**Kreiranje layouta**

**00:37:01**

Jedinica: 15 od 30

**+Rezime**

Određeni HTML elementi imaju zaduženje da služe kao omotači za druge HTML elemente. Oni uglavnom nemaju vizuelnu reprezentaciju na stranici, a koriste se kako bi se postigao željeni raspored HTML elemenata. Takvi elementi obrađeni su u lekciji *Grupisanje elemenata* u prvom modulu ovog kursa. Ipak, u pomenutoj lekciji, praktično, nije mnogo toga moglo biti viđeno, zato što pomenuti specijalni elementi za kreiranje layouta svoju punu moć iskazuju u kombinaciji sa CSS stilizacijom. Stoga će u ovoj lekciji biti prikazano kako se CSS može koristiti za raspoređivanje elementa na stranici, te za kreiranje željenog layouta.

**HTML elementi za kreiranje layouta**

Odmah na početku, dobro je podsetiti se elemenata opisanih u lekciji *Grupisanje elemenata*. Tabela 15.1 prikazuje najkorišćenije elemente za kreiranje layouta.

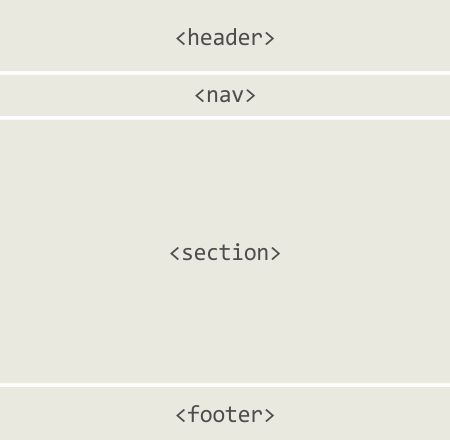
|  |  |
| --- | --- |
| **HTML element** | **Opis** |
| <div> | opšti element za grupisanje |
| <article> | nezavisni članak unutar dokumenta |
| <aside> | sadržaj pored glavnog sadržaja – sidebar |
| <figure> | sadržaj koji sadrži ilustraciju, sliku ili grafikon |
| <figcaption> | naslov za figure element |
| <footer> | dno (podnožje) stranice ili neke od njenih sekcija |
| <header> | zaglavlje stranice ili neke od njenih sekcija |
| <nav> | kontejner za navigacione linkove |
| <section> | sekcija dokumenta |

*Tabela 15.1 – Najkorišćeniji elementi za kreiranje layouta*

Sledeći primer ilustruje HTML strukturu jednog dokumenta sa definisanim različitim regionima korišćenjem HTML elemenata za grupisanje:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <body>     <header>   <h1>Site main heading</h1>   </header>     <nav>   Home<br>   About Us<br>   Contact   </nav>     <section>   <h1>Home</h1>   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed eget dolor eget libero aliquam hendrerit. Aliquam hendrerit condimentum blandit. Maecenas facilisis arcu vitae sem eleifend, eu volutpat libero rutrum. Morbi rhoncus mauris eget laoreet placerat. Fusce vel ex sit amet justo blandit rutrum.</p>   <p>Integer ultricies, sem eget volutpat commodo, massa justo iaculis sem, sit amet tempor dui lorem tristique elit. Phasellus nec ornare augue, quis scelerisque diam. Curabitur accumsan, lectus quis accumsan dictum, ligula sem fringilla enim, a rhoncus justo urna eu odio.</p>   </section>     <footer>   Copyright © Link Group   </footer>    </body> |

Prikazanim HTML kodom, zamišljeno je stvaranje strukture kao na slici 15.1.



*Slika 15.1 – Struktura jedne HTML stranice*

Ukoliko prikazani HTML kod otvorite u nekom od pretraživača, primetićete da raspored komponenata nije onakav kakav očekujemo. Razlog je više nego očigledan. Prikazani HTML elementi sami po sebi nemaju nikakvu moć raspoređivanja, već sadržaju daju samo semantičko značenje. Kako bi se uticalo na layout elementa, potrebno je koristiti CSS.

