u kreiramo novo pravilo koje će važiti samo za taj tip elementa.

```
p.crveno {
    padding:10px;
}
```

Ovim smo padding svojstvo dodelili <p> elementima, a ne i drugim elementima koji imaju vrednost klase "crveno".

5.3. ID selektor

Za razliku od *class* atributa, *id* atribut sa jednom vrednošću može se u unutar html dokumenta pojaviti samo jednom. Recimo, class="crveno" atribut se može vezati za više različitih elemenata, dok se id="crveno" može vezati za samo jedan element na stranici. Ovo je naročito značajno kada za jedan element želimo da definišemo posebno pravilo.

U css-u element se poziva preko vrednosti „id" atributa na sledeći način:

```
#crveno {
```

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

gde je “crveno” vrednost *id* atributa. Najpre se navodi # (taraba) pa vrednost atributa.

Naziv id-a ne sme početi brojem.

6. Svojstva

6.1. Formatiranje teksta

U nastavku ovog poglavlja naveščemo svojstva i njihove moguće vrednosti koji se koriste za podešavanje načina prikaza teksta na stranici.

color

Ovo svojstvo se koristi za podešavanje boje teksta. Prethodno smo pričali o bojama i njihovim vrednostima (heksadecimalni zapis, rgb zapis i naziv boje). Primer:

```
p { color: rgb(0,255,0); }    ili    p { color: #00FF00; }    ili    p{ color: green; }
```

Ako ne navedemo ovo svojstvo, podrazumevana boja za tekst je crna. Takođe, ukoliko želimo da boja teksta na celoj strani bude ista, a da to nije crna, potrebno je da <body> tagu dodelimo color svojstvo sa željenom vrednošću za boju.

text-align

Koristi se za definisanje horizontalnog poravnanja teksta (levo, desno, centralno, poravnanje sa leve i desne strane istovremeno - justify). Pa su moguće vrednosti ovog svojstva: left, right, center, justify. Primer:

```
p { text-align: left;
    color: green; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
p { text-align: right;
    color: green; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
p { text-align: center;
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
    color: green; }
```

```
p { text-align: justify;
    color: green; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

text-transform

Ovo svojstvo koristimo kada želimo sva slova u tekstu da pretvorimo u velika, u mala ili da prvo slovo svake reči bude veliko. Koriste se tri vrednosti ovog svojstva, to su: *capitalize, uppercase, lowercase*. Kada imamo postavljeno ovakvo svojstvo, ne moramo da brinemo da li smo uneli pravilnu veličinu slova, css će to ispraviti za nas. Primer:

```
p { text-transform: uppercase; }
```

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT UT LABORE ET DOLORE MAGNA ALIQUA.

```
p { text-transform: lowercase; }
```

lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
p { text-transform: capitalize; }
```

Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Incididunt Ut Labore Et Dolore Magna Aliqua.

text-decoration

Koristi se kada želimo da dodamo ili skinemo ukrase sa teksta. To mogu da budu: podvlačenje teksta linijom, linija iznad teksta, precrtavanje teksta (linija po sredini teksta). Ovo svojstvo se najviše koristi kod linkova koji su po default-u podvučeni. Ako želimo da uklonimo ovaj ukras sa linka, potrebno je da vrednost svojstva postavimo na „none“. Primer:

```
a{ text-decoration: none; }
```

Na ovaj način smo uklonili podvlačenje linkova.

text-indent

Definiše uvučenost prvog reda u tekstu (pasusu). Uvlačenje se može izraziti u px, em, pt, cm, %. Primer:

```
p { text-indent: 20px; }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

word-wrap

Ukoliko je reč dovoljno velika i izlazi iz okvira oko teksta, kada navedemo ovo svojstvo za ceo taj pasus, cela reč i tekst posle će preći u novi red. Primer:

```
p { word-wrap: break-word; }
```

word-break

Specificira kako će doći do preloma teksta u novi red. Moguće vrednosti su: *break-all* i *keep-all*. “break-all“ vrednost prelama celu reč u novi red, ako reč ne može da stane u okvir. Kod „keep-all“ vrednosti kada reč prevaziđe okvire, slova koja izlaze van okvira se prelamaju u novi red.

text-shadow

Koristi se za dodavanje senke slovima. Ovo svojstvo ima četiri vrednosti od kojih su dve obavezne. Obavezno se mora navesti horizontalno i vertikalno prostiranje senke, a *blur* efekat i boja senke nisu obavezni. Primer:

