CSS

CSS

- Je jezik koji nam služi da opisemo stil Web stranice (koju smo prethodno obeležili i struktuirali koristeći HTML)

- Stil može biti recimo boja slova, boja pozadine, font slova, veličina slova, pozicija na stranici i dr.

- Svakom HTML elementu (tagu) možemo dodeliti neki stil
- Stilovi se zadaju određenim pravilima

CSS pravila

- Forma pravila:

```
 selektor {
osobina : vrednost ;
osobina : vrednost ;
...
}
```

- Selektor – HTML tag(ovi), tj. element(i) na koje želimo da primenimo stil
- osobina – stil koji želimo da izmenimo
- vrednost – vrednost stil

CSS pravila - primer

```
p {
    color: red;
    color: red;
}
```

- Ovo pravilo se odnosi na sve **p** tagove I boji tekst unutar paragrafa u crveno (`color: red`)

Uključivanje CSS stila na HTML stranicu

- Postoje 3 načina za uključivanje stila na stranicu:

1. `style` atribut

    - svaki tag može imati `style` atribut čija vrednost može biti kod CSS stila koji će se primeniti na taj tag

```
<p style=“color:yellow; background-color: blue;”>
    Ovo je tekst paragrafa
</p>
```

Uključivanje CSS stila na HTML stranicu

2. `style` tag

- unutar head taga možemo imati style tag koji će sadržati kod CSS stila koji se primenjuje na sve tagove na stranici

```
<head>
  <style>
    p {  color:yellow;
         background-color: blue; }
  </style>
</head>
```

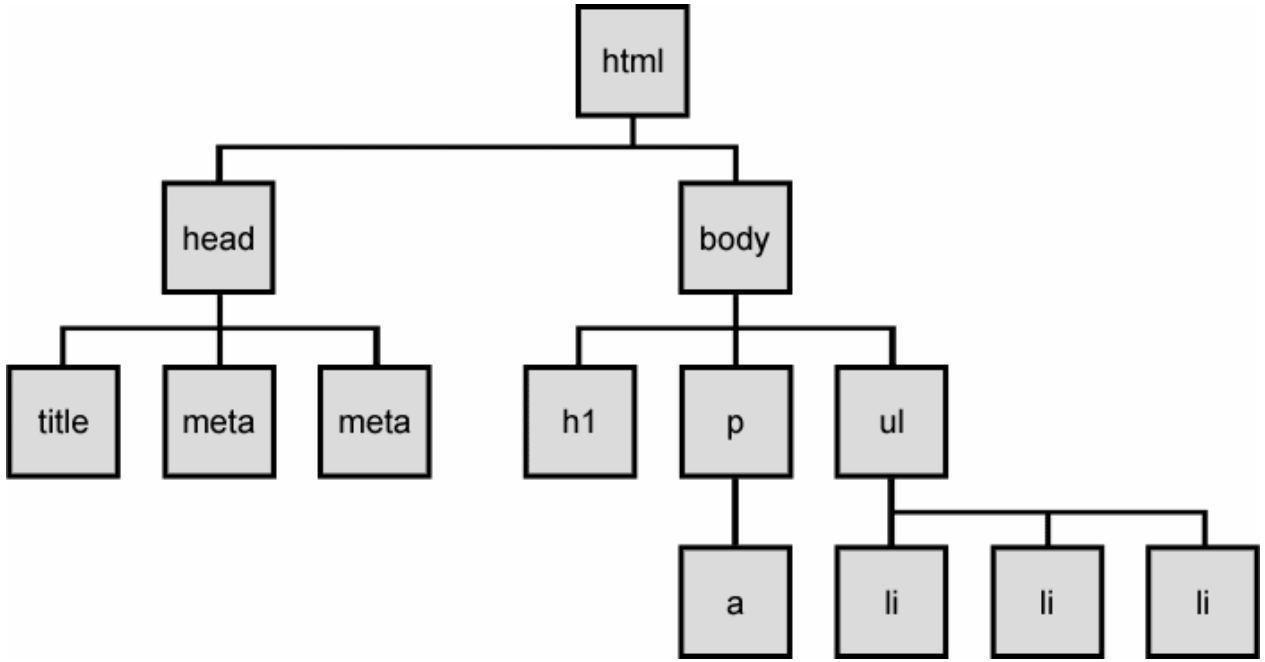
Uključivanje CSS stila na HTML stranicu

3. poseban fajl sa CSS kodom
   - najbolja praksa je da CSS kod pišemo u posebnom fajlu jer onda više HTML stranica mogu da dele isti stil i razdvojili smo HTML kod koji uređuje strukturu teksta od CSS stila
   - fajl u kome pisemo CSS kod mora imati .css ekstentziju
   - da bismo u HTML stranici naveli u kom fajlu se nalazi CSS stil koristimo tag link unutar taga head
   - tag link ima atribut `rel` čija vrednost treba biti `stylesheet` i atribut `href` čija vrednost je ime fajla u kome se nalazi CSS stil

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

HTML struktura

```
<html lang="en">
   <head>
        <meta charset="UTF-8">
        <meta name="autor" content="Comtrade">
        <title> CSS stil </title>
   </head>
   <body>
      <h1>CSS stil</h1>
      <p>
           Više o CSS trikovima možete naći na:
           <a href='https://css-tricks.com/'> ovom linku  </a>
      </p>
      <ul>
           <li> plava </li>
           <li> zelena </li>
           <li> crvena </li>
      </ul>
   </body>
</html>
```



CSS selektori

- Koristimo ih da navedemo na koje HTML tagove želimo da se primene CSS pravila
- Npr.

```
p {
  color:yellow;
}
```

  - u ovom slučaju `p` je selektor i on označava sve pasuse u HTML dokumentu

- Postoje različiti tipovi selektora, navešćemo neke koji se najčešće koriste u praksi

CSS selektori

- Univerzalni selektor

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

- * označava sve tagove u dokumentu
- u ovom primeru, pravilo da je font teksta Arial se primenjuje na sve HTML tagove

CSS selektori

- Selektor HTML tagova

```
h2, h3 {
  font-style: bold;
}
```

- ukoliko navedemo naziv HTML tagova (ili više naziva odvojeni zarezom) označili smo sve tagove sa tim nazivima u dokumentu
- u ovom primeru, pravilo da je tekst podebljan će se primeniti na sve naslove *h2 i h3* u dokumentu

CSS selektori

- Direktan potomak selektor

```
p>a {
  font-style: italic;
}
```

- ukoliko navedemo *selektor1* > *selektor2*, time smo označili sve *selektor2* koji su direktni potomci *selektor1*
- u ovom primeru, pravilo da je tekst iskošen će se primeniti na sve tagove *a* koji su direktni potomci taga *p*

CSS selektori

- Potomak selektor

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

- ukoliko navedemo *selektor1 selektor2*, time smo označili sve *selektor2* koji su potomci *selektor1*
- u ovom primeru, pravilo da je su sva slova u tekstu velika će se primeniti na sve tagove *a* koji su potomci taga *p*

CSS selektori

- ID selektor
- Svaki HTML tag može imati atribut `id`
- Vrednost atributa `id` treba da bude jedinstvena, odnosno ne smeju postojati dva taga koji imaju istu vrednost atributa `id`

```
<p id="boja"> Moja omiljena boja je plava. </p>
```

- Možemo označiti HTML tag na osnovu vrednosti atributa tako što navedemo karakter **#** i *vrednost* atributa `id`

```
#boja {
   color: blue;
 }
```

- u ovom primeru, pravilo da je su slova obojena plavom bojom će se primeniti na tag koji ima atribut `id` koji je jednak *boja*

CSS selektori

- Class selektor
- Svaki HTML tag može imati atribut `class`
- Vrednost atributa `class` ne mora biti jedinstvena, vise tagova mogu imati istu vrednost atributa `class`
- Atribut `class` može imati više vrednosti odvojene razmakom

```
<p class="podebljano iskoseno"> Ovaj tekst je podebljan i iskošen. </p>
```

- Možemo označiti HTML tagove na osnovu vrednosti atributa `class` tako što navedemo karakter `.` i *vrednost* atributa `class`

```
.podebljano  {
   font-weight: bold;
 }
```

- u ovom primeru, pravilo da je tekst podebljan će se primeniti na sve tagove koji imaju atribut `class` koji je jednak *podebljano*

Slaganje CSS stilova

- Obratimo pažnju na to da isti tag možemo označiti koristeći različite selektore
- Šta se dešava ukoliko pokušamo da primenimo dva različita stila na isti tag
- Npr.

```
<p id='jezik'> Srpski jezik </p>
```

```
#jezik {
  color: black;
}
```

```
p {
  color: orange;
}
```

Koje boje će biti tekst pasusa?