**Tehnike za raspoređivanje elemenata**

Za raspoređivanje elemenata na stranici postoji nekoliko pristupa koje je moguće koristiti. Ti pristupi su sledeći:

* HTML tabele
* CSS
* CSS framework
* CSS Flexbox

Često na internetu možete sresti pristup po kome se HTML tabele koriste za raspoređivanje elemenata. HTML table element namenjen je kreiranju tabela u HTML dokumentu, a nikako raspoređivanju HTML elemenata. Drugim rečima, HTML table element nije element za kreiranje layouta i savetuje se da se on u te svrhe ne zloupotrebljava.

Raspoređivanje komponenata se vrlo lako i brzo može obaviti korišćenjem nekog od frameworka koji su za to namenjeni. Dva najpopularnija frameworka te vrste su [Bootstrap](https://www.link-elearning.com/linkdl/opisPojma.php?id=144305) i [Foundation](https://www.link-elearning.com/linkdl/opisPojma.php?id=144306). Ipak, naš cilj u ovom kursu jeste da naučimo osnove CSS-a, tako da korišćenje različitih frameworka neće biti uzeto u razmatranje u ovoj lekciji. Na kraju, i pomenuti okviri za kreiranje layouta koriste principe koji će biti predstavljeni u nastavku ove lekcije.

CSS Flexbox je pristup koji je predstavljen u CSS3 verziji jezika i predstavlja veoma lak način za raspoređivanje elemenata. Ipak, ovakav pristup nije podržan u browserima starijim od IE10. O ovom pristupu će biti više reči kasnije.

U nastavku lekcije biće objašnjene osnovne tehnike za raspoređivanje HTML elemenata korišćenjem čistog CSS-a.

**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 (default)
* relative
* absolute
* fixed

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

Inicijalno, svi elementi unutar HTML stranice raspoređuju se po pravilima prirodnog toka (*normal flow*). To praktično znači da se block elementi raspoređuju jedan ispod drugog pri čemu zauzimaju kompletnu dostupnu širinu, a da se inline elementi ređaju jedan pored drugog u horizontalnoj liniji. Pozicioniranja static i relative zadržavaju elemente u prirodnom toku, dok pozicioniranja absolute i fixed, kao i korišćenje float svojstva, izbacuju element iz prirodnog toka stranice.

U nastavku će detaljno biti objašnjena različita pozicioniranja.

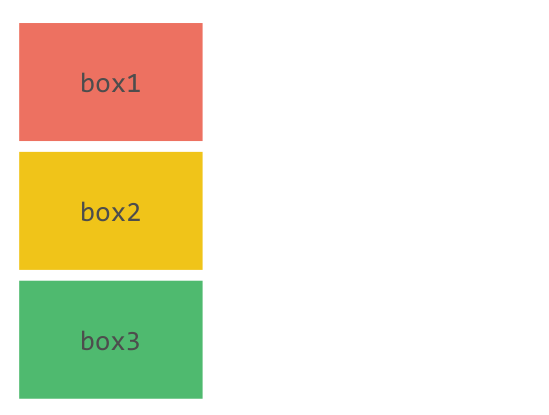
**Statičko pozicioniranje**

Statičko pozicioniranje je pozicioniranje koje HTML elementi imaju podrazumevano. Ovim pozicioniranjem elementi se 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <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 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 15.2.



*Slika 15.2 – Tri div elementa raspoređena po pravilima normalnog toka*

**Relativno pozicioniranje**

Element sa relativnim pozicioniranjem je pozicioniran 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 se drugačije nazivaju *Offset* svojstva.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <div id="box1">  </div>    <div id="box2">  </div>    <div id="box3">  </div> |