```
p { text-shadow: 1px 2px 5px #FFFFFF; color:#000000; }
```

Uzmimo da je tekst ovog paragrafa crne boje. Prva vrednost je horizontalna senka (prostiranje po x osi, dozvoljene su negativne vrednosti), druga vrednost (2px) je vertikalna senka (prostiranje po y osi, dozvoljene su negativne vrednosti). 5px označava *blur* efekat (zamagljena senka, bez oštrih ivica). Boja senke je bela.

6.2. Font

Font je kolekcija brojeva, simbola i znakova. Fontovi se klasifikuju u dve grupe: serifni i sans-serifni fontovi.

Serifni font. Slova pisana serifnim fontom imaju male linije (kukice) na svojim krajevima. Ova vrsta fonta se najviše koristi za naslove, jer se takva slova malo teže čitaju. Ovakvi fontovi su: Georgia, Times New Roman, Playfair Display,...

Sans-serifni font. Slova pisana fontom koji pripada sans-serifnoj grupi su češće korišćena jer su slova čista, bez kukica na krajevima i lako se čitaju. Primer fontova: Open Sans, Roboto, Arial, Verdana,...

Pomenućemo osnovna *font* svojstva, a to su: *font-family, font-size, font-style, font-weight*

font-family

Tip fonta se postavlja pomoću ovog svojstva. Vrednost je naziv fonta. Obično se navodi više vrednosti, jedna za drugom odvojene zarezima, i to za slučaj da browser ne podržava prvi font. Tada prelazi na sledeći. Primer:

```
p {font-family: Verdana, Arial, “Times New Roman”, sans-serif; }
```

Ako browser ne podržava Verdana font, prelazi na Arial itd, sve dok ne dođe do sans-serif vrednosti, tada može sam da izabere jedan sans-serifni font koji podržava.

Uvek prvo navodimo font koji najviše želimo da vidimo na stranici.

font-size

Ovim svojstvom podešavamo veličinu teksta (slova). Primer:

```
p { font-size: 20px; }   - svi paragrafi će imati veličinu fonta 20px.
```

Podrazumevana veličina fonta na stranici je 16px.

font-style

Ovo svojstvo koristimo kada želimo da iskosimo tekst (*italic*). Primer:

```
p { font-style: italic;
    font-size: 20px;
}
```

font-weight

Svojstvo koristimo za podešavanje debljine slova. Moguće vrednosti ovog svojstva su: *normal, bold, bolder, lighter*, brojevi (100-900). Ako se odlučimo da debljinu fonta

izrazimo brojem tada možemo da koristimo stotine od 100 do 900. Normalnoj debljini fonta odgovara broj 400, a podebljanoj (*bold*) odgovara broj 700. Primer:

```
p { font-weight: 300;
    font-size: 20px;
}
```

Debljina fonta od 300 je tanja od normalne.

@font-face pravilo

U praksi se koriste *web safe* fontovi. To su fontovi koji dolaze sa instalacijom operativnog sistema i nalaze se na računarima. Kada korisnik poseti sajt sa tekstom prikazanim nekim *web safe* fontom, gotovo je sigurno da neće doći do greške i da će se tekst lepo prikazati.

CSS3 uvodi novo pravilo *@font-face* koje nama kao developerima daje mogućnost da kreiramo sopstveni font koji nije *web safe*. Potrebno je da pronađemo fajlove željenog fonta i smestimo ih na web server (u okviru projekta), potom da fajlove povežemo sa našim css kodom i nove fontove primenimo na stranici.

Prednost ovakvog pravila je ta što možemo da koristimo font koji poželimo, a da ne brinemo kako će se prikazati u browser-u posetioca. Jer se takav font automatski prebacuje u browser korisnika.

U css-u sve što je potrebno da uradimo je sledeće:

```
@font-face {
   font-family: mojFont;
   src: url(playfair_display.ttf);
}

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

U okviru *@font-face* pravila definišemo sopstveni naziv fonta (mojFont). “src” svojstvo sadrži url font fajla koji se nalazi na serveru. Na ovaj način smo kreirali sopstveni font. Sve što je dalje potrebno jeste pozivanje fonta na odgovarajućim mestima u dokumentu. U primeru, svi paragrafi na stranici će imati mojFont.

Koriste se različiti formati font fajlova: ttf, otf, woff, svg, eot.

6.3. Pozadine

Background (pozadina) jednog elementa može da bude u određenoj boji ili slika. Navešćemo svojstva koja se koriste za postavljanje pozadine:

Pozadina u boji

Koristi svojstvo *background-color* čija je vrednost boja zadata nazivom, heksadecimalnom vrednošću, rgb ili rgba zapisom. O bojama je bilo više reči u HTML odeljku. CSS3 verzija uvodi još jedan zapis a to je rgba (*red, green, blue, alpha).* Vrednost *alpha* označava transparentnost boje, moguće su vrednosti od 0 do 1 (decimalne vrednosti se pišu sa tačkom, npr. 0.7). Nula predstavlja potpunu transparentnost (providnost), dok 1 označava punu boju (ovo je default vrednost).

Primer:

```
div.pozadina {
background-color: rgba (255, 0, 0, 0.5);          /* crvena poluprovidna pozadina */
ili
background-color: rgb (255, 0, 0);     /* crvena boja, rgb zapis */
ili
background-color: #FF0000;      /* crvena boja, heksadecimalni zapis */
ili
background-color: red;         /*crvena boja, naziv boje*/
}
```

Pozadina se dodaje svim *div* elementima sa klasom “pozadina”.