- Postoje pravila slaganja CSS stilova, odnosno prednosti nekih selektora u odnosu na druge
- Na ovoj stranici možete proveriti koji selektor ima veću prednost https://specificity.keegan.st/

Slaganje CSS stilova

- Ukoliko su dva selektora identična, primeniće se stil iz poslednjeg navedenog
- Ukoliko je jedan selektor specifičniji od drugog, primeniće se njegov stil

```
#jezik {                    p {
  color: black;                   color: orange !important;
}                           }
```

selektor `#jezik` je specifičniji od `p` jer obeležava tačno jedan tag koji ima vrednost atributa `id` tako da će tekst biti obojen u crno

- Ukoliko nakon CSS pravila navedemo oznaku `!important` to pravilo će imati prednost u odnosu na ostale, npr.

```
#jezik {                    p {
  color: black;                   color: orange !important;
}                           }
```

tekst će se obojiti u narandžasto

Nasleđivanje CSS stilova

```
<p id=‘jezici’>  Na sledededećim stranicama možete pronaći više informacija:
    <a href=‘eng.html’> Engleski jezik </a>
    <a href=‘fra.html’> Francuski jezik </a>
    <a href=‘nem.html’> Nemački jezik </a>
</p>

#jezici {
  font-family: Arial;
}
```

- Ukoliko ne navedemo stil za neki tag, on će naslediti stilove svojih roditelja
- U ovom primeru, tekst tagova a će biti ispisan fontom Arial
- Ukoliko želimo da naglasimo da neki tag treba da nasledi određeno pravilo od roditelja, vrednost tog pravila treba da postavimo na inherit

```
a {
  font-family: inherit;
}
```

CSS boje

- Postoje nekoliko načina zadavanja boja, mi ćemo ovde opisati 5:
  - ime boje
  - RGB
  - HEX
  - RGBA
  - HSLA

CSS boje – ime boje

- Boje mogu da se zadaju imenima na engleskom
- Lista boja je prilično velika i korisna je jer čim vidimo ime boje znamo o kojoj boji se radi
- Problem sa imenovanim bojama je što ne možemo promeniti nijansu boje, već za to moramo koristit neki drugi način zadavanja boja

```
.zelena {
 background:green;
}
.plava {
 background:blue;
}
.crvena {
background:red;
}
.zuta {
background:yellow;
}
```

CSS boje – RGB I RGBA

- Boju navodimo kao kombinaciju crvene, zelene i plave (Red Green Blue)
- Navodimo jačine ovih boja koje mogu biti u opsegu od 0-255 u sledećem formatu:

  rgb(*jacina_crvene*, *jacina_zelene*, *jacina_plave*)

- Npr. rgb(255, 165, 0) je narandžasta
- rgb(255, 255, 255) je bela a rgb(0, 0, 0) je crna
- Dodatno, kao četvrtu stavku možemo navesti jačinu providnosti boje
- Onda se boja zadaje ovako

  rgba(*jacina_crvene*, *jacina_zelene*, *jacina_plave,* *jacina_providnosti*)

- Jačina providnosti može biti u opsegu 0-1, 0 je potpuno providna, 1 nije providna

CSS boje – HEX, HSL i HSLA

- Navođenje boje HEX notacijom je slično RGB
- Sintaksa je ovakva: #000000
- Odnosno prva dva broja označavaju nijansu crvene, druga dva nijansu zelene i treća dva nijansu plave
- Ovi brojevi se pišu u heksadecimalnom sistemu tj. brojevnom sistemu sa osnovom 16 (mi svakodnevno koristimo dekadni odnosno sistem sa osnovom 10)

#ff0000 crvena boja

- Navođenje boje HSL i HSLA (Hue Saturation Lightness) notacijom je sličnog formata kao RGB
- Boju navodimo kao kombinaciju nijanse, zasićenja (od 0-100%) i svetlosti (od 0-100%) i opciono providnosti (0-1)

hsl(*nijansa, zasićenje, svetlost*)

hsl(120, 100%, 25%) zelena boja

Tekst

- Postoje razni načini stilozovanja teksta, neke od njih ćemo navesti ovde:
  - `color` – boja slova
  - `letter-spacing` – razmak između slova
  - `text-align` – horizontalno poravnanje teksta, može biti: `center, left, right, justified`
  - `word-spacing` – razmak između reči
  - `text-decoration` – dekoracija slova, može biti: `overline, line-through, underline`
  - `line-height` – visina prostora za slova
  - `font-family` – font teksta
  - `font-size` – veličina slova
  - `font-style` – stil fonta, može biti: `italic, oblique, normal`
  - `font-weight` – debljina slova, može biti: `normal, bold, lighter, …`

Merne jedinice

- Koristimo ih kada želimo da zadamo neku veličinu (npr. veličinu slova `font-size`, razmak između slova `letter-spacing` i dr.)

  - `px` - piksel (označava tačku na monitoru)
  - `%` - jedan procenat maksimalne širine elementa
  - `vw` – procenat (%) širine HTML dokumenta (ne mora biti širina ekrana)
  - `vh` - procenat (%) visine html dokumenta
  - `vmin` - procenat (%) manje dimenzija html dokummenta
  - `vmax` - procenat (%) veće dimenizija html dokumenta
  - `em` - širina slova "m" u trenutnom fontu

- Postoje I druge jedinice koje verovatno nećete koristiti ali treba ih prepoznati (`pt, cm, mm, in, ex, ...`)

```
.velika_slova {
    font-size:  20px;
}
```

Font

- Browseri podržavaju određenu listu fontova
- Pomoću `font-family` možemo navesti font koji želimo ili više njih razdvojene zarezom (ukoliko browser ne podržava prvi, primeniće se sledeći i tako redom)

```
font-family: "Times New Roman", Georgia, Serif;
```

- Međutim možemo koristiti i neke druge fontove, koje browser inicijalno ne podržava
- Za zadavanje novog font koristimo sledeće CSS pravilo

```
@font-face {
    font-family: ime_novog_fonta;
    src: url(putanja_do_fonta);
}
```

- Na ovaj način smo uključili font koji se nalazi u datoteci *putanja_do_fonta* i nazvali smo ga *ime_novog_fonta*
- Ukoliko želimo da zadamo nekom tagu naš nov font to radimo na sledeći način:

```
a {
    font-family:  ime_novog_fonta;
}
```

Font

- Browseri podržavaju određenu listu fontova
- Pomoću `font-family` možemo navesti font koji želimo ili više njih razdvojene zarezom (ukoliko browser ne podržava prvi, primeniće se sledeći i tako redom)

```
font-family: "Times New Roman", Georgia, Serif;
```

- Međutim možemo koristiti i neke druge fontove, koje browser inicijalno ne podržava
- Za zadavanje novog font koristimo sledeće CSS pravilo

```
@font-face {
    font-family: ime_novog_fonta;
    src: url(putanja_do_fonta);
}
```

- Na ovaj način smo uključili font koji se nalazi u datoteci *putanja_do_fonta* i nazvali smo ga *ime_novog_fonta*
- Ukoliko želimo da zadamo nekom tagu naš nov font to radimo na sledeći način:

```
a {
    font-family:  ime_novog_fonta;
}
```

CSS selektori – prvo slovo, linija

- Ponekad želimo da primenimo neki stil samo na prvo slovo ili prvu liniju teksta
- Kako bismo to postigli, možemo da koristimo posebne oznake pored CSS selektora

```
selektor:first-letter
selektor:first-line
```

```
p.uvod:first-line {
    font-size: 20px;
}
```

```
ul>li:first-letter {
color: blue;
}
```

CSS selektori – linkovi, akcije korisnika

- Možemo obeležiti linkove koje su korisnici nekad posetili ili ne i menjati im stil
- Oznake pored CSS selektora (koji obeležava link)

`a:visited` - ukoliko je link bio posećen
`a:link` - ukoliko link nije bio posećen

- Možemo obeležiti bilo koje tagove na osnovu korisničkih akcija i menjati im stil

`selektor:hover` - korisnik je prešao kursorom preko sadržaja taga
`selektor:active` - korisnik je aktivirao tag (npr. kliknuo na dugme)
`selektor:focus` - tag je u fokusu (npr. kursor se nalazi u tekstualnom polju, pripremljen za kucanje)

