

Tok HTML dokumenta

U dosadašnjem toku ovoga modula mogli ste da vidite da HTML i CSS jezici poseduju određena pravila na osnovu kojih se obavlja raspoređivanje HTML elemenata na stranici. Do sada smo se upoznali sa jednim svojstvom kojim se može uticati na takav postupak. Naravno, reč je o `display` svojstvu. Pored ovog svojstva, postoje još neka čije vrednosti utiču na raspoređivanje elemenata.

Inicijalno, svi elementi unutar HTML stranice raspoređuju se po pravilima prirodnog toka (engl. *normal flow*). To praktično znači da se block elementi raspoređuju jedan ispod drugog, pri čemu zauzimaju kompletnu dostupnu širinu, a inline i inline-block elementi ređaju se jedan pored drugog u horizontalnoj liniji. Ipak, pored `display` svojstva, postoje još neka svojstva koja mogu uticati na tok HTML dokumenta. Stoga će u lekciji pred vama biti obrađeno sledeće:

- pozicioniranje elemenata korišćenjem `position` CSS svojstva;
- CSS svojstva `float` i `clear`;
- `z-index` CSS svojstvo;
- `overflow` CSS svojstvo;
- `visibility` CSS svojstvo;
- `box-sizing` CSS svojstvo.

Pozicioniranje

Prvi pojam koji će biti objašnjen u ovoj lekciji o raspoređivanju elemenata jeste pojam *pozicioniranja*. Svaki HTML element može da bude pozicioniran na jedan od sledeća četiri načina:

- `static`;
- `relative`;
- `absolute`;
- `fixed`.

Upravo prikazani različiti načini pozicioniranja zapravo su različite vrednosti koje može imati CSS svojstvo `position`.

Pozicioniranja `static` i `relative` zadržavaju elemente u prirodnom toku, dok pozicioniranja `absolute` i `fixed` izbacuju element iz prirodnog toka stranice.

U nastavku će detaljno biti objašnjena različita pozicioniranja.

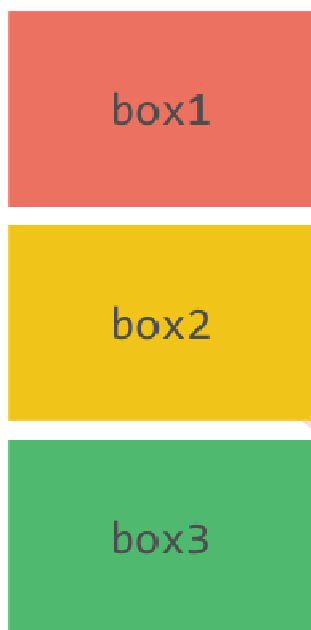
Statičko pozicioniranje

Statičko pozicioniranje je pozicioniranje koje se podrazumeva kod HTML elemenata. Ovim se pozicioniranjem elementi raspoređuju po normalnom toku dokumenta. Kada se za neki element kaže da **nije pozicioniran**, zapravo se misli na to da poseduje statičko pozicioniranje. Za ilustraciju statičkog, podrazumevanog pozicioniranja, biće iskorišćena sledeća HTML struktura:

```
<div id="box1">  
</div>  
  
<div id="box2">  
</div>  
  
<div id="box3">  
</div>
```

Prikazanu strukturu čine tri `div` elementa i to je struktura koja će biti korišćena za demonstraciju svih ostalih pozicioniranja. Da bi se elementi statički pozicionirali, nije potrebno postavljati nikakvu vrednost svojstva `position`, s obzirom na to da je `static` podrazumevana vrednost.

Ukoliko se prikazanim `div` elementima postavi visina i širina, donja margina i boja pozadine, oni će na stranici formirati prikaz kao na slici 15.1.



Slika 15.1. Tri `div` elementa raspoređena po pravilima normalnog toka

S obzirom na to da je reč o tri block elementa, svi oni se prikazuju u zasebnim redovima, jedan ispod drugog.

Relativno pozicioniranje

Element sa relativnim pozicioniranjem pozicioniran je relativno u odnosu na svoju normalnu poziciju koju bi imao da je pozicioniran kao statički element. Na taj način relativno pozicioniranje ne izbacuje element iz normalnog toka dokumenta, već samo omogućava da se na njegovu poziciju preciznije utiče. To uticanje na poziciju elementa postiže se korišćenjem četiri CSS svojstva: `top`, `right`, `bottom` i `left`. Ova svojstva drugačije se nazivaju *Offset* svojstva.

Sledeći primer ilustruje razliku između statičkog i relativnog pozicioniranja. HTML kod primera biće identičan prethodnom.

```
<div id="box1">
</div>

<div id="box2">
</div>

<div id="box3">
</div>
```

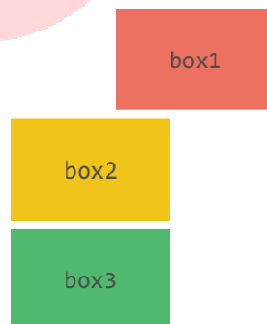
CSS stilizacija koja se odnosi na sva tri navedena div elementa je sledeća:

```
div{
    width: 200px;
    height: 100px;
    margin-bottom: 20px;
}
```

Na ovaj način se postavlja veličina div elemenata i njihova donja margina. Još uvek su svi elementi statički pozicionirani. Sledeći CSS opis postavlja relativno pozicioniranje za prvi div element:

```
#box1{
    position: relative;
    left: 120px;
}
```

Na ovaj način div element sa id-em, `box1`, dobija relativno pozicioniranje, i to definisanjem vrednosti `relative` za svojstvo `position`. Vrednost `relative` ovoga svojstva sama po sebi nema nikakav efekat dok se ne navede neko od offset svojstava. U primeru je navedeno offset svojstvo `left` i na taj način je element izmešten za 120 piksela udesno u odnosu na svoj prirodni položaj. Zapravo, vrednost od 120 px ukazuje na rastojanje leve ivice elementa od pozicije na kojoj bi se našla njegova leva ivica da je element pozicioniran statički. Zbog toga se navođenjem pozitivne vrednosti za `left` svojstvo element pomera udesno. Slika 15.2. to ilustruje.



