CSS - Cascading Style Sheets

- **CSS** je "jezik" koji opisuje stil HTML dokumenta
  - Opisuje kako HTML elementi trebaju izgledati
- **Cascading**
  - Odnosi se na način na koji CSS primjenjuje stilove jedan na drugi
- **Style Sheets**
  - Uređuju izgled web dokumenata

Prisjetimo se da moderno dizajniran i strukturiran web sastoji od:

- **HTML**: struktura
- **CSS**: definira izgled HTML elemenata
- **JavaScript**: djelovanje/ponašanje, unosi dinamiku
- **PHP, Python, Node.js i sl.**: pozadinska obrada podataka i upravljanje

Zašto CSS?

- Omogućava postavljanje specifičnog stila specifičnom HTML elementu
- Glavna prednost je da omogućava razdvajanje stila od sadržaja
- Definira stilove:
  - Dizajn
  - Raspored
  - Varijacije u prikazu
    - Npr. Za različite zaslone različitih uređaja
- Preporuka je sva oblikovanja izbaciti iz HTML dokumenta
  - Držati oblikovanja u zasebnoj CSS datoteci

CSS sintaksa

CSS sintaksa se sastoji od dva dijela:

- **selektor:** upućuje na HTML element koji želimo stilizirati
- **deklaracija:** blok deklaracije sadrži jednu ili više deklaracija, dok svaka deklaracija sadrži CSS svojstvo i pripadajuću vrijednost

˙Dakle, prvo se navodi selektor, zatim se unutar `{}` definira deklaracija:

```
selektor {
    svojstvo: vrijednost;
    svojstvo-2: "vrijednost-2";
}
```

Vrste selektora

Selektor elementa: može upućivati na elemente prema njihovom nazivu (sve elemente unutar HTML dokumenta):

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

h1 {
    text-align: center;
    background-color: orange;
}
```

`id` selektor

Koristi atribut `id` elementa za označavanje specifičnih HTML elemenata (`id` elementa bi trebao biti jedinstven unutar HTML dokumenta):

```
#element-sa-id-om {
    text-align:center;
    color: blue;
}
```

`class` selektor

Označava elemente specifičnog `class` atributa (npr. `<p class="naziv-klase">...</p>`):

```
.naziv-klase {
    text-align: center;
    color: red;
}
```

Napomena: HTML elementi mogu imati više klasa (odvajaju se razmakom):

```
<p class="naziv-klase naziv-klase2">Ovaj odlomak se referira na dvije klase.</p>
```

`class` selektor + specifični HTML element

Označava specifičan HTML element specifičnog `class` atributa:

```
p.naziv-klase {
    text-align: center;
    color: red;
}
```

Grupiranje selektora

Kada nam se neki stil ponavlja za više elemenata, ne trebamo ga više puta kopirati već se može isti stil definirati za više **selektora** odjednom. Samo je potrebno navesti željene selektore odvojene zarezom:

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

Naravno da selektori ne trebaju biti iste **vrste** da bi definirala grupna deklaracija, pa je tako moguće i sljedeće:

```
h1, .naziv-klase, p.naziv-klase2, h2 {
    text-align: center;
    color: red;
}
```

Uključivanje CSS-a u HTML

Moguća su 3 načina:

- ***Inline style***
- ***Internal/embedded style sheet***
- ***External style sheet***

Inline style

Koristi se kada želimo definirati jedinstveni stil za neki element. Željenom elementu u atribut `style` dodajemo deklaraciju stila:

```
<p stlye="color:white; background-color:red;">...</p>
```

Ovakva vrsta deklaracije stila se uglavnom izbjegava zbog jednostavnosti održavanja koda.

Internal/embedded style sheet

Stilovi se deklariraju na do sada opisani način i smještaju se unutar tag `style` koji se nalazi u tagu `head`:

```
...
<head>
    <style>
        body {
            background-color: gray;
        }
        h1 {
            color: red;
            text-align: center;
        }
    </style>
</head>
...
```

External style sheet

Sva stiliziranja su sadržana u vanjskoj tekstualnoj datoteci (nastavak *.css*). Vanjska se datoteka (ili više njih) u HTML dokumentu referencira/poziva pomoću taga `<link>` koji se smješta unutar taga `<head>`.

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

Npr.

Datoteka **main.css**:

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    text-align: center;
}
```