CSS selektori – atributi

- Možemo obeležiti tagove na osnovu njihovih atributa

selector[**ime**] - ukoliko tag ima atribut sa zadatim imenom

selector[**ime=vrednost**] - ukoliko tag ima atribut sa zadatim imenom I njegova vrednost je jednaka zadatoj vrednosti

```
li[class] {
    font-face: Serif;
}
```

```
input[type='text'] {
    font-style: italic;
}
```

CSS selektori – redosled

- Možemo obeležiti tagove na osnovu njihovog redosleda

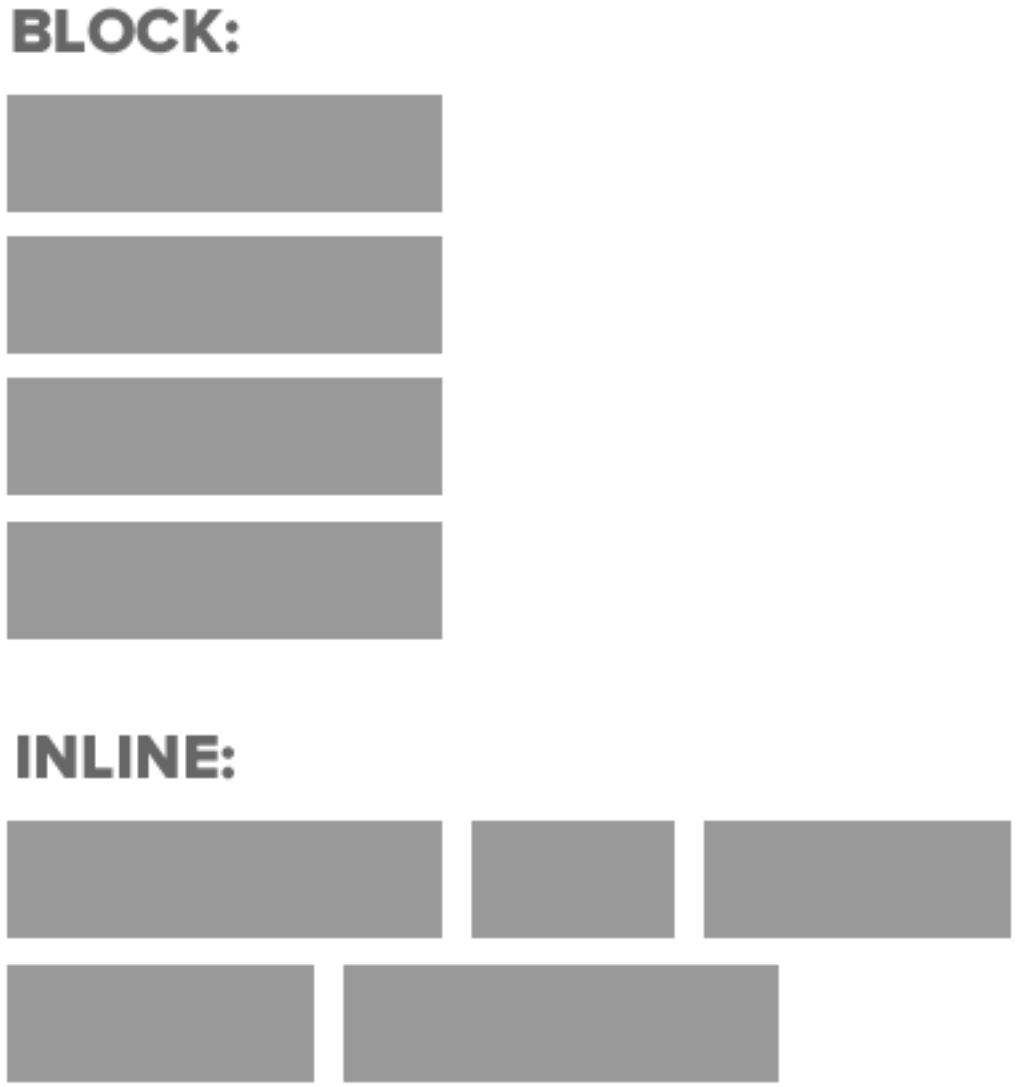
`selector:first-child` - prvi potomak
`selector:nth-child(n)` - potomak koji je *n* po redu, gde *n* moze biti
broj (npr. 3 - treci potomak po redu)
odd (neparni po redu)
even (parni po redu)

```
li:nth-child(3) {
    font-face: Serif;
}
```

```
input:first-child {
    font-style: italic;
                {
```

Block I inline HTML tagovi

- Ako zamislimo svaki HTML element (tag i sadržaj unutar njega) kao kutiju koja ima neki sadržaj, postoje dva tipa HTML elemenata:
  - **Block** – redjaju se jedan ispod drugog i zauzimaju što više širine mogu (širinu roditeljskog elementa, ili ekrana)
  - **Inline**– redjaju se jedan pored drugog i zauzmaju onoliko širine koliko im je potrebno
- Unutar **block** elementa možemo stavljati **block** i **inline** elemente, ali unutar **inline** elementa možemo stavljati samo **inline** elemente



Block I inline HTML tagovi

- Neki block tagovi su:
  - `address, article, aside, blockquote, canvas, dd, div, dl, dt, fieldset, footer, form, h1-h6, header, hr, li, main, nav, ol, p, pre, section, table, tfoot, ul, video`

- Neki inline tagovi su:
  - `a, br, button, cite, code, dfn, em, img, input, label, map, object, q, select, span, strong, textarea`

- Možemo promeniti način prikaza pravilom:
  - `display: inline;`
  - `display: block;`

Dimenzije

- Za menjanje dimenzija tagova koristimo sledeća CSS pravila:
  - `width` - širina
  - `max-width` – maksimalna širina
  - `min-width` – minimalna širina
  - `height` – visina
  - `max-height` – maksimalna visina
  - `min-height` – minimalna visina

```
div {
width:150px;
height:150px;
min-width:100px;
min-height:100px;
max-width:200px;
max-height:200px;
  }
```

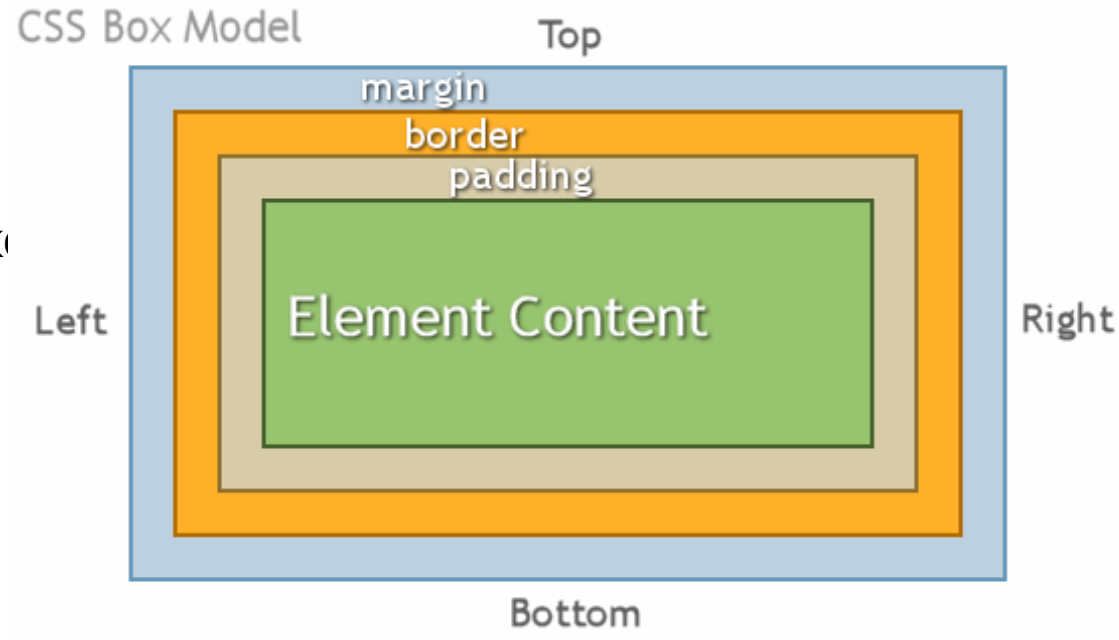
Dimenzije

- Blok elementi će pratiti zadatu širinu i visinu čak i u slučaju kada sadržaj koji smo napisali ne može da stane u zadate dimenzije
- Zbog toga postoji pravilo `overflow` koje određuje šta se dešava ukoliko sadržaj ne može da stane u tag sa navedenim dimenzijama
- Postoje I pravila `overflow-x, overflow-y` koja se primenjuju za x odnosno y osu
- `overflow, overflow-x, overflow-y` mogu imati vrednosti
  - `visible` – sadržaj će se prikazati ceo i izaći iz dimenzija elementa
  - `hidden` – sadržaj se neće prikazati ceo, već samo onaj deo koji staje u element
  - `scroll` – napraviće se scroll barovi
  - `auto` – desiće se ono što je podrazumevano ponašanje browsera