Slika 15.2. Relativno pozicioniran box1 div element

Apsolutno pozicioniranje

Kaže se da je neki element apsolutno pozicioniran onda kada je pozicioniran relativno najbližem pretku koji je pozicioniran na bilo koji način, osim korišćenjem `static` pozicioniranja. Ukoliko element ne poseduje nijednog pretka koji nije pozicioniran statički, pozicija elementa utvrđuje se na osnovu elementa `body`.

Apsolutno pozicioniranje jedno je od pozicioniranja koje izbacuje element iz prirodnog toka dokumenta.

Sledeći primer ilustruje apsolutno pozicionirani `div` element. HTML kod primera biće identičan prethodnom, kada je demonstrirano relativno pozicioniranje:

```
<div id="box1">
</div>

<div id="box2">
</div>

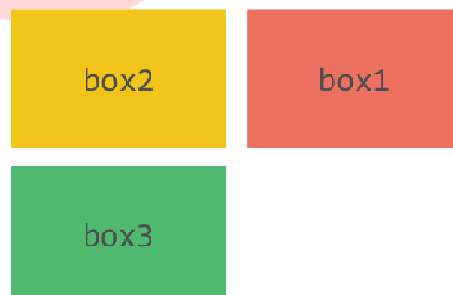
<div id="box3">
</div>
```

Od tri prikazana `div` elementa apsolutno će biti pozicioniran prvi. S obzirom na to da spomenuti element nema nikakvih potomaka pre `body` elementa, apsolutno pozicioniranje biće obavljeno u odnosu na `body` element. Kako bi se prvi `div` apsolutno pozicionirao, biće napisan sledeći CSS opis:

```
#box1{
  position: absolute;
  left: 220px;
}
```

Za vrednost CSS svojstva `position` postavljeno je `absolute`. Na taj način je element izbačen iz normalnog toka dokumenta i njegova pozicija se utvrđuje na osnovu `body` elementa i vrednosti offset svojstava (`top`, `right`, `bottom`, `left`). Za vrednost offset svojstva `left` postavljeno je 220 px, što znači da će rastojanje leve ivice `box1` elementa od leve ivice `body` elementa biti 220 px.

Efekat koji apsolutno pozicioniranje ima prikazan je slikom 15.3.

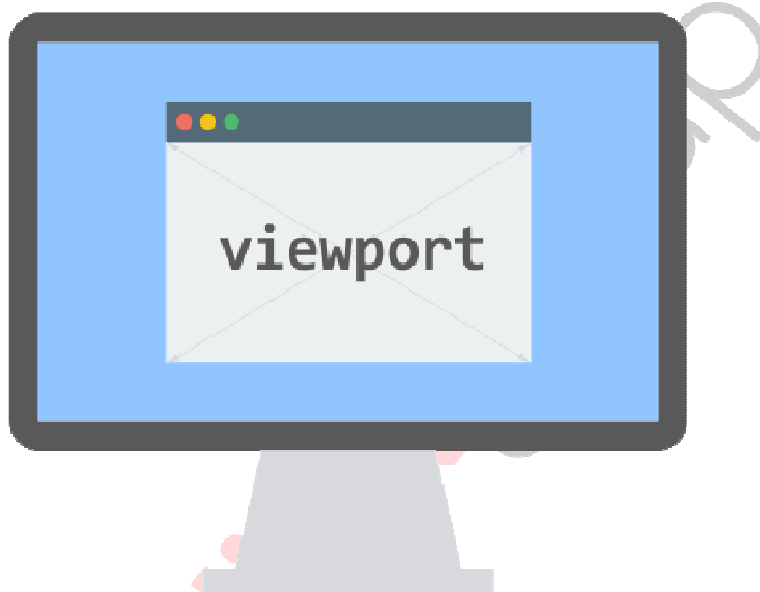


Slika 15.3. Apsolutno pozicioniran `box1` `div` element

Apsolutno pozicionirani elementi ne utiču na ostale elemente, što se može videti na slici. Elementi `box2` i `box3` ponašaju se kao da element `box1` ne postoji, te se tako element `box2` nalazi na poziciji na kojoj bi se inače nalazio element `box1`.

Fiksno pozicioniranje

Fiksno pozicioniranje omogućava da se element pozicionira relativno prikazu (engl. *viewport*) browsera. Viewport može se doživeti kao vidno polje browsera, odnosno to je onaj region unutar koga se prikazuje sadržaj HTML dokumenata. Slika 15.4. ilustruje pojam viewporta unutar jednog browsera. Bez obzira na to koliki je dokument koji browser prikazuje, viewport je uvek fiksne veličine, sve dok se ne promeni veličina prozora browsera.