Datoteka **index.html:**

```
<!DOCTYPE>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="main.css">
    </head>
    <body>
        <h1>Naslov</h1>
    </body>
</html>
```

Dodatan razlog zašto je ova vrsta deklaracije najpogodnija jest činjenica da se neka web aplikacija sastoji od više *.html* stranica gdje ćemo željeti zadržati konzistentnost izgleda. Dakle, možemo sva stil definirati u **jednoj** (može biti i više) *.css* datoteci i koristiti ga u **više** *.html* stranica pri čemu ćemo jednostavno postići konzistentnost izgleda.

Redoslijed deklaracija

Redoslijed važnosti deklaracija prema kojima se vrši primjena stila na element:

- ***Inline style*** (unutar HTML elementa)
- ***External and internal style sheets*** (unutar `<head>` dijela ili iz `.css` datoteke)
- ***Browser default*** (svaki preglednik ima unaprijed definirane stilove pojedinih elemenata)

Višestruke deklaracije: ako je neko svojstvo deklarirano više puta na različitim mjestima, primjenjuje se ono svojstvo koje se zadnje učita.

Npr.

Datoteka `main.css`:

```
h1 {
    color: blue;
}
```

Datoteka `index.html`:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="main.css">
        <style>
            h1 {
                color: orange;
            }
        </style>
    </head>
<body>

<h1>Koje sam boje?</h1>

</body>
</html>
```

U ovom slučaju će `<h1>` biti obojan narančastom bojom jer je ta deklaracija posljednja.

Kada bismo zamijenili `<link>` i `<style>` takove tako da je `<link>` smješten **ispod** taga `<style>`, tada bi `<h1>` bio obojan **plavom** bojom.

Međutim! kada bi se u navedenom primjeru u elementu preko atributa `style` definirala crvena boja (`<h1 style="color:red;">Koje sam boje?</h1>`), sve prethodne definicije bi postale nebitne i `<h1>` bi bio **crvene** boje.

CSS kombinatori (engl. CSS Combinators)

CSS selektor može sadržavati više od jednog jednostavnog selektora. Postoje četiri različita kombinatora u CSS-u:

- **selektor potomaka (engl. descendant selector)** (razmak)
- **selektor djece (engl. child selector)** (>)
- **selektor susjedne braće (engl. adjacent sibling selector)** (+)
- **selektor opće braće (engl. general sibling selector)** (~)

Descendant selector

Selektira sve elemente koji su potomci specifičnom elementu. Kada bi željeli selektirati sve elemente `<p>` unutar elementa `<div>` napisali bi sljedeće:

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

Što znači da ćemo imati sljedeću situaciju:

```
...
<div>
    <p>Ja sam žute boje jer sam unutar "div" elementa</p>

    <section>
        <p>I ja sam žute boje</p>
    </section>
</div>

<p>Ja nisam žute boje jer nisam unutar "div" elementa</p>
...
```

Child selector

Selektira sve elemente koji su djeca specifičnom elementu. Sličan je selektoru *descendant* ali se ograničava samo na svoju djecu.

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

Što znači da ćemo imati sljedeću situaciju:

```
...
<div>
    <p>Ja sam žute boje jer sam dijete "div" elementa</p>

    <section>
        <p>Ja nisam žute boje jer sam dijete section elementa</p>
    </section>
</div>

<p>Ja nisam žute boje jer nisam dijete "div" elementa</p>
...
```

Adjacent sibling selector

Selektira sve susjedne elemente koji su braća specifičnom elementu. Da bi elementi bili *braća* moraju imati istog roditelja, a *"adjacent"* znači *"odmah nakon"*.

Sljedeći primjer selektira sve `<p>` elemente koji su smješteni odmah nakon `<div>` elementa:

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

Što znači da ćemo imati sljedeću situaciju:

```
...
<div>
    <p>Ja nisam žute boje jer sam dijete "div" elementa</p>

    <section>
        <p>Ja isto nisam žute boje</p>
    </section>
</div>

<p>Ja sam žute boje jer sam brat "div" elementu i nalazim se odmah nakon
njega</p>
<p>Iako sam brat "div" elementu, ja nisam žute boje jer mu nisam prvi brat</p>
...
```

General sibling selector

Selektira sve elemente koji su braća specifičnog elementa. Slično *"Adjacent sibling selector"*-u samo što nije ograničeno na samo prvi susjedni element:

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

Što znači da ćemo imati sljedeću situaciju:

```
...
<p>Ja nisam žute boje jer dolazim prije "div" elementa</p>

<div>
    <p>Ja nisam žute boje jer sam dijete "div" elementa</p>

    <section>
        <p>Ja isto nisam žute boje</p>
    </section>
</div>

<p>Ja sam žute boje jer sam brat "div" elementu i nalazim se nakon njega</p>
<small>Ja sam brat "div" elementu, ali nisam "p" element pa nisam žute
boje</small>
<p>Ja sam isto žute boje jer sam brat "div" elementu i nalazim se nakon
njega</p>
...
```

Ulančavanje selektora

Prethodno smo promatrali jednostavne primjere CSS kombinatora, međutim u izrazu selektora se može koristiti više pod-selektora bez obzira na vrstu kombinatora. Prema tome su moguće sljedeće definicije:

```
div p span a {
    color: yellow;
}
div + div.naziv-klase > p {
    color: yellow;
}
#id-elementa div.naziv-klase > div > p a {
    color: yellow;
}
```

Svojstva

Background

CSS svojstvo `background` koristimo za definiranje pozadinskih efekata elemenata.

Tablica svojstava:

Naziv svojstva	Vrijednosti	Opis
`background-color`	**Naziv boje** **HEX** **RGB**	Definira pozadinsku boju
`background-image`	**url("slika.jpg")**	Definira slikovnu datoteku koja će se koristiti za pozadinu. Slika se po potrebi ponavlja kako bi ispunila cijeli element
`background-repeat`	**repeat** **repeat-x** **repeat-y** **no-repeat** **initial** **inherit**	Koristimo kad želimo definirati način ponavljanje slike kod gornjeg svojstva
`background-attachment`	**fixed** **scroll**	Koristimo kad želimo fiksirati sliku pozadine da se ne miče tijekom skrolanja
`background-position`	**top** **bottom** **left** **right** **center** **inherit**	Definira poziciju pozadine. Možemo koristiti više vrijednosti kod definiranja svojstva

Primjer:

```
body {
  background-color: purple;
  background-image: url("ninja.jpg");
  background-repeat:  repeat-y;
  background-attachment: fixed;
  background-position: center;
}
```

Isto se može definirati **skraćenim** načinom kroz svojstvo `background`:

```
body {
    background: purple url("ninja.jpg") repeat-y fixed center;
}
```

Napomena: Kod skraćenog načina pisanja treba samo upamtiti redoslijed pisanja vrijednosti svojstava:

1. `background-color`
2. `background-image`
3. `background-repeat`
4. `background-attachment`
5. `background-position`

Text

Za stiliziranje teksta možemo koristiti razna svojstva, ovo su neka od njih:

Naziv svojstva	Vrijednosti	Opis
`color`	**Naziv boje** **HEX** **RGB**	Definira boju teksta
`text-align`	**center** **left** **right** **justify**	Definira vodoravno poravnanje teksta
`text-decoration`	**none** **line-through** **underline** **overline**	Definira „ukrase” teksta. Ovim svojstvom možemo ukloniti donju crtu kod linkova
`text-transform`	**upercase** **lowercase** **capitalize**	Koristimo kad želimo tekst promijeniti u velika ili mala slova
`line-height`	**numerička vrijednost**	Definira razmak među redcima, tj. Prored
`text-shadow`	**x y boja**	Definira x i y udaljenost sjene od teksta, te njenu boju

Primjer:

```
p {
  color: purple;
  text-align: center;
  text-decoration: underline;
  text-transform: capitalize;
  line-height: 20px;
  text-shadow: 1px 1px red;
}
```

Font

Neka od svojstava fonta:

Naziv svojstva	Vrijednosti	Opis
`font-family`	**"Times New Roman"** **Times** **serif** **...**	Definira naziv fonta koji će se koristiti u elementu
`font-style`	**normal** **italic** **oblique**	Definira stil fonta
`font-weight`	**normal** **bold** **100** **...** **900**	Definira debljinu fonta
`font-size`	**numerička vrijednost**	Definira veličinu fonta

Svojstvo `font-family` bi trebalo sadržavati nazive nekoliko fontova radi sigurnosti prikaza teksta. Ako web preglednik ne podržava prvi font, probat će prikazati sljedeće navedeni font i tako dalje. Preporuka je prvo unijeti font koji želimo, dok na zadnje mjesto stavljamo neki generički font.

Primjer:

```
p {
    font-family: "Times New Roman", Times, serif;
    font-style: italic;
    font-weight: bold;
}
```

Napomena: koristimo navodnike ukoliko naziv fonta ima više od jedne riječi.

Svojstvo `font-size` služi za određivanje veličine fonta. Veličina se može postaviti kroz vrijednost u pikselima, ali preporučuje se koristiti vrijednost u mjernoj jedinici `em`. `1em` = trenutna zadana veličina fonta, što je kod mnogih preglednika `16px` što znači da je `1em = 16px`.

Razlog uvođenja nove mjerne jedinice jest mogućnost prilagođavanja veličine fonta prema korisničkim postavkama (DPI - *dots per inch*), umjesto da su veličine fontova fiksne u svakom slučaju.

Primjer:

```
p {
    font-size: 14px;
}
h1 {
    /*font-size: 40px; */
    font-size: 2.5em; /* 40px/16=2.5em */
}
```

Border

Naziv svojstva	Vrijednosti	Opis
`border-style`	**dotted** **dashed** **solid** **double** **groove** **ridge** **inset** **outset** **none** **hidden**	Definira stil cijelog obruba
`border-width`	**numerička vrijednost**	Definira širinu obruba Atribut `border-stlye` mora biti definiran!
`border-color`	**Naziv boje** **HEX** **RGB**	Definira boju obruba Atribut `border-stlye` mora biti definiran!
`border-radius`	**numerička vrijednost**	Definira jačinu zaokruživanje ruba Atribut `border-stlye` mora biti definiran!

Primjer:

```
div {
    border-style: solid;
    border-width: 5px;
    border-color: blue;
    border-radius: 5px;
}
```

Navedeno se može definirati kraćim zapisom navodeći redoslijedno `with`, `style`, `color`:

```
div {
    border: 5px solid blue;
}
```

Definiranje pojedine strane obruba

Naziv svojstva	Vrijednosti	Opis

`border-top-style`
`border-right-style`
`border-bottom-style`
`border-left-style`
	dotted
	dashed
	solid
	double
	groove
	ridge
	inset
		Definira stil pojedine strane obruba

Naziv svojstva	outset Vrijednosti none	Opis
	hidden	

Napomena: također se može zapisati u kraćoj varijanti

Primjer:

```
div {
    border-top: 1px solid blue;
    border-right: 1px solid blue;
    border-bottom: 1px solid blue;
    border-left: 1px solid blue;

    /*I radius se može odrediti za svaki kut zasebno*/
    border-bottom-left-radius: 10px;
    border-bottom-right-radius: 5px;
    border-top-left-radius: 20px;
    border-top-right-radius: 10px;
}
```

Margin

Margine koristimo kako bismo razmaknuli prostor od željenog elementa. Svojstvo margin napravi prazan prostor **okolo, izvan obruba** elementa.

Primjer:

```
div.example {
    margin: 20px 15px 10px 50px;
}
/* Isto kao: */
div.example {
    margin-top: 20px;
    margin-right: 15px;
    margin-bottom: 10px;
    margin-left: 5px;
}
```

Usporedba elemenata sa i bez margine:

```
<!-- Bez margine -->
<div style="border:1px solid gray;">
    <div style="border:1px solid blue;">
        "margin: 0"
    </div>
</div>

<!-- Sa marginom -->
<div style="border:1px solid gray;">
    <div style="border:1px solid blue;margin: 20px 15px 10px 5px">
        "margin: 20px 15px 10px 5px"
    </div>
</div>
```

Rezultat:

"margin: 0"

"margin: 20px 15px 10px 5px"

Kada element želimo **poravnati horizontalno na sredinu** u odnosu na spremnik koji ga sadrži možemo koristiti vrijednost `auto`:

```
div.example {
    width: 300px;
    margin: auto;
    border: 1px solid blue;
}
div.parent {
    border: 1px solid red;
}
```

```
<div class="parent">
    <div class="example">
        Poravnanje elementa na sredinu
    </div>
</div>
```

Rezultat:

Poravnanje elementa na sredinu

Margin collapse

Gornja i donja margina elemenata se ponekad spajaju u jednu marginu čija je vrijednost jednaka većoj vrijednosti od oba elementa (ne vrijedi za okomite margine, lijevu i desnu).

```
div.first {
    margin-bottom: 30px;
}
div.second {
    margin-top: 20px;
}
div.joined {
    margin-bottom: 50px;
}
div.first, div.second, div.joined, div.empty {
    border: 1px solid gray;
}
```

```
<div class="first">
    Prvi element
</div>
<div class="second">
    Drugi element
</div>
<!-- kada premjestimo svu marginu na jedan element dobijemo drugačije ponašanje
-->
<div class="joined">
    Prvi element
</div>
<div class="empty">
    Drugi element
</div>
```

Rezultat:

Prvi element

Drugi element

Prvi element

Drugi element

Padding

Padding također koristimo kako bismo razmaknuli prostor od željenog elementa, ali se prostor koji stvaramo/mičemo nalazi **unutar obruba** elementa.

Primjer:

```
div.example {
    padding: 20px 15px 10px 50px;
}
/* Isto kao: */
div.example {
    padding-top: 20px;
    padding-right: 15px;
    padding-bottom: 10px;
    padding-left: 5px;
}
```

Usporedba elemenata sa i bez margine:

```
<!-- Bez padding-a -->
<div style="border:1px solid blue;">
    "padding: 0"
</div>

<!-- Sa padding-om -->
<div style="border:1px solid blue;padding: 20px 15px 10px 5px">
    "padding: 20px 15px 10px 5px"
</div>
```

Rezultat:

"padding: 0"	"padding: 20px 15px 10px 5px"

Width & Height

Napomena: ova svojstva ne uključuju margine, obrub i padding. Ovime se postavljaju dimenzije unutar ta tri svojstva.

```
div.pct {
    height: 30px;
    width: 50%;
    background-color: powderblue;
}
div.px {
    height: 30px;
    width: 500px;
    background-color: blue;
}
```

```
<div>
    <div class="pct"></div>
</div>
<div>
    <div class="px"></div>
</div>
```

Rezultat:

Max & Min

Koriste se za postavljanje ograničenja na širinu i/ili visinu elementa. Ova ograničenja su korisna kada želimo dobiti određeno ponašanje kod promjene rezolucije i/ili zoom-a preglednika.

```
div {
    width: 50%;
    height: 50%;

    max-width: 300px;
    min-width: 50px;