Umesto *alpha* vrednosti u rgba zapisu možemo da koristimo i svojstvo “opacity” i to na sledeći način:

```
div.pozadina {
background-color: rgb(255, 0, 0);
opacity: 0.5;
}
```

Slika u pozadini

Koristi svojstva: *background-image, background-repeat, background-position, background-attachment.*

background-image - navođenje putanje do slike

```
body {
background-image: url(slike/slika.jpg);
}
```

background-repeat - da li će se slika ponavljati ili će se prikazati samo jednom u pozadini elementa. Moguće vrednosti ovog svojstva su: *no-repeat* (slika se ne ponavlja*), repeat-x* (ponavlja se po horizontali), *repeat-y* (ponavlja se po vertikali), *repeat* (ponavlja se i po vertikali i po horizontali). Ukoliko ne navedemo ovo svojstvo, a navedemo *background-image*, onda je podrazumevana vrednost *repeat*.

Primer:

```
body {
background-image: url(“slike/slika.jpg”);
background-repeat: no-repeat;
}
```

background-position – navođenje pozicije gde će se slika nalaziti u elementu. Ukoliko ne navedemo ovo svojstvo, podrazumevana vrednost je gornji levi ugao elementa. Ovo svojstvo prihvata dve vrednosti - horizontalna i vertikalna pozicija.

Primer:

```
body {
background-image: url(“slike/slika.jpg”);
background-repeat: no-repeat;
background-position: right top;
}
```

Slika će se prikazati u desnom gornjem uglu. Prva vrednost je horizontalno pozicioniranje i moguće vrednosti su: *left, center* i *right*. Druga vrednost je pozicioniranje po vertikali elementa i moguće vrednosti su: *top, center* i *bottom*.

Takođe, horizontalne i vertikalne vrednosti mogu da budu izražene u pikselima ili procentima. Npr.

```
background-position: 100px 40px;    ili    background-position: 20% 50%;
```

background-attachment – da li se slika pomera sa skrolom strane ili ostaje fiksirana. Moguće vrednosti su: *fixed, scroll, local*. Podrazumevana vrednost je *scroll.*

Primer:

```
body {
        background-image: url("slike/slika.jpg");
        background-repeat: no-repeat;
        background-attachment: fixed;
       }
```

background – svojstvo objedinjuje sva prethodno navedena background svojstva.

Primer:

```
background: #FF0000 url("slike/slika.jpg") no-repeat fixed right top;
```

Bitan je redosled navođenja svojstava:

1. boja 2. slika 3. ponavljanje 4. pomeranje pri skrolu 5. pozicija slike

6.4. Okvir (border)

Okvir sa oštrim ivicama

Za postavljanje okvira oko određenog elementa pridružujemo mu svojstva koja se odnose na stil, debljinu i boju okvira.

border-style – vrsta okvira (puna linija, isprekidana linija, tačkice,...). Moguće vrednosti su: *solid, dotted, dashed, double, groove, ridge, inset, outset, hidden, none.*

Primer:

```
p {
border-style: solid;
}
```

border-width - debljina okvira. Vrednosti se izražavaju u pikselima ili korišćenjem jedne od vrednosti: *thin, medium, thick.*

Primer:

```
p {
border-style: solid;
border-width: 2px;
}
```

Ovo svojstvo ne možemo koristiti ako prethodno ne navedemo svojstvo *border-style*.

border-color – boja okvira.

Primer:

```
p {
border-style: solid;
border-width: 2px;
border-color: #FF0000;
}
```

Kao što smo imali *background* kao skraćeno svojstvo, tako i ovde imamo *border* svojstvo u okviru kojeg navodimo debljinu, stil i boju (vodimo se ovim poretkom). Takođe, moramo navesti stil, ostale vrednosti su opcione.

Primer:

```
p {
border:  2px solid #FF0000;
}
```

Okvir se sastoji od četiri strane: gornja, desna, donja i leva. Za svaku stranu možemo navesti posebna svojstva.

Primeri:

```
1. border-left-style: dotted;
2. border-bottom: 1px solid #FF0000;
3. border-top-style:solid;
   border-top-color: #FF0000;
```

Okvir sa zaobljenim ivicama

Za definisanje zaobljenih ivica koristi se svojstvo *border-radius.*

Primer:

```
p {
border:  2px solid #FF0000;
border-radius: 20px;
}
```

U primeru sve ivice okvira oko paragrafa biće zaobljene za 20px.