Slika 15.4. Viewport browsera

Efekat fiksnog pozicioniranja biće prikazan na primeru identičnom prethodnim. HTML struktura je sledeća:

```
<div id="box1">
</div>

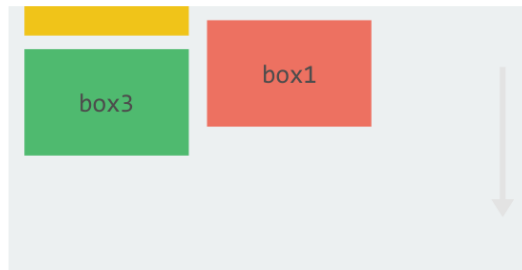
<div id="box2">
</div>

<div id="box3">
</div>
```

Div element sa id-jem `box1` biće fiksno pozicioniran na sledeći način:

```
#box1{
  position: fixed;
  left: 220px;
}
```

Koristi se identična vrednost `left` offset svojstva, kao u prethodnom primeru. Na prvi pogled fiksno pozicioniranje identično je apsolutnom, kada se apsolutno pozicioniranje odnosi na `body` element. Ipak, ukoliko se izvrši scroll stranice, postaje jasno da kod fiksnog pozicioniranja element zadržava svoju poziciju u odnosu na viewport. Takvo ponašanje ilustruje slika 15.5.



Slika 15.5. Element `box1` sa fiksnim pozicioniranjem

Pitanje

Koje je podrazumevano pozicioniranje koje HTML elementi imaju?

- a) **Static;**
- b) Relative;
- c) Absolute;
- d) Fixed.

Objašnjenje

Statičko pozicioniranje je pozicioniranje koje se kod HTML elemenata. Ovim pozicioniranjem elementi se raspoređuju po normalnom toku dokumenta.

Float svojstvo

Nakon različitih načina za pozicioniranje HTML elemenata na stranici, u ovoj lekciji biće prikazana uloga koju `float` svojstvo ima na raspoređivanje elemenata na stranici. Korišćenjem ovog svojstva moguće je izvesti element iz normalnog toka dokumenta i pozicionirati ga skroz levo ili skroz desno unutar njegovog roditeljskog kontejnera. Efekat koji `float` svojstvo ima biće prikazan na primeru sledeće HTML strukture:

```
<div id="box1">
</div>

<div id="box2">
</div>

<div id="box3">
</div>

<div id="box4">
</div>
```

```
<div id="box5">  
</div>
```

```
<div id="box6">  
</div>
```

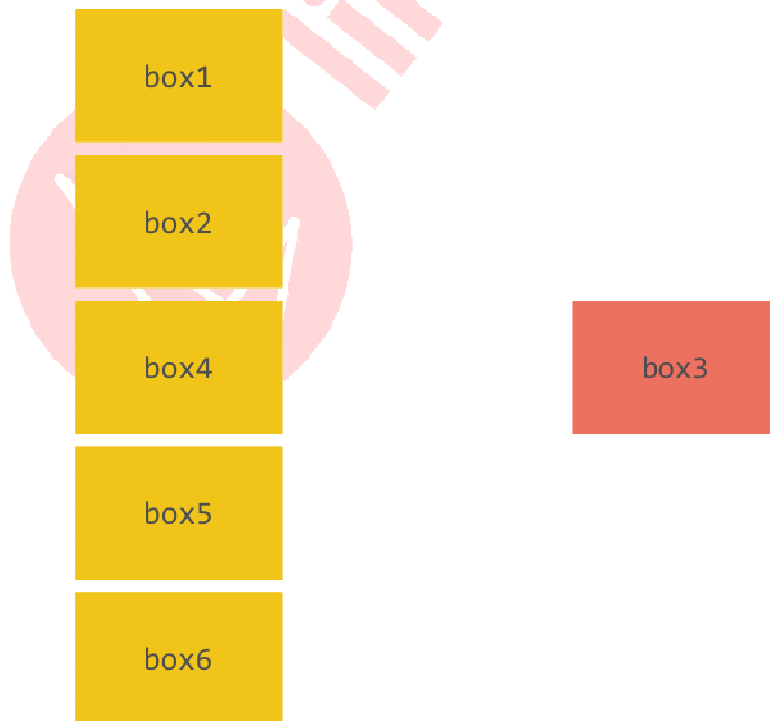
Prikazana struktura sastoji se iz 6 `div` elemenata, koji se nalaze unutar `body` dela stranice. Stilizacija koja se odnosi na sve `div` elemente je sledeća:

```
div{  
  width: 200px;  
  height: 100px;  
  margin-bottom: 20px;  
  background-color: #F0C419;  
}
```

Nad `div` elementima definisana je širina i visina, donja margina i boja pozadine. Pogledajte sada šta će se dogoditi ukoliko se na jednom od `div` elemenata definiše vrednost `float` svojstva na sledeći način:

```
#box3{  
  float: right;  
  background-color: #ED7161;  
}
```

Postavljanje `float` svojstva na `right` za element `box3` ima efekat kao na slici 15.6.



Slika 15.6. Element `box3` sa vrednošću `float` svojstva postavljenom na `right`

Na slici 15.6. jasno se može videti da postavljanjem `float` svojstva element izlazi iz prirodnog toka dokumenta. S obzirom na to da je `float` svojstvo postavljeno na `right`, element `box3` pozicionirao se skroz desno unutar svog roditeljskog kontejnera (u ovom slučaju `body` elementa), dok je prvi sledeći element zauzeo njegovo mesto.

Na kraju lekcije imate radno okruženje pa možete sve ove elemente i svojstva da iskusate u okviru njega i vidite njihov rezultat prilikom pokretanja stranice.