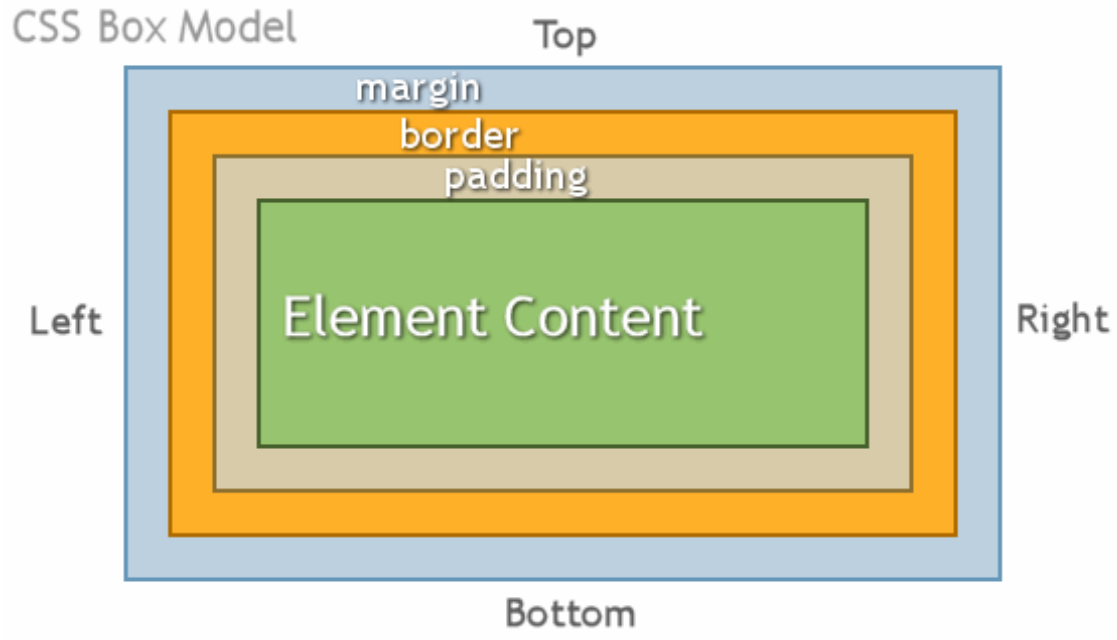
BOX model

- Svaki element možemo zamisliti kutiju unutar k

  se nalazi sadržaj
- Ta kutija se sastoji od 4 celine:
  - sadržaj (eng. content)
  - udaljenost od okvira (eng. padding)
  - okvir (eng. border)
  - udaljenost od drugih kutija (eng. margin)
- Možemo zadati veličine svakoj celini
- Veličinu sadržaja (content) zadajemo pravilima `width, max-width, min-width, height, max-height, min-height`



BOX model - padding



- Padding je razmak između okvira (border)

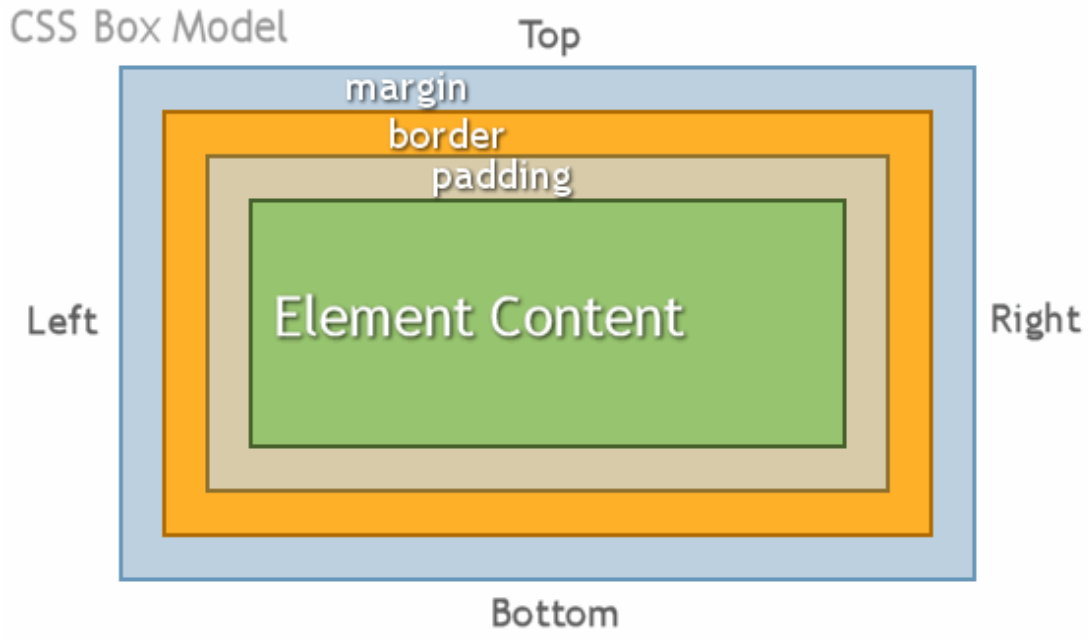
  i teksta tj. sadržaja
- Zadajemo ga pravilima:
  - `padding-top`
  - `padding-right`
  - `padding-bottom`
  - `padding-left`
- Možemo ga zadati i korišćenjem jednog pravila `padding` navođenjem:
  - 4 veličine u redosledu *top, right, bottom, left*

    `padding: 10px 5px 15px 10px;` top = 10px, right = 5px, bottom = 15px, left = 10px
  - 2 veličine u redosledu *top_bottom, right_left*

    `padding: 10px 20px;` top,bottom = 10px, right,left = 20px
  - 1 veličinom koja se onda primenjuje na sve 4 strane

    `padding: 15px;` top,right,bottom,left = 15px

BOX model - border

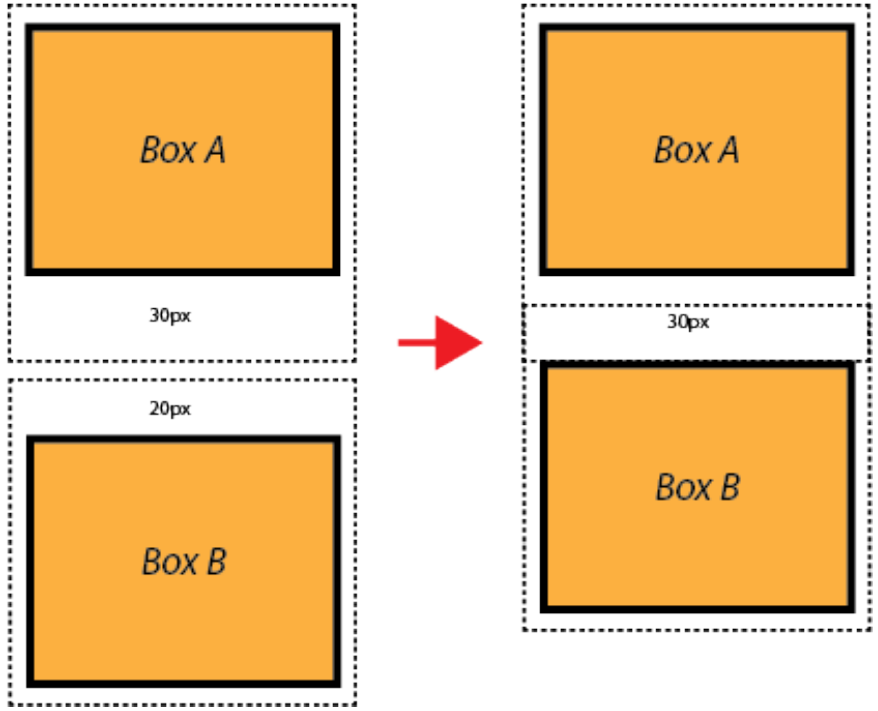


- Border je okvir oko content i padding
- Možemo zadavati veličinu bordera, stil i boju
- Veličina i boja se zadaju kao sto smo videli na drugim primerima
- Stil može biti `solid, dotted, dashed, double` itd.
- Pravila:
  - `border-top-width` `border-top-style` `border-top-color`
  - `border-right-width` `border-right-style` `border-right-color`
  - `border-bottom-width` `border-bottom-style` `border-bottom-color`
  - `border-left-width` `border-left-style` `border-left-color`
- Takođe možemo koristiti i skraćena pravila
  - `border-width` (top, right, bottom, left)
  - `border-style` (top, right, bottom, left)
  - `border-color` (top, right, bottom, left)
  - `border` (width, style, color)

```
boder: 1px solid gray;
```