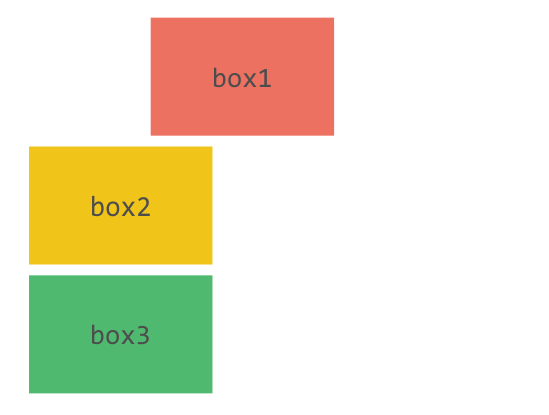
CSS stilizacija koja se odnosi na sva tri navedena div elementa je sledeća:

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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:

|  |  |
| --- | --- |
| 1  2  3  4 | #box1{   position: relative;   left: 120px;  } |

Na ovaj način, div element sa id-jem 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 15.3 to ilustruje.



*Slika 15.3 – 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 se utvrđuje na osnovu elementa body.

Apsolutno pozicioniranje je jedno 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:

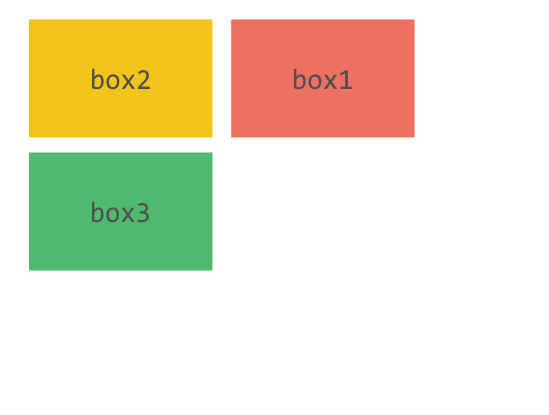
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <div id="box1">  </div>    <div id="box2">  </div>    <div id="box3">  </div> |
| 1 |  |

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

|  |  |
| --- | --- |
| 1  2  3  4 | #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 na slici 15.4.



*Slika 15.4 – Apsolutno pozicioniran box1 div element*

Apsolutno pozicionirani elementi ne utiču na ostale elemente, što se može videti na slici 15.4. Elementi box2 i box3 se ponašaju kao da element box1 ne postoji, te se tako element box2nalazi na poziciji na kojoj bi se inače nalazio element box1.

**Fiksno pozicioniranje**

Fiksno pozicioniranje omogućava da se element pozicionira relativno u odnosu na prikaz (viewport) browsera. Viewport se može doživeti kao vidno polje browsera, odnosno to je onaj region unutar koga se prikazuje sadržaj HTML dokumenata. Slika 15.5 ilustruje pojam viewporta unutar jednog browsera. Bez obzira koliki je dokument koji browser prikazuje, viewport je uvek fiksne veličine, sve dok se ne promeni veličina prozora browsera.

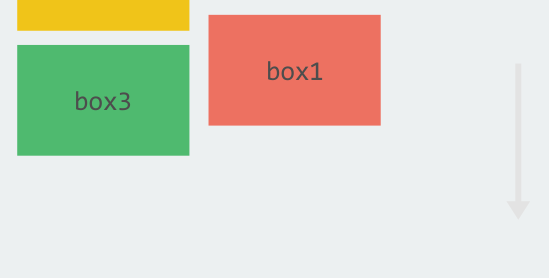


*Slika 15.5 – Viewport browsera*

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <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 je identično 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 15.6.



*Slika 15.6 – Element box1 sa fiksnim pozicioniranjem*

**Koje je podrazumevano pozicioniranje koje HTML elementi imaju?**

 Static

 Relative

 Absolute

 Fixed

**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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <div id="box1">  </div>    <div id="box2">  </div>    <div id="box3">  </div>    <div id="box4">  </div>    <div id="box5">  </div>    <div id="box6">  </div> |