    max-height: 300px;
    min-height: 50px;
}
```

Display

CSS `display` svojstvo je jedno od važnijih svojstava za upravljanje oblikovanja sadržaja, njime definiramo kako će element biti prikazan.

Svaki HTML element ima zadanu vrijednost `display` svojstva koja ovisi o vrsti elementa:

- Block-level elements – zadana vrijednost `display: block;`
- Inline elements – zadana vrijednost `display: inline;`

Što znači da ovim svojstvom možemo definirati tip kutije HTML elementa.

Napomena: svojstvo `display` samo mijenja način prikaza elementa, ne i njegovu vrstu što znači da i dalje vrijedi da ***inline*** elementi ne mogu sadržavati druge tipove elementa dok ***block-level*** mogu.

Vrijednosti svojstva `display`:

Vrijednost	Opis
`inline`	Prikazuje element kao *inline* element (npr `<span>`). Svojstva `width` i `height` neće moći djelovati na element
`block`	Prikazuje element kao *block* element (npr `<div>`). Element započinje u novom retku i zauzima cijelu dostupnu dužinu
`inline-block`	Prikazuje element kao *inline-level* blokovski spremnik. Element je formatiran kao *inline* element, ali može imati definirana svojstva `width` i `height`
`none`	Element je u potpunosti skriven sa stranice

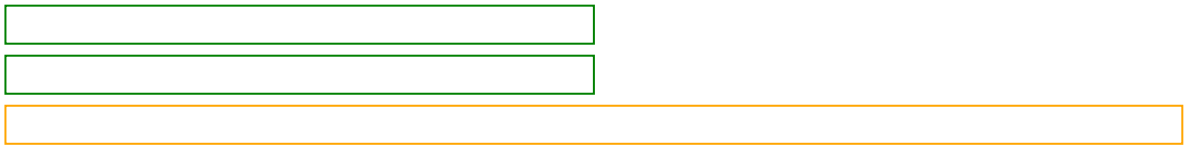
[Cijeli popis svih vrijednosti za svojstvo `display`]

Primjer:

```
.none {
    display: none;
    height: 20px;
    border: 1px solid red;
}
.block {
    display: block;
    height: 20px;
    border: 1px solid orange;
}
.inline-block {
    display: inline-block;
    height: 20px;
    width: 50%;
    border: 1px solid green;
}
```

```
<div>
    <div class="none"></div>
    <div class="inline-block"></div>
    <span class="inline-block"></span>
    <span class="block"></span>
</div>
```

Rezultat:



Position

Naziv svojstva	Vrijednosti	Opis
`position`	**Static** **Relative** **Fixed** **Absolute**	Definira način pozicioniranja elementa unutar nadređenog elementa i/ili dokumenta

`top`
`right`
`bottom`
`left`
	numerička vrijednost	Pozicioniranje elemenata s obzirom na određenu stranu elementa.
		Svojstvo `position` mora biti postavljeno da bi svojstva mogla djelovati

---

Static

Zadano pozicioniranje za sve elemente ukoliko nije drugačije navedeno. Na statički pozicionirane elemente na utječu svojstva `top`, `right`, `bottom` i `left`. Dakle, koristeći ovaj način, element nije posebno pozicioniran.

```
div.static {
    position: static;
    border: 1px solid blue;
}
```

Relative

Na ovaj način pozicioniramo element relativno u odnosu na njegovu uobičajenu poziciju, koriste se svojstva `top`, `right`, `bottom` i `left`.

```
div.relative {
    position: relative;
    width: 100px;
    height: 100px;
}
div.relative.blue {
    left: 30px;
    border: 1px solid blue;
}
div.relative.red {
    right: -30px;
    bottom: -30px;
    border: 1px solid red;
}
```

```
<div style="border: 1px solid gray;">
    <div class="relative blue"></div>
    <div class="relative red"></div>
</div>
```

Rezultat:

Fixed

Element s ovom vrijednošću je uvijek na istom mjestu relativno u odnosu na naše gledište. Element je na istom mjestu čak i ako skrolamo stranicu. Koriste se svojstva `top`, `right`, `bottom` i `left`.

Napomena: ovako pozicioniran element ne ostavlja prazninu na mjestu gdje bi inače bio pozicioniran (kao da je izrezan sa stranice i zalijepljen na naš ekran).

```
div.fixed {
    position: fixed;
    top: 10px;
    right: 10px;
    width: 100px;
    height: 100px;
    border: 2px solid blue;
}
div.normal {
    border: 1px solid gray;
    height: 100px;
    margin-bottom: 5px;
}
```

```
<div>
    <div class="normal"></div>
    <div class="normal"></div>
    <div class="fixed"></div>
</div>
```

Rezultat:

Absolute

Element s ovom vrijednošću se pozicionira relativno u odnosu na svog najbližeg pretka. Za razliku kao kod `fixed` vrijednosti koja se pozicionira u odnosu na gledište. Ukoliko element s ovom vrijednošću nema svog pretka, koristi se cijela stranica (*Body* element).

Napomena: pozicioniranim elementima se smatraju samo oni koji imaju vrijednost svojstva `position` različitu od `static`.