Ukoliko želimo da ivice budu različito zaobljene koristimo svojstvo za svaku ivicu i to: *border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius.* Ili skraćeno svojstvo sa vrednostima za svaku ivicu:

Primeri:

```
border-radius: 20px 25px 15px 10px;     /*20px gornja leva, 25px gornja desna, 15px donja desna, 10px donja leva */
border-radius: 20px 15px 10px;  /* 20px gornja leva, 15px gornja desna i donja leva, 10px donja desna */
border-radius: 20px 15px;  /*20px gornja leva i donja desna, 15px gornja desna i donja leva */
```

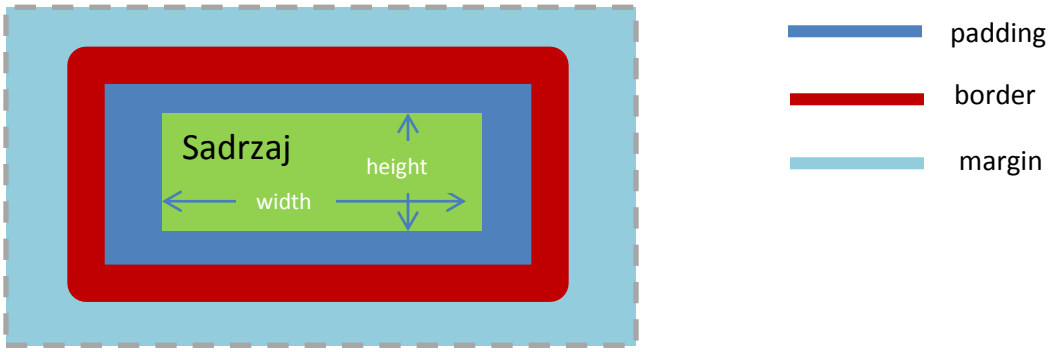
border-radius svojstvo možemo dodati i elementima koji nemaju okvir već pozadinu u boji ili sliku kao pozadinu.

7. Box model

Sve HTML elemente možemo posmatrati kao kutije u kojima se nalazi određeni sadržaj. Dimenzije jednog elementa ne čine samo širina i visina već i druga svojstva koja se nalaze oko sadržaja elementa. Jedan box model čine: širina i visina (sadržaj elementa), padding, border , margine.

širina elementa = levi border + levi padding + širina sadržaja + desni padding + desni border

visina elementa = gornji border + gornji padding + visina sadržaja + donji padding + donji border



7.1. Width i height svojstva

Unutrašnje dimenzije elementa definišemo pomoću svojstava *width* i *height*, gde *width* označava širinu a *height* visinu elementa.

Primer:

```
p {
width: 400px;
height: 300px;
color: #000000;
}
```

Takođe, postoje i svojstva za definisanje maksimalne i minimalne širine i visine. To su: *max-width, min-width, max-height, min-height.*

7.2. Padding

Padding svojstvo definiše razmak između border-a i sadržaja elementa. Tačnije, govori koliko će sadržaj biti udaljen od okvira elementa. Koristi se svojstvo *padding*. Postoji gornji, desni, donji i levi *padding*.

Primer:

```
p {
padding: 20px;
width: 400px;
height: 300px;
}
```

Na ovaj način smo zadali isti *padding* sa svih strana elementa. Ako želimo različite *padding* vrednosti za različite strane elementa to možemo učiniti na jedan od dva načina:

1. Koristimo *padding-top, padding-right, padding-bottom, padding-left* svojstva i zadajemo im željene vrednosti.

Primer:

```
p {
padding-left: 20px;
padding-right:20px;
padding-top: 10px;
padding-bottom: 15px;
}
```

2. Koristimo skraćeno *padding* svojstvo.

Primer:

```
- padding: 20px 20px 10px 15px; /* 20px gornji, 20px desni, 10px donji, 15px levi */
- padding: 20px 10px 15px;  /* 2opx gornji, 10px desni i levi, 15px donji */
- padding: 20px 10px;  /* 20px gornji i donji, 10px desni i levi */
- padding: 20px;  /* 20px gornji, desni, donji i levi */
```

7.3. Margine

Margina je prazan i transparentan prostor oko elementa. Postoje gornja, desna, donja i leva margina.

Primer:

```
p {
margin: 10px;
padding: 20px;
width: 400px;
height: 300px;
}
```

Kao i kod *padding* svojstva i za margine važe ista pravila pri zadavanju pojedinačnih vrednosti za levu, desnu, gornju i donju (*margin-left, margin-right, margin-top, margin-bottom*).

Za razliku od *padding*-a, margine mogu da imaju negativne vrednosti.

7.4. Box sizing svojstvo

„box-sizing“ je značajno CSS3 svojstvo. Uzmimo dva elementa koji imaju iste vrednosti za *width* i *height*. Ako drugom elementu dodamo svojstvo „padding: 10px;“, dimenzije ova dva elementa neće biti iste iako imaju istu širinu i visinu. Tačnije, dimenzije drugog elementa su: ukupna širina=width+20px, ukupna visina= height+20px.

Primer:

```
div {
width:200px;
height:250px;
padding:10px;
}
```

Ukupna širina ovog elementa je 220px, dok je ukupna visina 270px.

Da bi se rešio ovaj problem uvedeno je *box-sizing* svojstvo. Ovo svojstvo ne dodaje *padding* i *border* vrednosti na definisanu širinu i visinu sadržaja, već od definisanih *width* i *height* vrednosti odvaja za *border* i *padding.*

Moguće vrednosti ovog svojstva su: *content-box, padding-box, border-box.* „content-box“ je default vrednost definisana od strane ranijih css verzija*. padding* i *border* se dodaju na definisanu širinu i visinu. „padding-box“ u *width* i *height* vrednosti uključuje padding vrednosti , dok se *border* vrednost dodaje.