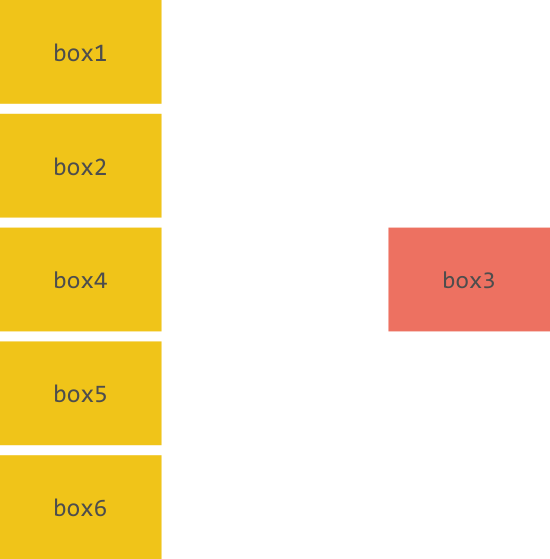
Prikazana struktura sastoji se iz 6 div elemenata koji se nalaze unutar body dela stranice. Stilizacija koja se odnosi na sve div elemente je sledeća:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | div{   width: 200px;   height: 100px;   margin-bottom: 20px;   background-color:#F0C419;  } |

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 postavi float svojstvo na right, na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4 | #box3{   float: right;   background-color: #ED7161;  } |

Postavljanje float svojstva na right za element box3 ima efekat kao na slici 15.7.



*Slika 15.7 – Element box3 sa vrednošću float svojstva postavljenom na right*

Sa slike 15.7 se jasno može videti da postavljanjem float svojstva, element izlazi iz prirodnog toka dokumenta. S obzirom da je float svojstvo postavljeno na right, element box3 se pozicionirao skroz desno unutar svog roditeljskog kontejnera (u ovom slučaju body elementa), dok je prvi sledeći element zauzeo njegovo mesto.

Inače float svojstvo može imati tri različite vrednosti:

* none
* left
* right

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

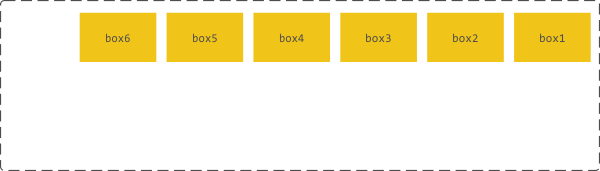
|  |  |
| --- | --- |
| 1  2  3  4 | div{   margin-right: 20px;   float: left;  } |

Na ovaj način, svi div elementi će imati vrednost float svojstva left, što će rezultovati prikazom kao na slici 15.8.



*Slika 15.8 – 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 15.9.



*Slika 15.9 – 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.* |

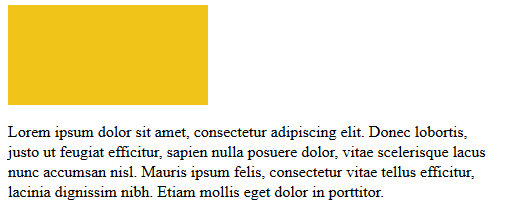
**Clear svojstvo**

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

S obzirom da definisanjem floatsvojstva 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:

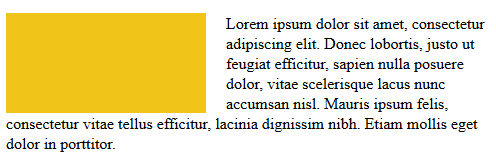
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <div id="box1">  </div>    <p>  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis, justo ut feugiat efficitur, sapien nulla posuere dolor, vitae scelerisque lacus nunc accumsan nisl. Mauris ipsum felis, consectetur vitae tellus efficitur, 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 bivaju pozicionirana na stranici kao na slici 15.10.



*Slika 15.10 – Dva elementa pozicionirana po prirodnom toku dokumenta*

Ukoliko se div element floatuje na levu stranu, situacija se menja, s obzirom da takav element izlazi iz normalnog toka, pa je raspored elemenata kao na slici 15.11.