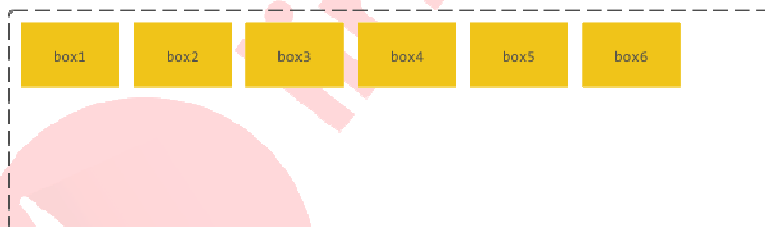
Inače `float` svojstvo može imati tri različite vrednosti:

- `none`
- `left`
- `right`

Vrednost `none` podrazumevana je i označava da element nije pomeren ni na jednu stranu. `Float left` pozicionira element krajnje levo unutar roditelja, a `float right` krajnje desno. Interesantno je da je korišćenjem `float` svojstva moguće slagati block elemente jedan pored drugog na identičan način na koji se slažu i inline-block elementi. To ilustruje sledeći primer:

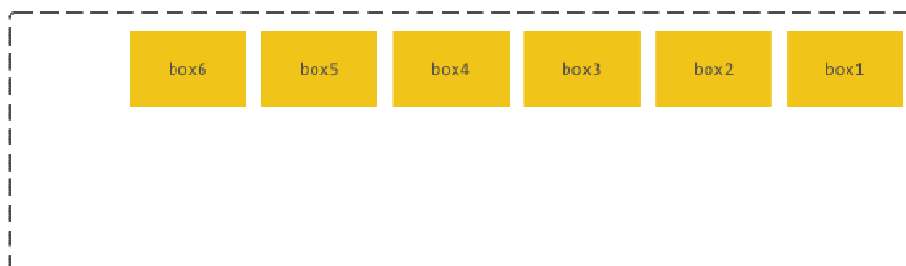
```
div{  
    margin-right: 20px;  
    float: left;  
}
```

Na ovaj način svi `div` elementi imaju vrednost `float` svojstva `left`, što će rezultovati prikazom kao na slici 15.7.



Slika 15.7. Div elementi sa float svojstvima postavljenim na left

Ukoliko bi svi `div` elementi imali vrednost `float` svojstva `right`, postigao bi se efekat kao na slici 15.8.



Slika 15.8. Div elementi sa float svojstvima postavljenim na right

Napomena

Apsolutno pozicionirani elementi ignorišu float svojstvo. Takođe, vertikalne margine floated elemenata nikada se ne stapaju sa marginama elemenata iznad ili ispod.

Clear svojstvo

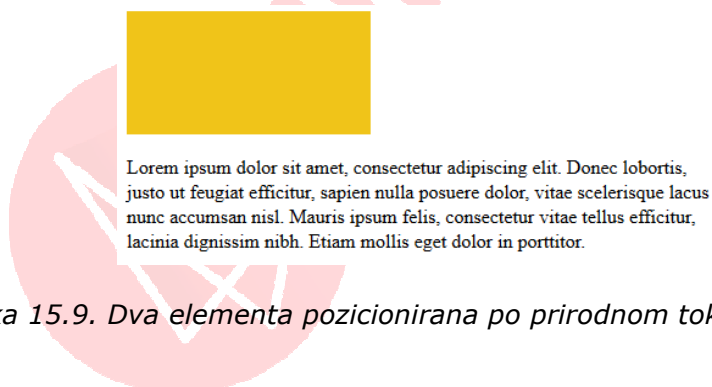
Svojstvo `clear` direktno je povezano sa upravo opisanim svojstvom `float`. Njegova svrha jeste kontrola ponašanja elemenata koji se nalaze oko elementa koji su pomereni korišćenjem svojstva `float`.

S obzirom na to da definisanjem `float` svojstva na nekom elementu on izlazi iz prirodnog toka dokumenta, ostali elementi teže da popune prostor oko takvog elementa. Tako nešto ilustrovaće primer sa sledećom HTML strukturom:

```
<div id="box1">  
</div>
```

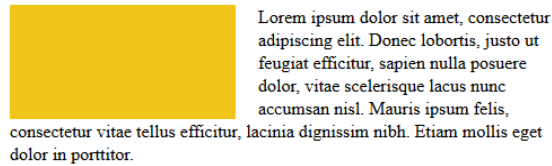
```
<p>  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis,  
justo ut feugiat efficitur, sapien nulla posuere dolor, vitae scelerisque  
lacus nunc accumsan nisl. Mauris ipsum felis, consectetur vitae tellus efficitur,  
lacinia dignissim nibh. Etiam mollis eget dolor in porttitor.  
</p>
```

Prikazana HTML struktura sastoji se iz jednog `div` elementa i jednog paragrafa. Po normalnom toku dokumenta ova dva elementa bila bi pozicionirana kao na slici 15.9.



Slika 15.9. Dva elementa pozicionirana po prirodnom toku dokumenta

Ukoliko se `div` element korišćenjem `float` svojstva pomeri na levu stranu, situacija se menja, s obzirom na to da takav element izlazi iz normalnog toka, pa je raspored elemenata kao na slici 15.10.



Slika 15.10. Div i paragraf elementi, pri čemu je div floatovan na levo

Evidentno je da sadržaj paragrafa nastoji da obmota div element sa definisanom float vrednošću. Ali šta ukoliko tako nešto mi ne želimo?