Najznačajnija i najčešće korišćena vrednost je „border-box“. Kada navedemo ovu vrednost svojstva možemo biti sigurni da element zadržava definisanu visinu i širinu, bez obzira koliki *padding* ili *border* postavimo.

Primer:

```
div {
width:200px;
height:250px;
padding:10px;
border: 1px solid #FF0000;
box-sizing: border-box;
}
```

Ukupna širina elementa ostaje 200px, dok je ukupna visina 250px.

Praktikuje se definisanje box-sizing svojstva za sve elemente na stranici. Da bismo to postigli moramo definisati css pravilo:

```
*{
box-sizing: border-box;
}
```

Selektor “ * ” označava da se pravilo odnosi na sve elemente html dokumenta sa kojim je povezan dati css dokument. Ovde možemo navesti i druga svojstva koja želimo da se primene na svim html elementima.

8. Prikaz elemenata

Postoje dva tipa HTML elemenata. To su: blok i inline elementi. Blok elementi su recimo: <p>, , <h1>, <form>, <li>, <header>, <footer, <section>. Svi elementi čiji se sadržaj prikazuje u novom redu, formirajući prazan prostor iznad i ispod svog sadržaja. Inline elementi su, npr: <a>, , <b>,… Nižu se u istom redu, jedan za drugim.

Pomoću „display“ svojstva možemo menjati način prikaza elementa, tj. block element možemo prikazati kao inline i obrnuto. Ovim ne menjamo tip elementa, već samo njegov prikaz.

Moguće su tri vrednosti „display“ svojstva: *block, inline, inline-block*.

Block vrednost

Primer:

```
a {
display:block;
}
```

Iznad i ispod elementa se formira prazan prostor i ne dopušta postojanje drugih elemenata sa leve i desne strane.

Inline vrednost

Primer:

```
p {
display:inline;
}
```

Ovaj element se pojavljuje u istom redu sa drugim *inline* ili *inline-block* elementima. Ovakvom elementu ne možemo da zadamo *width* i *height* vrednosti. Može da poseduje levu i desnu marginu i *padding*, ali ne i gornju i donju marginu i *padding.*

Inline-block vrednost

Primer:

```
p {
```

```
display:inline-block;
}
```

Element sa ovom display vrednošću se prikazuje kao *inline*, a ponaša kao *block* element. Tačnije, prikazuje se bez prelaska u novi red (inline osobina), i možemo mu zadati *width, height*, sve *margin* i sve *padding* vrednosti (block osobina).

Pored ove tri vrednosti „display“ svojstvo može da ima i „none“ vrednost. Element kojem zadamo svojstvo sa ovom vrednošću, neće se prikazati na stranici. U istu svrhu možemo da koristimo i svojstvo „visibility“ sa vrednošću „hidden“. Element sa ovom vrednošću se neće prikazati, ali će za razliku od „display:none“ svojstva, ostati prazan prostor na mestu gde je element trebao da se prikaže.

9. Pozicioniranje elemenata

U radu ćemo opisati dva načina pozicioniranja elemanata na stranici:

1. pozicioniranje promoću „position“ i „left, right, top, bottom“ svojstava.

2. pozicioniranje pomoću „float“ svojstva

9.1. Statičko, relativno, apsolutno i fiksno pozicioniranje („position“ svojstvo)

Statičko pozicioniranje

Svi elementi na web stranici su po default-u statički. To znači da su prikazani u redosledu kako su navedeni u html dokumentu. Ovakvo pozicioniranje je najjednostavnije i retko se koristi. Primer:

Pretpostavimo da html dokument sadrži sledeći kod sa tri paragrafa:

```
<html>
<p> Prvi paragraf</p>
<p> Drugi paragraf</p>
<p> Treći paragraf</p>
</html>
```

rezultat →

Prvi paragraf

Drugi paragraf

Treći paragraf

U rezultatu vidimo da se paragrafi pojavljuju baš u onom redosledu kako su navedeni u html dokumentu.

Relativno pozicioniranje

Relativno pozicioniranje znači da se element pozicionira u odnosu na njegovu statičku poziciju, odnosno u odnosu na poziciju na kojoj se javlja u prirodnom toku (po default-u). Kada relativno pozicioniramo element kao da govorimo internet pretraživaču: 'Uzmi ovaj element i premesti ga 10 piksela niže i 10px desno od mesta gde bi trebao da stoji'. Navešćemo taj primer:

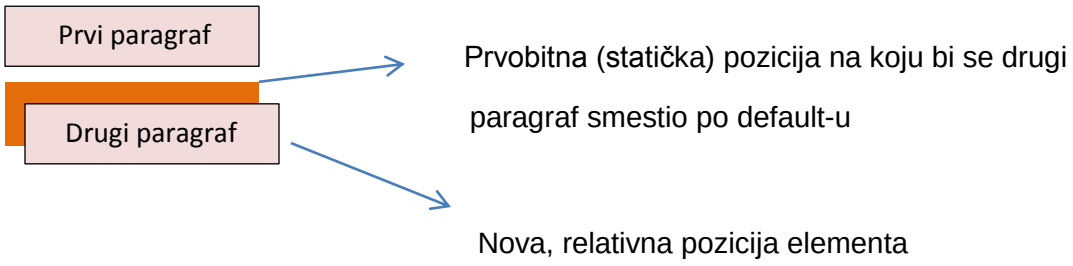
HTMl kod:

```
<html>
<p> Prvi paragraf</p>
<p id=”drugi”>Prvi paragraf</p>
</html>
```

Css pravilo:

```
p{
width:100px;
height:30px;
background-color:#FF0000;
}
#drugi{
position: relative;
top:10px;
left:10px;
}
```

Izgled stranice:



Važno je znati da prvobitna pozicija elementa (mesto na kojem se element nalazio pre relativnog pozicioniranja) ostaje zauzeta. To znači da prostor u kome je element bio pre pozicioniranja ostaje popunjen kao da je element još uvek tu bez obzira što je element pozicioniran na drugom mestu u dokumentu.

Ako element želimo da relativno pozicioniramo, moramo da mu zadamo *position* svojstvo sa vrednošću *relative.*

Kada navedemo samo ovo svojstvo, element se neće pomeriti, jer nismo naznačili gde želimo da ga premestimo. Da bismo to postigli moramo koristiti *top, right, left* ili *bottom* svojstvo u zavisnosti da li element pometamo horizontalno (*left, right*) ili vertikalno (*top, bottom*). Ova svojstva mogu imati i negativne brojčane vrednosti. Takođe, ova svojstva ne možemo koristiti bez prethodnog navođenja *position* svojstva. Ova svojstva se koriste kod svih vrsta pozicioniranja.

Kod relativnog pozicioniranja važi sledeće:

- „top“ vrednost - udaljenost od gornje strane statičke pozicije elementa.
- „bottom“ vrednost – udaljenost od gonje strane statičke (prvobitne) pozicije elementa
- „left“ - udaljenost od leve strane prvobitne pozicije elementa
- „right“ - udaljenost od desne strane prvobitne pozicije elementa

Vrednosti za ova četiri svojstva se mogu izraziti pomoću brojčane vrednosti za dužinu, procentualne vrednosti ili korišćenjem ključne reči *auto*.

Kada se za svojstvo definiše brojčana vrednost za dužinu, tada se položaj elementa određuje u zavisnosti od toga koja je vrednost dodeljena svojstvu *position.*

Kada je svojstvo definisano pomoću procentualnih vrednosti, tada se element pozicionira na osnovu vrednosti koja je izračunata kao zadati procenat od širine elementa roditelja.

Ako nije podešena, podrazumevana vrednost za svojstvo je *auto.*

Relativno pozicionirani elementi se obično koriste kao kontejneri (roditelji) za apsolutno pozicionirane elemente.

Apsolutno pozicioniranje

Apsolutno pozicioniranje znači pozicioniranje elementa u odnosu na roditelja. Treba zapamtiti da se apsolutno pozicioniranje ne vrši u odnosu na dokument i ne treba ga mešati sa relativnim pozicioniranjem. Znači, ako imamo određeni element koji se nalazi u okviru nekog drugog elementa, tada će naš element biti pozicioniran u odnosu na taj element (roditelj). Sa druge strane, ako naš element nema roditelja nego je direktni potomak <body> taga, tek tada će se element apsolutno pozicionirati u odnosu na dokument, jer je dokument njegov element roditelj. Prilikom apsolutnog pozicioniranja element izlazi iz prirodnog toka dokumenta i biva postavljen na mesto koje smo odredili pomoću *top, left, bottom* ili *righ*t svojstva.

Treba znati i to, da kada se vrši apsolutno pozicioniranje, element roditelj ne sme imati statičku poziciju, već mora biti pozicioniran apsolutno ili relativno. Ako nam je ipak potrebno da nam na stranici element roditelj bude u prirodnom toku, možemo ga pozicionirati relativno i podesiti svojstva *top* i *left* na 0. Ovim element roditelj zadržava svoj prirodni tok u dokumentu, ali istovremeno omogućava da drugi elementi budu pozicionirani u odnosu na njegovu poziciju.

Navešćemo primer apsolutnog pozicioniranja „prvi“ paragrafa u odnosu na „kontejner“ div roditelja koji je relativno pozicioniran.

HTML kod:

```
<html>
	<body>
		<div id="kontejner">
			<p id="prvi">Prvi paragraf</p>
		</div>
	</body>
</html>
```

CSS pravila:

```
div#kontejner {
        position:relative;
        top:0;
        left:0;
        width:200px;
        height:100px;
        background-color:#FF0000;
        }
p {
        width:100px;
        height:30px;
        background-color:#333333;
        color:#FFFFFF;
        }
#prvi{
        position:absolute;
        top:20px;
        left:-15px;
}
```

Rezultat:

Fiksno pozicioniranje

Fiksno pozicioniran element je pozicioniran relativno u odnosu na prozor pretraživača. Element se pozicionira u odnosu na gornju i levu ivicu prozora i ne pomera se kada skrolujemo stranu. Vrednost *position* svojstva je „fixed“. Dokument i drugi elementi se ponašaju kao da fiksno pozicioniran element ne postoji.