*Slika 15.11 – Div i paragraf elementi, pri čemu je div floatovan nalevo*

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:

|  |  |
| --- | --- |
| 1  2  3 | p{   clear: left;  } |

U primeru je iskorišćena left vrednost clear svojstva, s obzirom da je potrebno *počistiti* efekat istoimenog float svojstva. Svojstvo clear može imati nekoliko različitih vrednosti i one su prikazane tabelom 15.2.

|  |  |
| --- | --- |
| **Vrednost svojstva** | **Opis** |
| left | ne dozvoljava pojavu float elemenata sa leve strane |
| right | ne dozvoljava pojavu float elemenata sa desne strane |
| both | ne dozvoljava pojavu float elemenata kako sa leve, tako i sa desne strane |

*Tabela 15.2 – Vrednost clear svojstva*

Jedna od čestih primena svojstva clear jeste u situacijama kada kontejnerski element sadrži isključivo floatovane elemente. U takvoj situaciji, roditeljski element nema sopstvenu visinu, s obzirom da su floatovani elementi izbačeni iz prirodnog toka dokumenta. Kada dođe do takve situacije, svaki naredni element na stranici neće uzimati u obzir realnu visinu floatovanih elemenata, te tako dolazi do preklapanja elemenata na stranici. Sledeći primer ilustruje takav problem.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <div id="container1">   <div>   </div>     <div>   </div>     <div>   </div>  </div>      <div id="container2">  </div> |

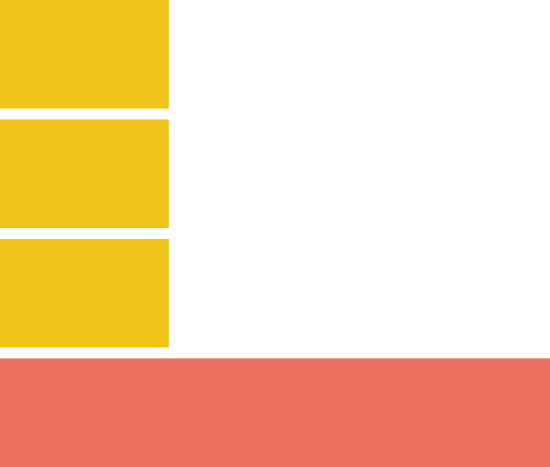
HTML struktura se sastoji iz dva kontejnerska div elementa. Prvi div, sa id-jem *container1*, sadrži tri div elementa, dok je drugi div, sa id-jem *container2*prazan. Prikazani elementi biće stilizovani na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #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 definisana je visina, margine i boja.

Na prikazan način, elementi će na stranici biti raspoređeni kao na slici 15.12.



*Slika 15.12 – 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?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #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. Na taj način, element *container1* neće imati svoju visinu, pošto ne sadrži nijedan element koji nije floatovan. Ovakva situacija proizvešće efekat prikazan slikom 15.13.

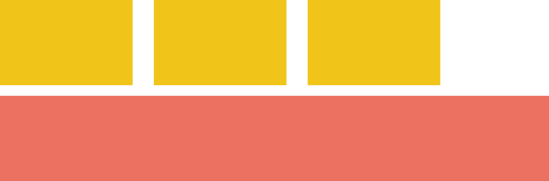


*Slika 15.13 – Kada su svi elementi unutar nekog kontejnera izvedeni iz prirodnog toka, njihov roditeljski element nema visinu, pa ga ostali elementi na stranici ignorišu*

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

|  |  |
| --- | --- |
| 1  2  3  4 | #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 floatovanim potomcima ne poseduje sopstvenu visinu. Efekat će biti kao na slici 15.14.



*Slika 15.14 – Upotreba clear svojstva za rešavanje problema elemenata koji nemaju visinu*

**Z-index svojstvo**

U dosadašnjem toku lekcije, videli ste da je korišćenjem relativnog, apsolutnog i fiksnog pozicioniranja, ali i korišćenjem float svojstva moguće postići preklapanje HTML elemenata na stranici. Kada se dva element preklope, onaj element koji se nalazi kasnije u kodu pojaviće se vizuelno iznad elementa koji je ranije u kodu.