Dovoljno je iskoristiti `clear` svojstvo, kao u sledećem primeru:

```
p{
    clear: left;
}
```

U primeru je iskorišćena `left` vrednost `clear` svojstva, s obzirom na to da je potrebno *počistiti* efekat istoimenog `float` svojstva. Svojstvo `clear` može imati nekoliko različitih vrednosti i one su prikazane tabelom 15.1.

| Vrednost svojstva | Opis |
|--------------------|---|
| <code>left</code> | Ne dozvoljava pojavu float elemenata sa leve strane |
| <code>right</code> | Ne dozvoljava pojavu float elemenata sa desne strane |
| <code>both</code> | Ne dozvoljava pojavu float elemenata kako sa leve, tako i sa desne strane |

Tabela 15.1. Vrednost clear svojstva

Jedna od čestih primena svojstva `clear` jesu situacije kada kontejnerski element sadrži isključivo elemente pomerene korišćenjem `float` svojstva. U takvoj situaciji, roditeljski element **nema sopstvenu visinu**, s obzirom na to da su elementi izbačeni iz prirodnog toka dokumenta. Kada dođe do takve situacije, svaki naredni element na stranici neće uzimati u obzir realnu visinu elemenata sa `float` svojstvom, te tako dolazi do preklapanja elemenata na stranici. Sledeći primer ilustruje takav problem.

```
<div id="container1">
  <div>
  </div>

  <div>
  </div>

  <div>
  </div>
</div>

<div id="container2">
</div>
```

HTML struktura sastoji se iz dva `div` elementa. Prvi `div`, sa id-em `container1`, sadrži tri `div` elementa, dok je drugi `div`, sa id-em `container2` prazan. Prikazani elementi biće stilizovani na sledeći način:

```
#container1 div{
    width: 200px;
    height: 100px;
    margin: 20px;
    background-color: #F0C419;
}

#container2 {
    height: 100px;
    margin: 20px;
    background-color: #ED7161;
}
```

Prikazanim CSS opisima definisana je visina i širina elemenata koji se nalaze u `container1` `div` elementu, njihove margine i boja pozadine.

Za `container2` element definisane su visina, margine i boja.

Na prikazan način elementi će na stranici biti raspoređeni kao na slici 15.11.



Slika 15.11. Četiri `div` elementa raspoređena prirodnim tokom

Svi elementi učestvuju u normalnom toku dokumenta, tako da nema nikakve dileme o njihovom pozicioniranju na stranici. Ali šta bi se dogodilo ukoliko bi svi elementi `container1` `div` elementa imali `float` vrednost `left`, kao u sledećem primeru?

```
#container1 div{
    width: 200px;
    height: 100px;
    margin: 20px;
    background-color: #F0C419;
    float: left;
}
```

Na ovaj način `container1` `div` element sadrži isključivo elemente koji imaju `float` svojstvo postavljeno na `left`. Zbog toga element `container1` neće imati svoju visinu, pošto ne sadrži nijedan element raspoređen po normalnom toku dokumenta. Ovakva situacija proizvešće efekat prikazan slikom 15.12.



Slika 15.12. Kada su svi elementi unutar nekog kontejnera izvedeni iz prirodnog toka, njihov roditeljski element nema visinu, pa ga ostali elementi na stranici ignorišu

Naravno da je cilj smestiti elemente jedan ispod drugog, a da bi se tako nešto postiglo, dovoljno je iskoristiti `clear` CSS svojstvo.

```
#container2 {  
...  
    clear: left;  
}
```

Jednostavnim dodavanjem `clear` svojstva sa vrednošću `left` na element `container2` rešava se problem preklapanja elemenata, koji je izazvan činjenicom da roditeljski element sa svim pomerenim potomcima ne poseduje sopstvenu visinu. Efekat će biti kao na slici 15.13.



Slika 15.13. Upotreba `clear` svojstva za rešavanje problema elemenata koji nemaju visinu

Z-index svojstvo

U dosadašnjem toku lekcije prikazano je nekoliko pristupa pomoću kojih je moguće postići preklapanje više elemenata. Kada se kaže preklapanje, misli se na prikaz jednog elementa iznad nekog drugog, baš kao na slici 15.14.



Slika 15.14. Dva div elementa koja se preklapaju

Preklapanje se može postići korišćenjem relativnog, apsolutnog i fiksnog pozicioniranja, ali i korišćenjem `float` svojstva. Kada se dva elementa preklape, njihov vizuelni raspored na stranici određen je njihovim položajem u samom kodu. Tako se element koji je prvi naveden u kodu na stranici pojavljuje ispod elementa koji je naveden kasnije. Primer koji ilustruje slika 15.14. može se dobiti na sledeći način:

```
div{
    width: 180px;
    height: 100px;
    margin: 20px;
}

#box1{
    background-color: #F0C419;
    position: relative;
    top: 70px;
    left: 50px;
}

#box2{
    background-color: #ED7161;
    position: relative;
}

<div id="box1">
</div>

<div id="box2">
</div>
```

HTML struktura sastoji se iz dva `div` elementa. Oba `div` elementa poseduju identičnu visinu i širinu i pozicionirana su relativno. Div element `box1` pomeren je za 70 px nadole i 50 px udesno. Tako se dobija situacija već ilustrovana slikom 15.14.

S obzirom na to da je `div` element `box1` pozicioniran u kodu pre elementa `box2`, on se na stranici prikazuje vizuelno ispod elementa `box2`. Na ovako podrazumevano ponašanje može se uticati korišćenjem **z-index** svojstva:

```
#box1 {
  ...
  z-index: 1;
}

#box2{
  ...
  z-index: 0;
}
```

Na ovaj način postiže se prikaz kao na slici 15.15.



Slika 15.15. Z-index svojstvom utiče se na redosled elemenata na stranici

Svojstvo `z-index` može da prihvati bilo koji pozitivni ili negativni ceo broj. Prilikom utvrđivanja redosleda kojim će elementi biti pozicionirani, u obzir se uzima vrednost `z-index` svojstva. Element koji poseduje višu vrednost ovoga svojstva prikazuje se iznad elemenata sa nižom vrednošću.