Preklapanje elemenata (z-index svojstvo)

Kod pozicioniranja elemenata na opisane načine, neretko se događa da se elementi preklapaju. Da bi definisali koji će element biti ispod a koji iznad koristi se „z-index“ svojstvo.

Kada se dva elementa preklapaju, za njih navodimo *z-index* vrednosti. Onaj element koji ima veću vrednost ovog svojstva biće ispred elementa koji ima manju vrednost. Primer:

HTML kod:

```
<html>
	<body>
		<h1>Naslov najvišeg nivoa</h1>
		<img src="slika.png" width="100" height="100" alt=”Slika koja ne moze da se prikaze”/>
	</body>
</html>
```

CSS pravila:

```
img{
	position: absolute;
	left: 0px;
	top: 0px;
	 z-index: -1;
}
```

U ovom primeru kao rezultat prikazaće se sadržaj naslova preko slike koja ima „z-index“ vrednost -1.

Dakle, kada se vrednost svojstva z-index specificira za dva elementa koji imaju istog roditelja, tada je element sa većim z indeksom ispred elementa koji ima niži z-index. To znači da je z-index elementa roditelja važan pri određivanju koji element prekriva a koji biva prekriven.

Pretpostavićemo da imamo apsolutno pozicioniranu sliku sa z-indeksom 10 u paragrafu koji ima z-index 1 i sliku sa z-indeksom 2 unutar drugog paragrafa koji ima indeks 5. Ako se dve slike preklapaju, koja je iznad? U ovom slučaju je to slika sa nižim z-indeksom, zato što njegov paragraf roditelj ima veći z-index. Dakle, paragraf sa većim z-indeksom prekriva onog sa sa manjim z-indeksom, i kao posledica slika sa nižim z-indeksom prekriva sliku sa višim indeksom. Zaključujemo da u ovom slučaju glavnu ulogu igraju paragrafi, na osnovu čijiih z-indeksa se određuje redosled, a da su z-indeksi sadržaja ta dva paragrafa nebitni.

Po default-u kada se dva elementa preklapaju, a ne navedemo z-index svojstvo, na vrhu će sa naći onaj element koji je poslednji napisan u HTML dokumentu.

Overflow svojstvo

Ovo svojstvo se koristi u slučaju kada dimenzije elementa nisu dovoljno velike da prihvate sav sadržaj koji želimo da postavimo unutar tog elementa. Svojstvo *overflow* može uzeti vrednosti koje su opisane ključnim rečima: *visible, hidden, scroll* i *auto*.

Vrednost *visible* znači da pretraživač sam treba da uveća dimenzije elementa kako bi ceo sadržaj mogao da se prikaže.

Vrednost *hidden* znači da pretraživač treba da odseče deo sadržaja koji ne može da se prikaže.

Vrednost *scroll* znači da web pretraživač treba da ubaci scrollbar na element bilo da je sadržaj elementa veći od dimenzija elementa ili nije.

Vrednost *auto* znači da pretraživač treba da doda horizontalni ili vertikalni scrollbar kada se za to javi potreba, odnosno samo onda kada sadržaj prevazilazi definisane dimenzije elementa.

Ako nije definisana, podrazumevana vrednost za ovo svojstvo je *visible.*

9.2. *Float* svojstvo

Float pozicioniranje podrazumeva pozicioniranje elemenata po horizontali. Dakle, pomoću ovog svojstva elemente možemo pozicionirati levo ili desno unutar pripadajućeg elementa, ali ne i pomerati gornju i donju poziciju samog elementa. Ovakav način pozicioniranja dosta se primenjuje kod pozicioniranja slika oko kojih se nalazi određeni tekst. Npr. ako slici zadamo levo pozicioniranje, sav tekst koji posle nje sledi smestiće se odmah uz nju sa desne strane. Pored upotrebe za pozicioniranje slike i teksta, *float* svojstvo se koristi i za definisanje izgleda (layout) same stranice.

Vrednost svojstva float može biti definisano jednom od tri ključne reči: *none, left, right*.

Vrednost *none* znači da element ne menja svoj prirodni tok u dokumentu.

Vrednosti *left* znači da je element izmešten iz svog prirodnog toka u dokumentu i da je tretiran kao blok element sa leve strane ostatka sadržaja elementa roditelja.

Vrednosti *right* znači da je element izmešten iz svog prirodnog toka u dokumentu i da je tretiran kao blok element sa desne strane ostatka sadržaja elementa roditelja.

Podrazumevana vrednost za ovo svojstvo je *none*.