```
div.absolute {
    position: absolute;
    top: 30px;
    right: 10px;
    width: 100px;
    height: 100px;
    border: 2px solid blue;
}
div.normal {
    border: 1px solid gray;
    height: 100px;
    margin-bottom: 5px;
}
```

```
<div>
    <div class="normal"></div>
    <div class="normal"></div>
    <div class="absolute"></div>
</div>
```

Rezultat:

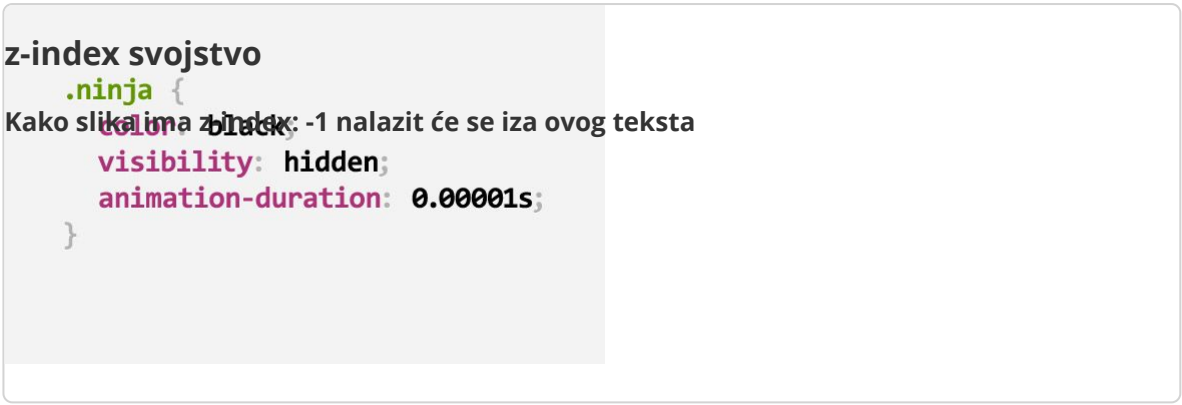
Z-index

Omogućava preklapanje elemenata, moguće su pozitivne i negativne vrijednosti.

```
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
    height:200px;
}
```

```
<div>
    <h4>z-index svojstvo</h4>
    <p><b></b>Kako slika ima z-index: -1 nalazit će se iza ovog teksta</b></p>
    <img src="ninja.jpg"/>
</div>
```

Rezultat: (napomena: u normalnom slučaju bi slika bila iznad elementa)



Float

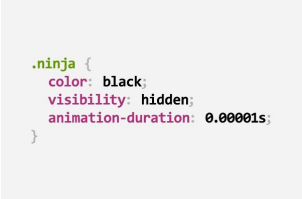
Definira treba li određeni element *plutati*. Koristi se u kombinaciji s `clear` svojstvom, a služi za kontrolu „toka" plutanja.

Najjednostavnija primjena je kod prelamanja teksta okolo slike:

```
img {
    float: right;
    margin: 0 0 10px 10px;
    height: 100px;
}
```

```
<div>
    <img src="ninja.jpg">
    <p>Lorem Ipsum...</p>
</div>
```

Rezultat:



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Clear

Koristi se za upravljanje plutanja elemenata. Sadržaj elemenata nakon elementa s `float` svojstvom se prelama okolo tog elementa. `clear` svojstvo definira oko koje strane `float` elementi ne smiju plutati.

```
p {
    clear: right;
}
img {
    float: right;
    margin: 0 0 10px 10px;
    height: 100px;
}
```

```
<div>
    <img src="ninja.jpg">
    <p>Lorem Ipsum...</p>
</div>
```

Rezultat:

```
.ninja {
  color: black;
  visibility: hidden;
  animation-duration: 0.00001s;
}
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

The clearfix Hack

Ukoliko je element viši od elementa koji ga sadrži, te je postavljen da *pluta*, *prelit* će se izvan elementa koji ga sadrži.