Napomena

Veoma je bitno napomenuti da svojstvo `z-index` ima efekta samo na pozicioniranim elementima. Nešto ranije je rečeno da se pod pojmom pozicioniranih elemenata smatraju oni koji su apsolutno, relativno ili fiksno pozicionirani.

Visibility svojstvo

U jednoj od prethodnih lekcija u kojoj je bilo reči o `display` CSS svojstvu demonstriran je i efekat `none` vrednosti takvog svojstva. Mogli ste da vidite da element sa `none` vrednošću `display` svojstva nestaje sa stranice i da drugi elementi zauzimaju njegovo mesto. Korišćenjem CSS-a moguće je postići i nešto slično. Naime, moguće je element sakriti sa stranice, ali njegov prostor ostaviti rezervisanim tako da drugi elementi ne mogu da ga zauzmu. Ovo se postiže korišćenjem svojstva `visibility`.

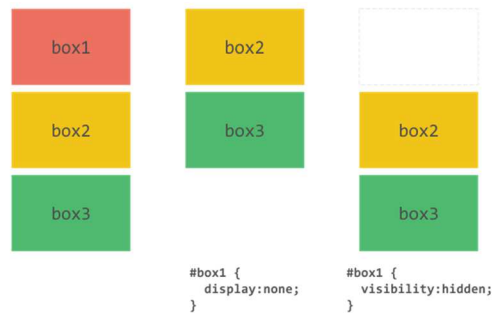
Svojstvo `visibility` može imati dve vrednosti:

- `visible;`
- `hidden.`

Podrazumevana vrednost je `visible`, što znači da će element biti vidljiv na stranici unutar browsera. Postavljanjem vrednosti `hidden` element se skriva:

```
span {
  visibility:hidden;
}
```

Slika 15.16. ilustruje razlike između `display: none` i `visibility: hidden` vrednosti.



Slika 15.16. Razlike između `display: none` i `visibility: hidden` vrednosti

Overflow svojstvo

Ponekad se može dogoditi da sadržaj prevazilazi veličinu svog roditeljskog elementa. U takvim situacijama CSS omogućava da se definiše šta će se dogoditi sa sadržajem koji izlazi iz okvira svog roditelja korišćenjem svojstva `overflow`. Tri najznačajnije vrednosti `overflow` svojstva prikazane su tabelom 15.2.

| Vrednost | Opis |
|----------|---|
| visible | Sadržaj koji izlazi iz okvira roditelja je vidljiv; ovo je podrazumevana vrednost |
| hidden | Sadržaj koji izlazi iz okvira roditelja nije vidljiv |
| scroll | Sadržaj koji izlazi iz okvira roditelja nije vidljiv, ali se prikazuju trake za skrolovanje kako bi se mogao videti sadržaj koji izlazi iz okvira roditelja |

Tabela 15.2. Vrednosti `overflow` svojstva

Sledeći primer ilustruje upotrebu svojstva `overflow`. HTML kod je sledeći:

```
<style>

#container1{
    width: 200px;
    height: 300px;
    margin: 20px;
    background-color: #F0C419;
    display: inline-block;
}

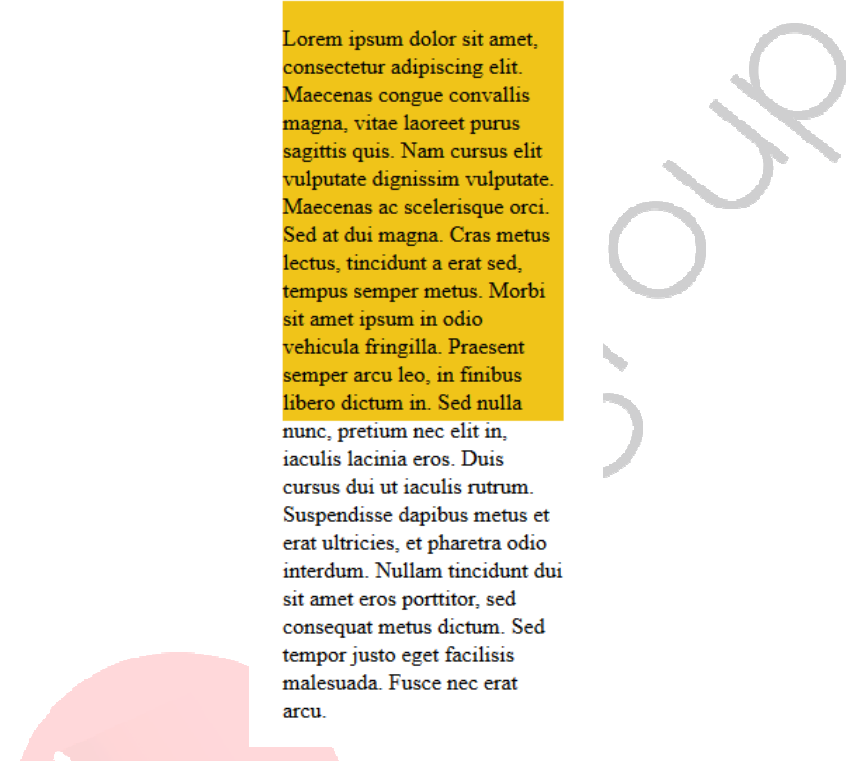
</style>

<div id="container1">
    <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas
        congue convallis magna, vitae laoreet purus sagittis quis. Nam cursus elit
        vulputate dignissim vulputate. Maecenas ac scelerisque orci. Sed at dui
        magna. Cras metus lectus, tincidunt a erat sed, tempus semper metus. Morbi
        sit amet ipsum in odio vehicula fringilla. Praesent semper arcu leo, in
```

finibus libero dictum in. Sed nulla nunc, pretium nec elit in, iaculis lacinia eros. Duis cursus dui ut iaculis rutrum. Suspendisse dapibus metus et erat ultricies, et pharetra odio interdum. Nullam tincidunt dui sit amet eros porttitor, sed consequat metus dictum. Sed tempor justo eget facilisis malesuada. Fusce nec erat arcu.