Ipak, na ovako podrazumevano ponašanje može se uticati korišćenjem z-index svojstva. Sledeći primer ilustrovaće situaciju u kojoj se dva elementa preklapaju:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | 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 naniže i 50 px udesno. Na taj način stvoriće se struktura kao na slici 15.15.



*Slika 15.15 – Dva div elementa koja se preklapaju*

S obzirom da je div element *box1* pozicioniran u kodu pre elementa *box2*, on se na stranici renderuje vizuelno ispod elementa *box2*. To je jednostavno moguće promeniti korišćenjem z-index svojstva, na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #box1 {  ...      z-index: 1;  }    #box2{  ...   z-index: 0;  } |

Na ovaj način postiže se prikaz kao na slici 15.16.



*Slika 15.16 – Z-index svojstvom se utiče na redosled elemenata na stranici*

Svojstvo z-index može da prihvati bilo koji pozitivni ili negativni ceo broj. Vrednosti se definišu proizvoljno. Prilikom redosleda kojim će elementi biti pozicionirani, uzima se u obzir koji je broj veći, pa se na osnovu toga i formira prikaz.

|  |
| --- |
| **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.* |

**Display svojstvo**

U dosadašnjem toku kursa, već nekoliko puta pominjano je display svojstvo, i to najviše kada je bilo reči o inline i block elementima. Tada je rečeno da se svi HTML elementi mogu generalno podeliti na block i inline, i da su block i inline vrednosti svojstva display. Pored ove dve vrednosti, display svojstvo može imati i neke druge vrednosti, kojima se definiše tip *kutije* u kojoj se nalazi HTML element. Pomenućemo samo najznačajnije vrednosti koje display svojstvo može imati:

* inline
* block
* none
* inline-block
* flex

U praksi, veoma česta je upotreba **none** vrednosti display svojstva. Korišćenjem ove vrednosti, element neće biti vidljiv na stranici, iako će ostati u HTML strukturi stranice. Ova tehnika se veoma često koristi za manipulaciju vidljivošću elementa, korišćenjem jezika JavaScript.

Takođe, u praktičnom radu, veoma korisna vrednost display svojstva je i vrednost **inline-block**. Nešto ranije u ovoj lekciji, prikazano je kako se float svojstvo može koristiti za horizontalno raspoređivanje više block elemenata. Mana takvog pristupa je neophodnost korišćenja clearsvojstva na prvom sledećem elementu. Identičan efekat upravo opisanom može se mnogo lakše postići korišćenjem inline-block vrednosti. Sledeći primer će to ilustrovati.

HTML struktura je sledeća:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <div id="container1">   <div>   </div>     <div>   </div>     <div>   </div>  </div>      <div id="container2">  </div>  Prikazani elementi stilizovani su na sledeći način:  #container1 div{   width: 200px;   height: 100px;   margin: 20px;   background-color: #F0C419;   display: inline-block;  }      #container2 {   height: 100px;   margin: 20px;   background-color: #ED7161;  } |

U prikazanim CSS opisima može sa primetiti display svojstvo sa vrednošću inline-block. Na ovaj način se vrlo elegantno postiže identičan efekat koji je nešto ranije bio postignut korišćenjem float i clear svojstava. Efekat je identičan kao na slici 15.14.

Na kraju, vrednost **flex**, display svojstva ukazuje na to da se radi o elementu koji predstavlja kontejner za fleksibilne elemente. Reč je o *Flexbox Layout* načinu raspoređivanja elemenata koji je predstavljen u CSS3 verziji jezika. To je način raspoređivanja elemenata koji je razvijen kako bi odgovorio na potrebu kreiranja kompleksnih UI okruženja modernih aplikacija. Ovom načinu raspoređivanja elemenata biće posvećena cela lekcija pod naslovom *Flexbox Layout* u nastavku ovog kursa.