BOX model - margin

- Margin je rastojanje od bordera dva susedna elementa
- Margine dva susedna elementa se sažimaju (kao na slici)
- Veličinu margine zadajemo pravilima:
  - `margin-top`
  - `margin-right`
  - `margin-bottom`
  - `margin-left`
- Takođe možemo koristiti i skraćeno pravilo
  - `margin`

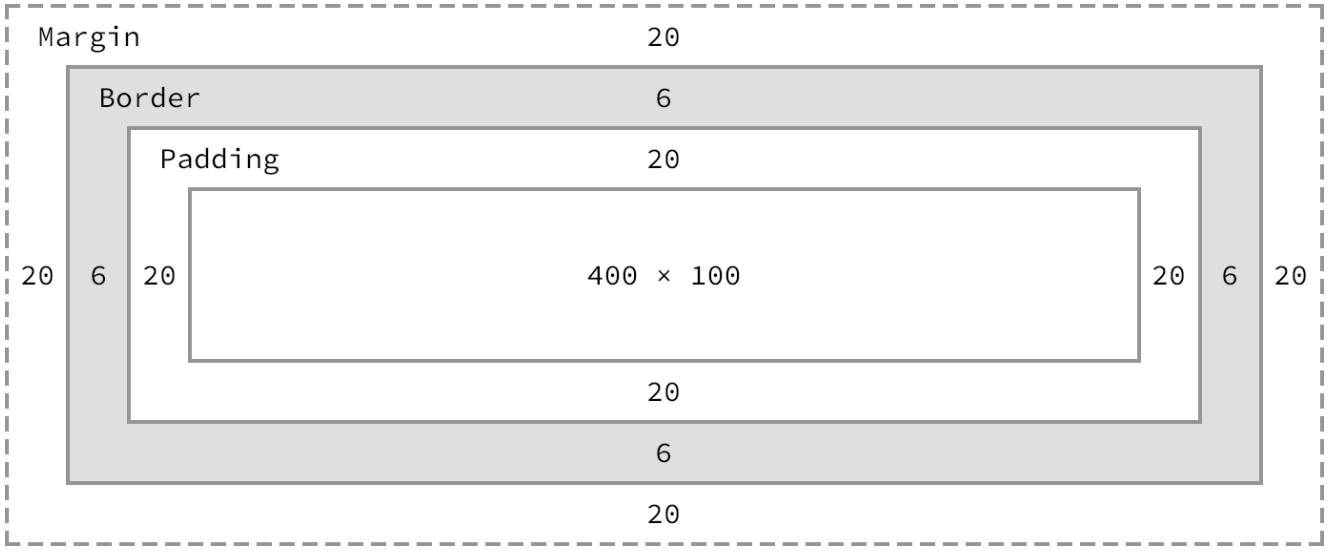


- Ukoliko želimo da centriramo element, treba da postavimo
  - `margin: auto;`

- Primer:
  - `margin: 5px 15px;` top,bottom = 5px, right,left = 15px

BOX model – dimenzije

```
div {
    width: 400px;
    padding: 20px;
    border: 6px solid gray;
    margin: 20px;
}
```



- Koliko će ovaj div biti širok, odnosno koliko prostora će zauzeti na ekranu?

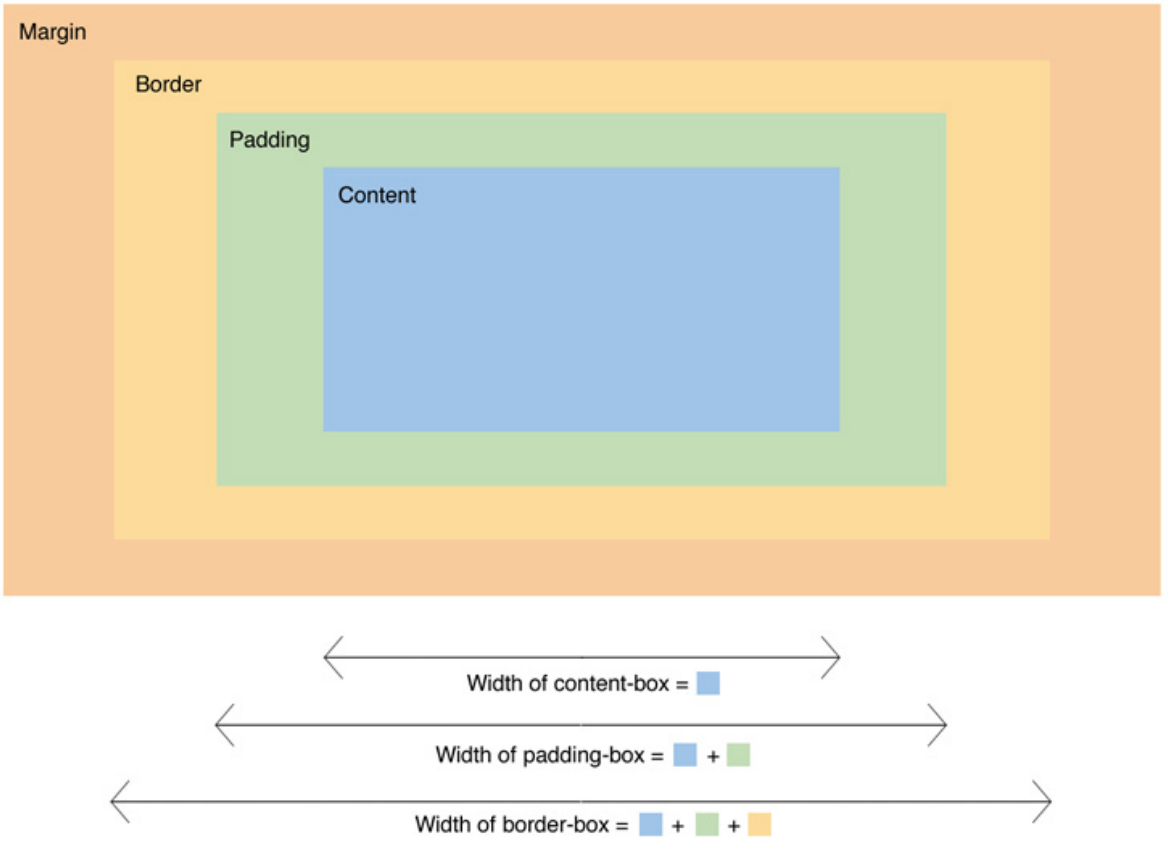
```
margin-left + border-left + padding-left + width + padding-right + border-right + margin-right

  20px     +     6px    +     20px    + 400px +      20px    +      6px    +
  20px
```

= 492px

BOX model – box-sizing

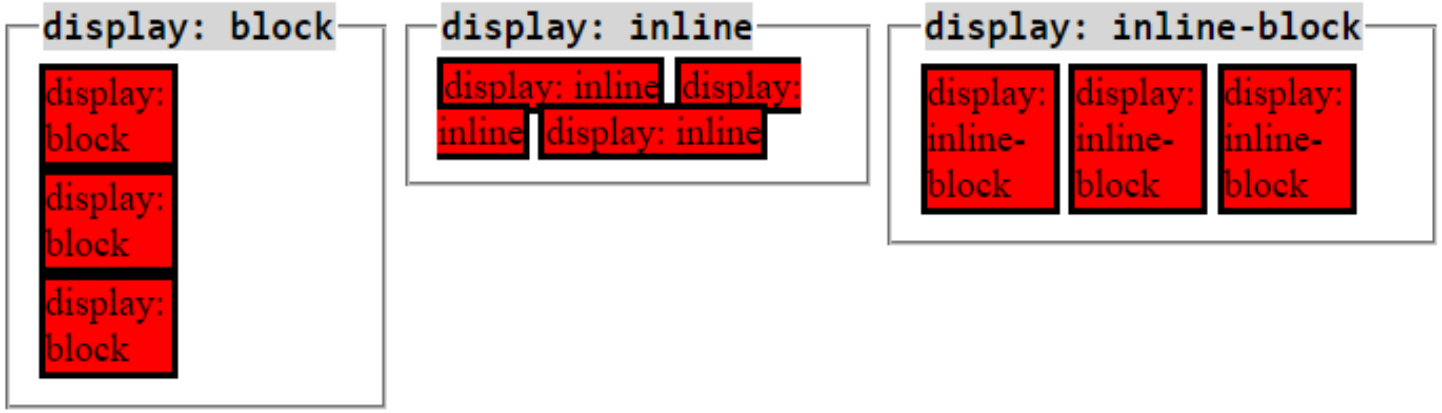
- `box-sizing` pravilo može imati vrec
  - `content-box` – `width` i `height` odnose na veličinu sadržaja
    content
  - `padding-box` – `width` i `height` odnose na veličinu sadržaja
    content + padding