</p>
</div>

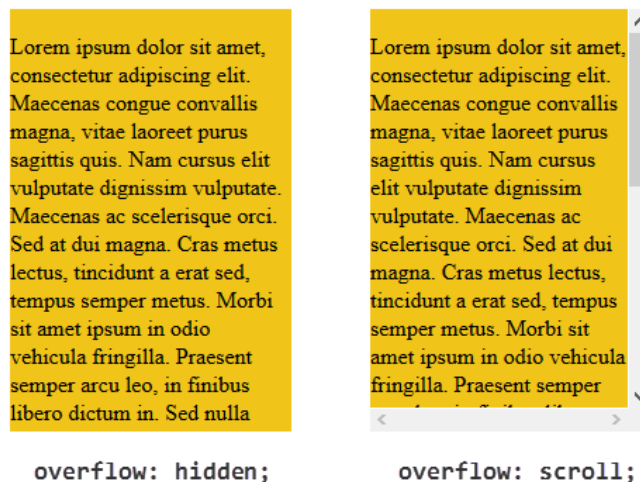
Navedeni kod proizvešće efekat kao na slici 15.17.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas congue convallis magna, vitae laoreet purus sagittis quis. Nam cursus elit vulputate dignissim vulputate. Maecenas ac scelerisque orci. Sed at dui magna. Cras metus lectus, tincidunt a erat sed, tempus semper metus. Morbi sit amet ipsum in odio vehicula fringilla. Praesent semper arcu leo, in finibus libero dictum in. Sed nulla nunc, pretium nec elit in, iaculis lacinia eros. Duis cursus dui ut iaculis rutrum. Suspendisse dapibus metus et erat ultricies, et pharetra odio interdum. Nullam tincidunt dui sit amet eros porttitor, sed consequat metus dictum. Sed tempor justo eget facilisis malesuada. Fusce nec erat arcu.

Slika 15.17. Sadržaj div elementa prevazilazi njegove okvire

Na slici 15.17 jasno se vidi da sadržaj `div` elementa prevazilazi njegove okvire. S obzirom na to da je podrazumevana vrednost svojstva `overflow visible`, sadržaj koji se nalazi izvan roditeljskog okvira jeste vidljiv. Ipak, ukoliko se navedu neke druge vrednosti ovoga svojstva, postižu se drugačiji efekti, prikazani slikom 15.18.



Slika 15.18. Različiti efekti koji se postižu upotrebom overflow svojstva

Ukoliko se za vrednost svojstva `overflow` postavi `hidden`, sadržaj koji prevazilazi okvire svog roditelja neće bit vidljiv. Takav efekat prikazan je na levoj polovini slike 15.18. Ukoliko je potrebno korisniku ipak omogućiti pregled sadržaja koji prevazilazi okvire roditeljskog elementa, moguće je za vrednost svojstva `overflow` postaviti `scroll`. U takvoj situaciji, na elementu čiji je sadržaj isečen pojavljuje se jedan ili se pojavljuju dva scrollbara. Na slici 15.18. ovakva situacija prikazana je na desnoj strani, gde se može videti `div` element sa vertikalnim i horizontalnim trakama za skrolovanje, od kojih je aktivna vertikalna traka.

Radno okruženje

Kao što je prethodno navedeno, sada možete da iskucate obrađene primere u lekciji i na taj način da ih proverite kako se ponašaju na stranici. Slobodno zamenite neke delove koda, npr. vrednosti i svojstva i vidite kakav će ishod biti nakon toga.

Primer: Kreiranje layouta korišćenjem pristupa iz ove lekcije

U prethodnoj lekciji ilustrovani su prvi primeri kreiranja layouta. Nakon različitih CSS pojmova iz ove lekcije, u stanju smo da layout iz prethodne lekcije unapredimo. Stoga će u nastavku biti prikazano kreiranje identičnog layouta onom iz prethodne lekcije, ali bez upotrebe inline-block elemenata. Stoga će struktura tela HTML dokumenta biti identična onoj iz prethodne lekcije:

Radno okruženje

HTML fajl

```
<header>
  
</header>
```

```

<nav>
  <a href="#">Home</a>
  <a href="#">About Us</a>
  <a href="#">Contact</a>
</nav>

<h1>Home</h1>

<section>
  <h2>Main content</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum vehicula quam eros, eget condimentum augue dictum sed. Vivamus blandit purus nibh, sit amet molestie felis sollicitudin id. Morbi tortor enim, varius eget gravida eu, dictum in velit. Suspendisse sit amet metus aliquet, viverra est id, efficitur diam. Integer laoreet nisi arcu, a venenatis nisl convallis a. Cras malesuada lobortis ex. Praesent vel massa a eros dignissim commodo. Ut justo purus, tincidunt ut ultrices ac, ultrices eget tellus. Cras sodales libero sed velit aliquet condimentum.</p>

  </section><aside>
    <h2>Side content</h2>
    <p>Pellentesque sem dolor, tempus at felis quis, dignissim lacinia enim. Curabitur non efficitur eros. Praesent pretium neque diam, non sollicitudin purus vulputate sed. Vestibulum posuere tincidunt commodo. Aliquam nec neque feugiat, gravida mi quis, pellentesque lacus. Cras mollis odio ac dignissim tincidunt. Phasellus sed blandit eros. Donec malesuada lectus vel massa ultricies, sed rhoncus nulla vestibulum. Mauris vulputate ac ipsum at elementum. Proin aliquam sit amet justo dapibus cursus. Donec id dignissim arcu. Sed sed justo in lectus efficitur maximus.</p>
  </aside>

<footer>
  Copyright © Link Group
</footer>