**Visibility svojstvo**

Svojstvo visibility omogućava da se određeni HTML element sakrije, ali da pritom prostor koji on zauzima ostane rezervisan. Stoga ovo svojstvo može imati dve vrednosti visible i hidden. Podrazumevana vrednost je visible. Sledeći primer ilustruje upotrebu ovog svojstva:

|  |  |
| --- | --- |
| 1  2  3 | span {     visibility:hidden;  } |

Možda ste primetili određenu sličnost između visibility hidden i display none vrednosti. Na oba načina se postiže skrivanje elementa sa stranice, s tim što korišćenjem display nonevrednosti, prostor koji je rezervisan za element zauzimaju ostali elementi. Tako nešto nije slučaj kada se koristi visibility hidden vrednost.

**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 15.3.

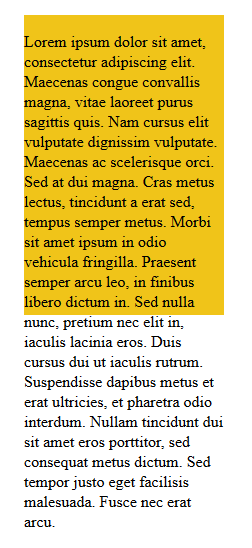
|  |  |
| --- | --- |
| **Vrednost** | **Opis** |
| visible | sadržaj koji izlazi iz okvira roditelja je vidljiv; ovo je podrazumevana vrednost |
| hidden | sadržaj koji izlazi iz okvira roditelja nije vidljiv |
| scroll | sadržaj koji izlazi iz okvira roditelja nije vidljiv, ali se prikazuju trake za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

*Tabela 15.3 – Vrednosti overflow svojstva*

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

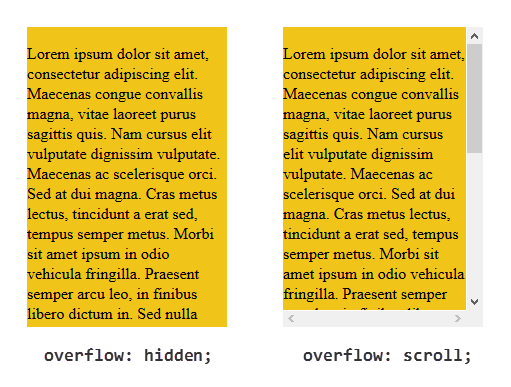
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <style>    #container1{   width: 200px;   height: 300px;   margin: 20px;   background-color: #F0C419;   display: inline-block;  }    </style>    <div id="container1">   <p>   Lorem ipsum dolor sit amet, consectetur adipiscing 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.



*Slika 15.17 – Sadržaj div elementa prevazilazi njegove okvire*

Na slici 15.17 se jasno vidi da sadržaj div elementa prevazilazi njegove okvire. S obzirom da je podrazumevana vrednost svojstva overflow visible, sadržaj koji se nalazi izvan roditeljskog okvira je 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 se pojavljuju jedan ili dva scrollbara. Na slici 15.18 je ovakva situacija prikazana na desnoj strani, gde se može videti div element sa vertikalnim i horizontalnim trakama za skrolovanje od kojih je aktivna vertikalna traka.