  Napomena: podržan samo u Firefox
  - `border-box` – `width` i `height` se odnose na veličinu sadržaja
    content + padding+ border



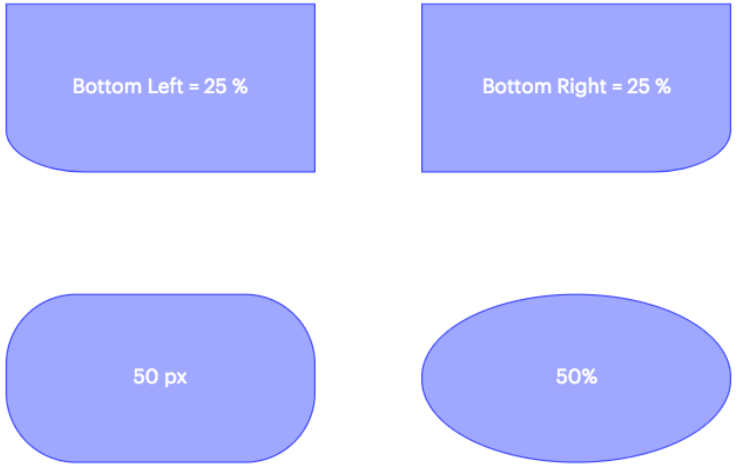
Display

- Na inline elemente ne utiče menjanje dimenzija (`width`, `height` i druga pravila)
- Na inline elemente takođe ne utiče menjanje margina (`margin-top`, `margin-left` i druga pravila)
- Ukoliko želimo da inline elementi poštuju zadate vrednosti dimenzije i margine, možemo im postaviti:
  - `display: inline-block;`
- Najprostije rečeno, ovo pravilo govori elementu da ispoštuje dimenzije i marginu, ali i dalje će ređati elemente jedne do drugih



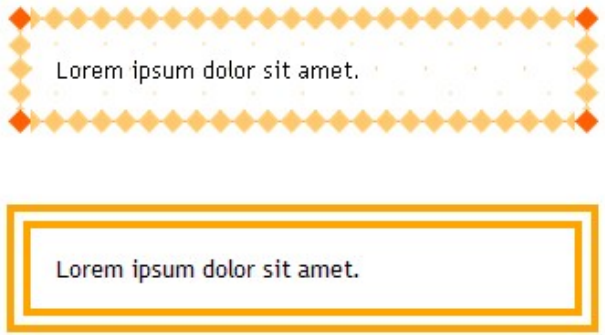
BOX model – border-image, border-radius

- Okvir možemo iskriviti pri ivicama
- Koristimo pavilo:
  - `border-radius: velicina;`
- Ili specizirati veličinu za svaku stranu posebno:
  - `border-radius: 10px 5px 10px 15px;`



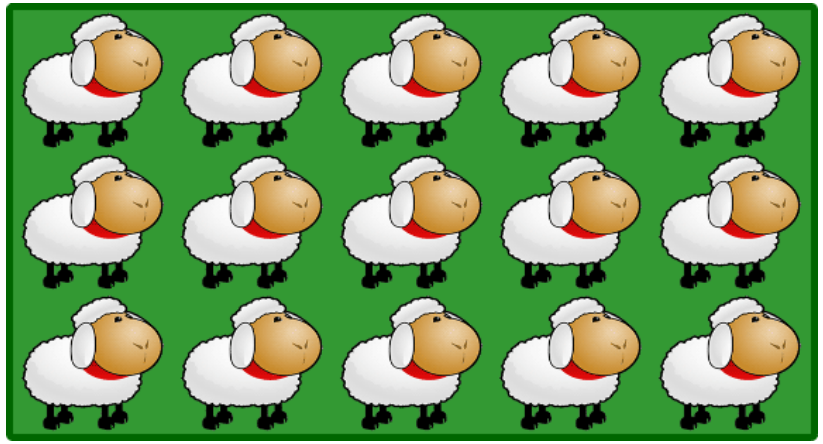
- Možemo postaviti sliku kao okvir
- Koristimo pavilo:
  - `border-image: url(slika.png);`
- Također možemo zadavati visinu slike, širinu, način ponavljanja, itd.

(više o tome se može naći u literaturi)



Pozadina - Background

- Možemo menjati pozadinu elementa pomoću pravila
  - `background-color` – boja
  - `background-image` – slika
  - `background-repeat` – način ponavljanja, može biti `repeat, repeat-x, repeat-y, no-repeat`
  - `background-position` – pozicija, može biti zadata na više načina (pogledati literaturu)



- Možemo pisati i skraćeno:

```
background: url(“ovcica.gif”) repeat center;
```

Liste

- Možemo stilizovati nabrajanja, i `ol` i `ul` tagove
- `list-style-type`
  - Za `ul` liste vrednosti mogu biti `circle, disc, square`
  - Za `ol` liste vrednosti mogu biti `decimal, lower-alpha, upper-alpha` i dr.
- `list-style-position`
  - vrednosti mogu biti `inside` i `outside`
- `list-style-image`
  - možemo postaviti sliku kao simbol nabrajanja kod `ul` taga
- Skraćeno pišemo:
  - `list-style: circle inside;`

Tabele

- Ukoliko tabela ima prazne ćelije njih možemo sakriti koristeći pravilo `empty-cells` koje može imati vrednosti `hide` i `show`

- Možemo podešavati razmak između ćelija tabele:
  - `border-spacing: 10px 5px;`

- Ukoliko želimo da se borderi dve susedne ćelije spoje, to možemo postići pravilom:
  - `border-collapse: collapse;`

- Ukoliko tabela ima naslov, njegovu poziciju mozemo promeniti pravilom:
  - `caption-side: top;`

Pozicioniranje

- Određuje gde se element nalazi na ekranu, odnosno njegovu poziciju
- Element možemo pomerati koristeći pravila:
  - `top`
  - `bottom`
  - `right`
  - `left`
- Vrednost može biti veličina zapisana u bilo kom formatu

```
left: 20px;
```

- Postoji više tipova pozicioniranja, menjamo ih pravilom `position`
- Tip pozicioniranja određuje u odnosu na šta se element pomera (ekran, roditeljski element, i dr.)

Pozicioniranje - static

- Ukoliko ne navedemo pravilo `position` ovo je odrazumevani tip pozicioniranja
- Možemo i navesti:
  - `position:static`

- Na element koji je statički pozicioniran pomeraji `top, bottom, right, left` ne utiču na poziciju elementa

Pozicioniranje - fixed

- Element se pozicionira fiksno u odnosu na ekran (površinu koju korisnik vidi)
  - `position:fixed`

- Ovako pozicionirani elementi ne utiču na pozicije susednih elemenata
- Ne pomeraju se ukoliko korisnik scroll-uje stranicu gore, dole, levo ili desno već uvek ostaju na istom mestu na ekranu
- Ovaj tip pozicioniranja može biti korisan kada želimo da se neki element uvek nalazi na ekranu, čak i kada korisnik pomera stranicu (npr. reklama, forma za login i sl.)