Primjer:

```
.clearfix {
    overflow: auto;
}
img {
    float: right;
    margin: 0 0 10px 10px;
    height: 100px;
}
div.parent {
    background: lightgray;
    min-height: 20px;
}
/* Moderniji pristup */
.clearfix::after {
    content: "";
    clear: both;
    display: table;
}
```

```
<div class="parent">
    <img src="ninja.jpg">
</div>

<!-- reset float svojstava -->
<div style="clear:both;"></div>
<hr>

<!-- sa overflow-om -->
<div class="parent clearfix">
    <img src="ninja.jpg">
</div>
```

```
.ninja {
  color: black;
  visibility: hidden;
  animation-duration: 0.00001s;
}
```

```
.ninja {
  color: black;
  visibility: hidden;
  animation-duration: 0.00001s;
}
```

Liste

Naziv svojstva	Vrijednosti	Opis
`list-style`		„Kratki način” definiranja svih svojstava za oblikovanje listi
`list-style-type`	**disc** **circle** **cjk-ideographic** **decimal** **decimal-leading-zero** **none** **square** **initial** **inherit** **...**	Definira tip simbola liste
`list-style-position`	**inside** **outside** **initial** **inherit**	Definira položaj simbola liste (unutar ili izvan sadržaja)
`list-style-image`	**none** **url** **initial** **inherit**	Definira sliku za simbol liste

Primjer:

```
ul {
    list-style-type: circle;
    list-style-position: inside;
    /*Slika: list-style-image: url("car_icon.jpg");*/
}
```

```
Automobili:
<ul>
    <li>Ferrari</li>
    <li>Alfa Romeo</li>
    <li>Pagani</li>
</ul>
```

Rezultat:

Automobili:

- Ferrari
- Alfa Romeo
- Pagani

Pseudo-classes

Koristi se kako bi se odredilo specifično stanje elementa. Npr. može se koristiti kada želimo:

- stilizirati element kada korisnik mišom prijeđe iznad njega
- stilizirati posjećene i neposjećene linkove drugačije
- stilizirati element kada dobije fokus

Sintaksa:

```
selector:pseudo-class {
  property: value;
}
```

Link pseudo-classes

```
/* neposjećeni link */
a:link {
  color: blue;
}

/* posjećeni link */
a:visited {
  color: black;
}

/* miš iznad linka */
a:hover {
  color: orange;
}

/* odabran link */
a:active {
  color: red;
}
```

```
<a href="index.html" target="_blank">Idi na index.html</a>
```

Pseudo-classes i CSS klase

Primjer 1:

```
div.box {
    background-color: blue;
    height:30px;
}
div.box:hover {
    background-color: orange;
}
```

```
<div class="box"></div>
```

Primjer 2:

```
p {
    display: none;
    background-color: yellow;
    padding: 20px;
}

div:hover p {
    display: block;
}
```

```
<div>Prođi mišom iznad mene pa će se prikazati p element
    <p>Tada! Evo me!</p>
</div>
```

`:first-child` pseudo-class

Odabire određeni element koji je prvo dijete elementa nad kojim se stil primjenjuje:

```
p:first-child {
  color: blue;
}
```

```
<p>Ovo će biti plavi tekst</p>
<p>Neki tekst</p>

<div>
    <p>Ovo će biti plavi tekst</p>
    <p>Neki tekst</p>
</div>
```

Rezultat:

Ovo će biti plavi tekst

Neki tekst

Ovo će biti plavi tekst

Neki tekst

CSS komentari

Kao i kod HTML-a, web preglednik ignorira komentare kao i naredbe koje su unutar njih. Komentari se zaokružuju znakovima `/*` i `*/`.

```
p {
    color: yellow;
    /*background: red; Ovo je komentar*/
}
/*
Ovo je komentar
u više redaka
*/
```