Važno je napomenuti da *float* svojstvo važi za sve tipove elemenata, bilo da su oni *block, inline* ili *inline-block*. Takođe, svaki element sa zadatom *float* vrednošću (*left* ili *right*) postaje *block* element. To znači, da ako <a> tagu dodelimo neku *float* vrednost, svi linkovi koji su do tada bili *inline*, postaju *block* elementi kojima se mogu zadati širina i visina.

Clear svojstvo

Kada upotrebimo *float* svojstvo nad jednim elementom svi elementi koji se u html dokumentu nalaze posle datog elementa slagaće se jedni za drugim jer su nasledili ovo *float* svojstvo. Na primer, ako imamo glavni deo (*content*) sajta koji je organizovan u dve kolone korišćenjem *float* svojstva i *footer* sekciju koja se nalazi ispod *content* sekcije, i *footer* sekcija će promeniti svoj normalni tok jer prati prethodnu *float*-ovanu content sekciju. Da bismo ukinuli *float* svojstvo nakon content sekcije i time omogućili *footer* sekciji da se prikaže u novom redu bez *float*-ovanja, potrebno je sa *footer* sekciji dodamo *clear* svojstvo.

Clear svojstvo govori da li, i sa koje strane neki element dozvoljava da drugi element bude priljubljen uz njega.

Clear svojstvo može da ima četiri vrednosti: *both, left, right* i *none*. Podrazumevana vrednost je *none*.

Vrednost *left* označava da element ne dozvoljava priljubljivanje elemenata sa svoje leve strane.

Vrednost *right* označava da element ne dozvoljava priljubljivanje drugih elemenata sa svoje desne strane.

Both vrednost označava da element ne dozvoljava priljubljivanje ni sa svoje leve ni sa svoje desne strane.

10. Pseudo klase i pseudo elementi

Pseudo klase

Pseudo klase koristimo kada elementu želimo da dodamo druga stanja u kojima će on drugačije izgledati. Ta stanja najčešće možemo videti kod linkova : prelazak mišem preko linka (hover), izgled linka nakon što kliknemo na njega (visited), izgled aktivnog linka (active), izgled neposećenog linka (link).

Pseudo klase su predefinisane (nazivi su unapred definisani). Sintaksa pseudo klase je sledeća:

```
selektor:pseudoklasa{
    svojstvo:vrednost;
}
```

Primer:

```
a:link{
color:#333333;
}
a:visited{
color:#0000FF;
}
```

```
a:hover{
color:#FF0000;
}
a:active{
color: #00FF00;
}
```

Kada navodimo pseudo klase <a> taga moramo voditi računa o redosledu navođenja: *hover* klasu ne možemo navesti pre *link* ili *visited* klase ako postoje, takođe *active* klasu ne možemo navesti pre *hover* klase.

Pseudo elementi

Za razliku od pseudo klasa, pseudo elementi se koriste za dodavanja određenih elemenata preko css-a. Dva najčešće korišćena pseudo elementa si *before* i *after.*

::before – pseudo element se koristi za dodavanje određenog sadržaja pre elementa koji je naveden kao selektor.

::after – koristi se za dodavanje sadržaja nakon elementa koji je naveden kao selektor

Primer:

```
p::before{
content: ‘ “ ’;
color:red;
}
p::after{
content:’ ” ’;
color:red;
}
```

U primeru ispred svih paragrafa dodaje se otvarajući dupli navodnik, dok se nakon paragrafa dodaje zatvarajući dupli navodnik.

11. CSS3 transformacije

2D transformacije

Transformacije nam omogućavaju da menjamo oblik, veličinu i poziciju elementa. Možemo da pomeramo (translacija), rotiramo, skaliramo (promena veličine) i iskosimo element.

translate() – ovaj metod nam omogućava da menjamo trenutnu poziciju elementa po x ili y osi. Primer:

```
img{
transform: translate(100px, 30px);
}
```

Prva vrednost od 100px označava pomeranje svih img elemenata 100px po x osi (horizontalno) i 30px po y osi (vertikalno).

rotate() – rotiranje elementa za zadati ugao.

Primer:

```
 img{
transform: rotate(30deg);
}
```

Slike se rotiraju za 30 stepeni u smeru kazaljke na satu. Ukoliko navedemo negativnu vrednost ugla rotacija se vrši u suprotnom smeru od kretanja kazaljke na satu.

scale() – povećavanje ili smanjivanje veličine elementa.

Primer:

```
img{
transform: scale(2,3);
}
```

Skaliramo sliku tako da bude dva puta šira od trenutne širine i tri puta viša od trenutne visine.

skew() – metod koristimo kada želimo da iskosimo element po x ili y osi.

Primer:

```
img{
transform: skew(30deg, 20deg);
}
```

Slike će biti iskošene za 30 stepeni po x i 20 stepeni po y osi.

3D transformacije

rotateX() – rotacija elementa oko x ose za zadati ugao.

Primer:

```
img{
transform: rotateX(30deg);
}
```

rotateY() – rotacija elementa oko y ose za zadati ugao.

```
img{
transform: rotateY(100deg);
}
```

rotateZ() – rotacija elementa po z osi za zadati ugao.

```
img{
transform: rotateZ(120deg);
}
```