```

CSS fajl

```

body {
  width: 960px;
  margin: 20px auto;
}

header {
  border-bottom: 1px solid gray;
  padding: 8px 0 8px 0;
}

nav {
  padding: 16px 0 16px 0;
  border-bottom: 1px solid gray;
}

a {
  margin-right: 6px;
}

```

```
}

section {
  float: left;
  width: 600px;
  padding: 0 8px 0 0;
  border-right: 1px solid gray;
  margin-bottom: 8px;
  box-sizing: border-box;
}

aside {
  float: left;
  width: 360px;
  padding: 0 0 0 8px;
  box-sizing: border-box;
}

footer {
  padding: 16px 0 16px 0;
  border-top: 1px solid gray;
  clear: left;
}
```

Veći deo prikazanog CSS-a identičan je onom iz prethodne lekcije. Ipak, postoji nekoliko bitnih razlika. Prva je ta da `section` i `aside` elementi više nisu `inline-block`. Sada su oni klasični `block` elementi (zato što su semantički elementi podrazumevno `block` elementi). Ipak, da bi se dobio prikaz sadržaja u dve kolone, ova dva elementa sada su pomerena iz normalnog toka dokumenta korišćenjem `float` svojstva, koje je za oba elementa postavljeno na `left`. Kako ne bi došlo do problema sa prikazom `footer` elementa, koji u ovakvoj situaciji sledi elementima koji su pomereni korišćenjem `float` svojstva, na njemu je postavljeno `clear` svojstvo, sa vrednošću `left`.

Pokušajte i sami unutar radnog okruženja da provežbate nekoliko primera layouta, gde ćete koristiti svojstva koja su obrađena u ovoj lekciji. Na taj način ćete kreirati osnovni raspored elemenata, koji je osnova svakog veb sajta.

Na kraju, prikazani CSS poseduje još jedno unapređenje. Možete videti da su širine `section` i `aside` elemenata postavljene na 600 i 360 piksela, respektivno. Iako oba elementa poseduju i unutrašnje razmake (po 8 piksela na desnoj, odnosno levoj ivici), a `section` element i okvir od jednog piksela sa desne strane, prikaz je identičan kao u prethodnoj lekciji. Zbog čega smo sada u mogućnosti da dostupan prostor direktno podelimo između dva elementa, bez uzimanja u obzir njihovih eventualnih unutrašnjih razmaka i okvira? Odgovor na ovo pitanje leži u upotrebi **box-sizing** svojstva.

Svojstvo box-sizing

CSS svojstvo `box-sizing` omogućava odabir načina na koji će biti računate dimenzije HTML elementa. Tako je korišćenjem ovog svojstva moguće definisati da li će unutrašnji razmaci i okviri ući u ukupnu visinu i širinu elementa ili ne.

Svojstvo `box-sizing` može imati sledeće vrednosti:

- `content-box;`
- `border-box.`

Vrednost `content-box` je podrazumevana i definiše da će visina i širina obuhvatati samo sadržaj elementa. Sa druge strane, vrednost `border-box` u kalkulaciju visine i širine uključuje i unutrašnje razmake i okvire. Kao što ste mogli videti iz upravo prikazanog primera, vrednost `border-box` je i više nego korisna u pojedinim situacijama, jer nas lišava potrebe nepotrebnog dodatnog računanja prilikom raspodele dostupnog prostora između više elemenata.

Rezime

- Svaki HTML element može da bude pozicioniran statično, relativno, apsolutno ili fiksno.
- Statičko pozicioniranje je pozicioniranje koje se podrazumeva kod HTML elemenata i kojim se elementi raspoređuju po normalnom toku dokumenta.
- Element sa relativnim pozicioniranjem pozicioniran je relativno svojoj normalnoj poziciji koju bi imao da je pozicioniran kao statički element.
- Element je apsolutno pozicioniran onda kada je pozicioniran relativno najbližem pretku koji je pozicioniran na bilo koji način, osim korišćenjem `static` pozicioniranja.
- Fiksno pozicioniranje omogućava da se element pozicionira relativno prikazu (viewportu) browsera.
- Korišćenjem svojstva `float` moguće je izvesti element iz normalnog toka dokumenta i pozicionirati ga skroz levo ili skroz desno unutar njegovog roditeljskog kontejnera.
- Svojstvom `clear` kontroliše se ponašanje elemenata koji se nalaze oko elemenata koji su pomereni korišćenjem svojstva `float`.
- Kada se dva elementa preklape, onaj element koji se nalazi kasnije u kodu pojaviće se vizuelno iznad elementa koji je ranije u kodu; na ovo ponašanje može se uticati korišćenjem `z-index` svojstva.
- Svojstvo `visibility` omogućava da se određeni HTML element sakrije, ali da pritom prostor koji on zauzima ostane rezervisan.
- CSS svojstvo `overflow` omogućava da se definiše šta će se dogoditi sa sadržajem koji izlazi iz okvira svog roditelja.
- CSS svojstvo `box-sizing` omogućava odabir načina na koji će biti računate dimenzije HTML elementa.