Pozicioniranje - relative

- Element se pozicionira relativno u odnosu poziciju koju bi mu browser dodelio odnosno u odnosu na mesto gde bi inače stajao
  - `position:relative`

- Ovako pozicionirani elementi ne utiču na pozicije susednih elemenata, oni ostaju na mestu gde bi inače stajali

- Primetimo da može da se dogodi da pomeranjem ovako pozicioniranog elementa možemo preklopiti njemu susedne elemente

Pozicioniranje - absolute

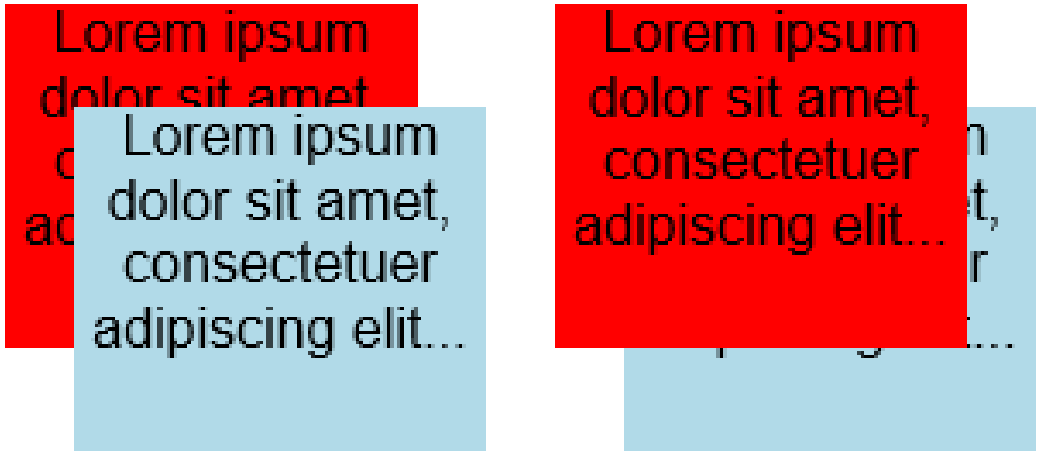
- Element se pozicionira u odnosu na prvog nestatičkog roditeljskog elementa odnosno roditeljskog elementa koji ima bilo koje pozicioniranje koje nije `static`
  - `position:absolute`
- Ukoliko ne postoji roditeljski element koji je nestatički pozicioniran onda se element pomera u odnosu na `body`
- Ovako pozicioniran element ne utiče na pozicije susednih elemenata, odnosno susedni elementi se pozicioniranju kao da ovaj element uopste ne postoji

Pozicioniranje – preklapanje elemenata

- Korišćenje pozicioniranja može dovesti do preklapanja elemenata
- Za kontrolisanje preklapanja elemenata koristimo pravilo:
  - `z-index`
- Vrednosti `z-index` mogu biti pozitivne (+) i negativne (-)
- Pozitivne označavaju da element treba da se nalazi ispred, a negativne da element treba da se nalazi iza

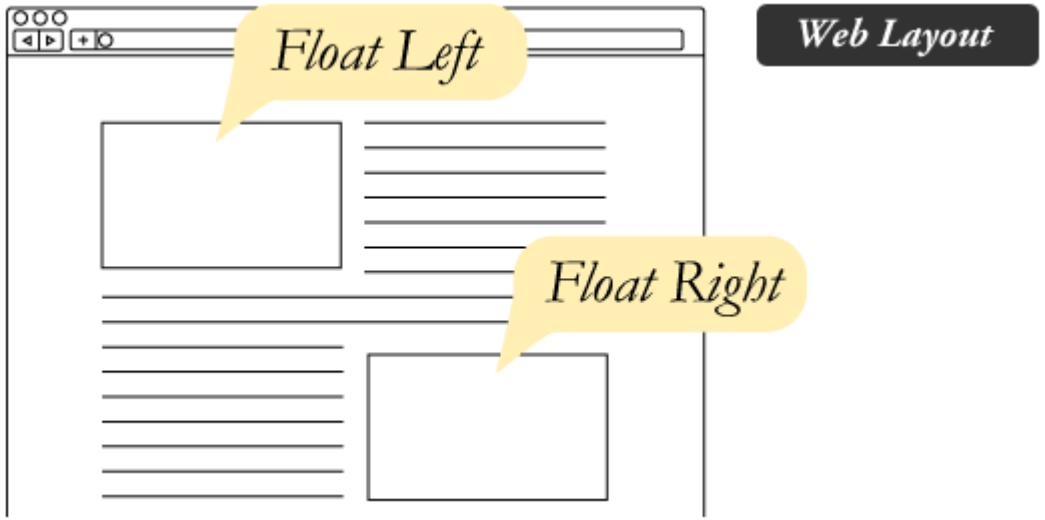
```
#crveni {
z-index: -10px;
}
#plavi {
z-index: 10px;
}
```

```
#crveni {
z-index: 10px;
}
#plavi {
z-index: -10px;
}
```



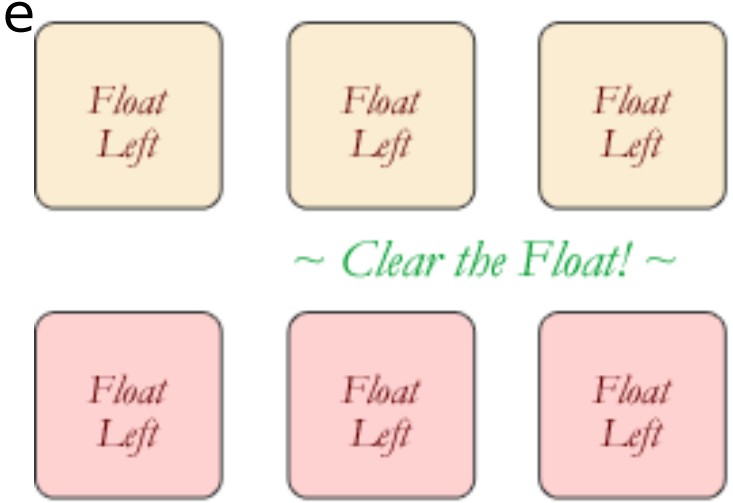
Pozicioniranje – float

- Ovo je poseban tip pozicioniranja koje se zadaje pravilom
  `float`
- Ovaj način pozicioniranja ‘zalepi’ element za levu ili desnu ivicu roditeljskog elementa
- Sve ostalo sto se nalazi u roditeljskom elementu će se plutati oko ovog elementa
- Najčešće se koriste vrednosti `left` I `right` koje ‘lepe’ element za levu ti. desnu ivicu



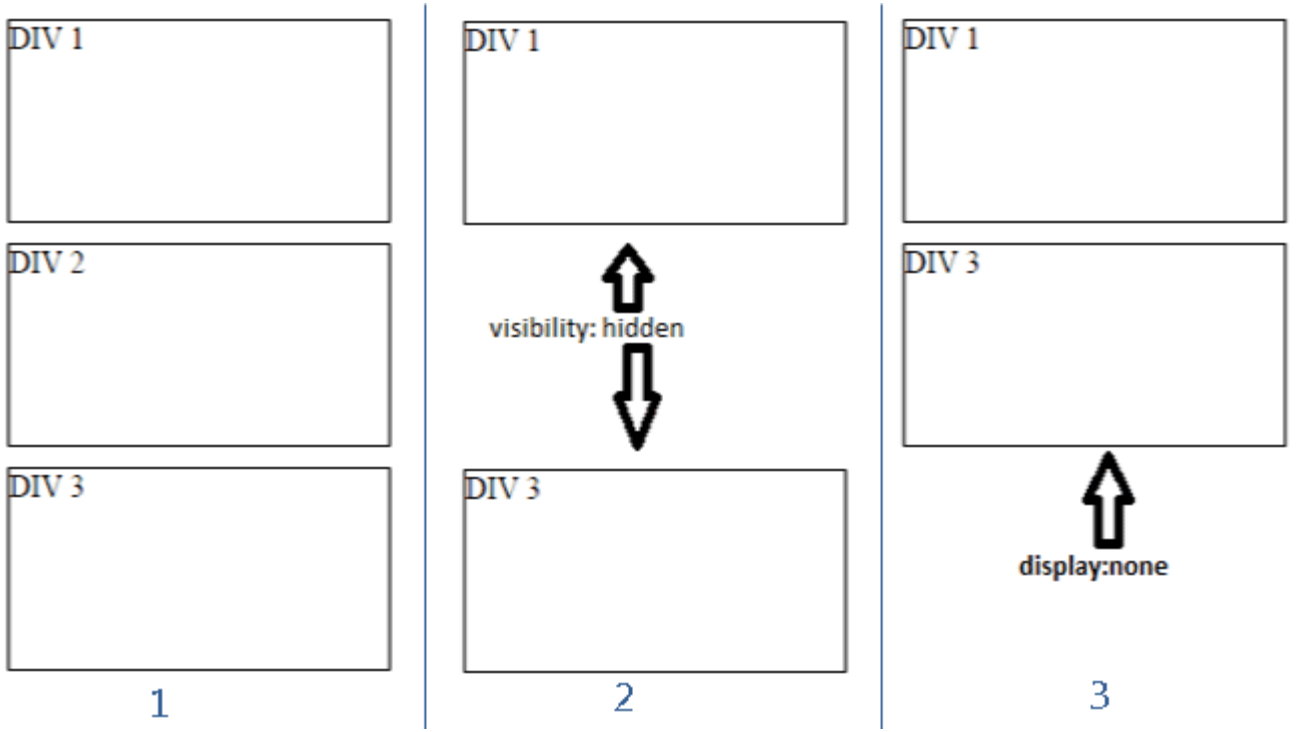
Pozicioniranje – float

- Ovaj način pozicioniranja se koristi i kada želimo daređamo susedne elemente jedan za drugim
- Možemo zamisliti kao da se na levoj (tj. desnoj) strani nalazi gravitacija koja privlači elemente tako da se oniređaju jedan pored drugog
- Kada želimo da zaustavimoređanje elemenata koristimo pravilo
  - `clear`
- Koje može imati vrednosti `right`, `left`, `both` u zavisnosti od toga kada želimo da zaustavimoređanje elemenata sa leve, desne



Nevidljivost elemenata

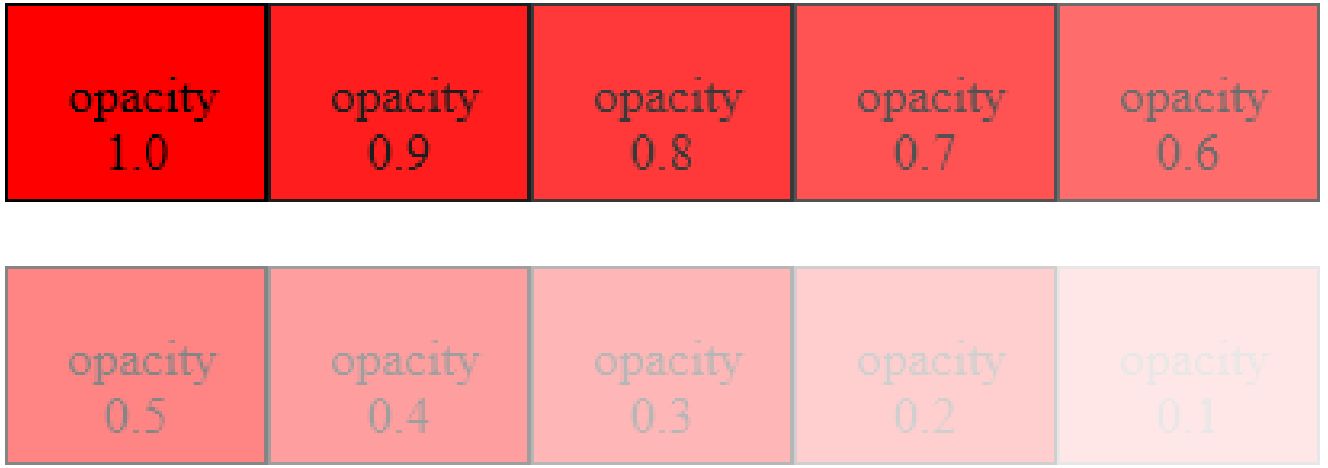
- Ukoliko želimo da neki element učinimo nevidljivim, to možemo uraditi na nekoliko načina:
  - `display: none;` - na ovaj način element neće zauzimati nikakav prostor, njegovi susedni elementi će se pozicionirati kao da on ne postoji
  - `visibility: hidden;` - na ovaj način element neće će biti nevidljiv ali će zauzimati prostor koji bi i inače zauzimao da je vidljiv



Nevidljivost elemenata

- `opacity` može imati vrednosti od 0-1, 0 označava potpunu providnost, 1 označava da element nije ni malo providan

  `opacity: 0;` - na ovaj način element neće će biti nevidljiv ali će zauzimati prostor koji bi i inače zauzimao da je vidljiv



Media query

- Koristimo ih kada želimo da primenimo različite stilove u zavisnosti od toga sa kojim uređajem smo otvorili stranicu (veliki monitor, ekran laptopa, tablet, telefon i sl.), na koji način smo otvorili stranicu, da li nam je prozor uvećan, umanjen itd.
- Najčešća upotreba media query je kada želimo da pozicioniramo elementre stranice tako da budu pogodni za prikaz na bilo kom uredjaju

```
  @media not|only tip and (svojstvo) {
...css kod...
  }
```

- *tip*: screen, print, all, ...
- *svojstvo*: width, height, device-width, device-height, max-width, max-height, min-width, min-height, aspect-ratio, orientation,

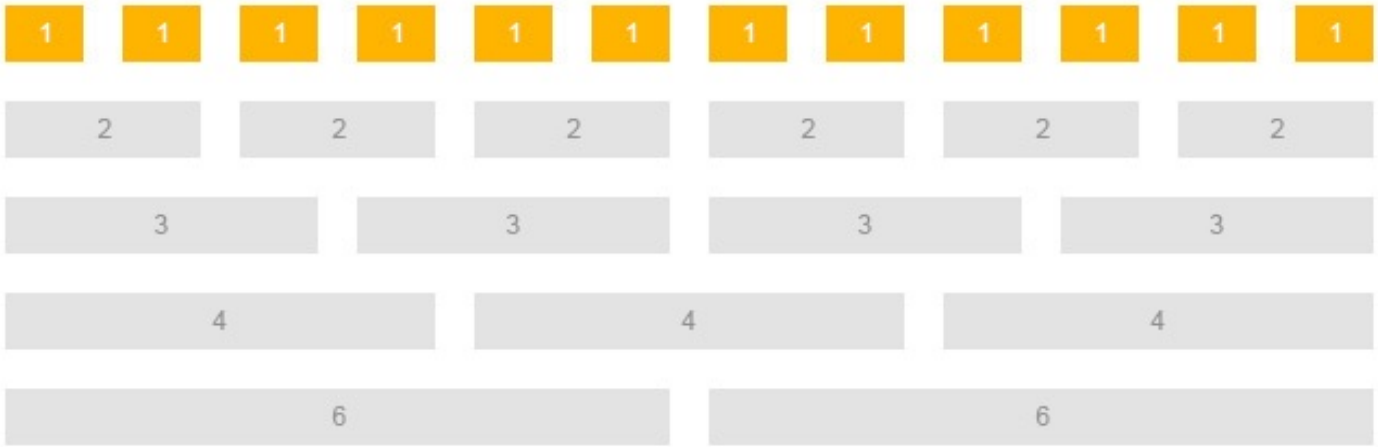
Media query

```
@media only screen and (min-width: 600px) {
#nav {
width: 80%;
}
}

@media (min-width: 600px) and (orientation: landscape) {
#nav {
width: 100%;
}
}
```

Bootstrap biblioteka

- https://getbootstrap.com/
- Pruža nam brojne CSS klase i još mnogo toga
- Uključujemo je pomoću `link` i `script` tagova (pogledati dokumentaciju)
- Mi ćemo je koristi za pozicioniranje ali biblioteka sadrži još mnogo korisnih svojstava

Bootstrap pozicioniranje



- Sadržaj je podeljen u skladu sa *grid* sistemom
- Glavnom elementu koji je omotač našeg sadržaja treba postaviti klasu `container` ili `container-fluid`
- Stranica se deli na redove, svakom redu postaviti klasu `row`
- Redovi se dele u 12 kolona koje se mogu spajati po potrebi (npr. 4-4-2-2, 6-3-3)

```
<div class="container">
  <div class="row">
    <div class="col-sm">
    prva kolona
    </div>
    <div class="col-sm">
     druga kolona
    </div>
    <div class="col-sm">
     treca kolona
    </div>
  </div>
</div>
```

Bootstrap pozicioniranje

- Veličina jedne kolone zavisi od veličine ekrana (ako imamo 12 kolona, jedna kolona je velika jednu dvananestinu ekrana)
- Kolonama postavljamo klase:

`col-*-*`

`col-xs-*` - telefoni (veličina ekrana extra small)

`col-sm-*` - tableti (small)

`col-md-*` - ekran racunara (medium)

`col-lg-*` - veliki ekran racunara (large)

`col-*-1`

`col-*-2`

...

`col-*-12` - broj predstavlja broj kolona koje zelimo da zauzmemo

npr. klasa `col-sm-4` predstavlja kolonu koja ce na telefonu zauzeti sirinu 4 susedne kolone

Font awesome biblioteka

- http://fontawesome.io
- Uključujemo je pomoću `link` taga
- Sadrži veliku kolekciju često korišćenih ikonica i sličica poput kante za smeće, plus kao dugme za dodavanje, itd.

- Ikonice pravimo pomoću `i` taga i CSS klase za tu ikonicu
- Npr:

```
<i class="fa fa-trash-o" aria-hidden="true"></i>
```