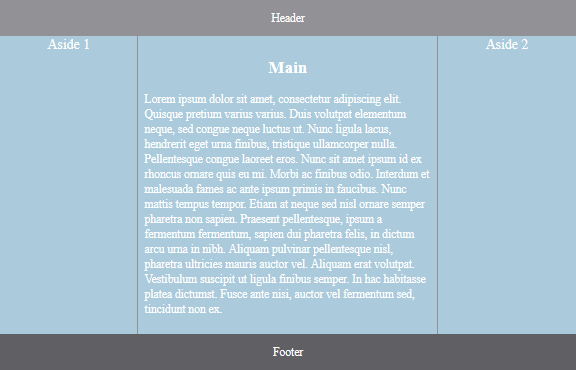
***Primer***

***Kreiranje layouta korišćenjem osnovnih CSS pristupa***

*U nastavku biće prikazan primer standardnog layouta web sajta koji se sastoji iz tri kolone, headera i footera. U prilogu lekcije se nalazi izvorni kod primera.*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | <!DOCTYPE html>  <html>  <head>   <title>Standard 3-Column Layout Demo</title>   <style>    body{     font-size: 100%;     color: white;    }    .wrapper {     width: 960px;     margin: 0 auto;    }    .center{     width: 100%;    }    .aside {     float:left;     width: 229px;    }    .main {     float: left;     width: 480px;     padding: 10px;     border-style: solid;     border-width: 0px 1px 0px 1px;     border-color: #929296;      }    .header, .footer, .aside, .center{     font-size: 1.2em;     text-align: center;    }    header, footer{     line-height: 60px;    }    header{     background-color: #929296;    }    .center{     background-color: #ABCADB;     overflow: hidden;    }    .main{       text-align: left;    }    .main > h2{     text-align: center;    }    footer{     background-color: #5F5F64;    }    /\*another technique for clearing floated elements\*/    /\*.clear-fix:after {     visibility: hidden;     display: block;     content: "";     clear: both;     height: 0;    }\*/   </style>  </head>  <body>   <div class="wrapper">    <header class="header">     Header    </header>    <div class="center clear-fix">     <aside class="aside aside-1">      Aside 1     </aside>     <div class="main">      <h2>Main</h2>      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque pretium varius varius. Duis volutpat elementum neque, sed congue neque luctus ut. Nunc ligula lacus, hendrerit eget urna finibus, tristique ullamcorper nulla. Pellentesque congue laoreet eros. Nunc sit amet ipsum id ex rhoncus ornare quis eu mi. Morbi ac finibus odio. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nunc mattis tempus tempor. Etiam at neque sed nisl ornare semper pharetra non sapien. Praesent pellentesque, ipsum a fermentum fermentum, sapien dui pharetra felis, in dictum arcu urna in nibh. Aliquam pulvinar pellentesque nisl, pharetra ultricies mauris auctor vel. Aliquam erat volutpat. Vestibulum suscipit ut ligula finibus semper. In hac habitasse platea dictumst. Fusce ante nisi, auctor vel fermentum sed, tincidunt non ex. </p>     </div>     <aside class="aside aside-2">      Aside 2     </aside>    </div>    <footer class="footer">     Footer    </footer>   </div>  </body>  </html> |

*Efekat koji se na ovaj način proizvodi prikazan je na slici 15.19.*



*Slika 15.19 – Često korišćeni raspored komponenata na sajtovima*

*Prikazani primer ilustruje kreiranje layouta sačinjenog od headera, footera i centralnog dela sa tri kolone. Leva i desna kolona predstavljaju bočne trake (sidebars), dok je centralna kolona rezervisana za sadržaj.*

*Kompletna HTML struktura stranice započinje kontejnerskim div elementom sa klasom wrapper. To je kontejner koji će obuhvatiti kompletan sadržaj stranice. Element wrapper ima definisanu širinu na 960 px i centriran je unutar body elementa. To je postignuto na sledeći način:*

|  |  |
| --- | --- |
| 1  2  3  4 | .wrapper {  width: 960px;  margin: 0 auto;  } |

*Unutar wrapper elementa nalaze se tri direktna potomka: header, center i footer. S obzirom da su u pitanju block elementi, oni će zauzeti kompletnu dostupnu širinu unutar svog roditelja (wrapper).*

*Visina div elemenata formira se na osnovu visine pripadajućeg sadržaja. Na taj način, div bez sadržaja neće imati visinu uopšte. S obzirom da header i footer sadrže po jednu reč teksta, njihova visina zavisiće od veličine takvog teksta. U primeru, visina header i footer elemenata regulisana je korišćenjem line-height svojstva.*

|  |  |
| --- | --- |
| 1  2  3 | header, footer{  line-height: 60px;  } |

*Interesantna situacija se dešava unutar center elementa koji se sastoji iz tri kolone. Tri div elementa su raspoređena jedan pored drugog u jednom redu. To je postignuto korišćenjem float svojstva i vrednosti left. Pored toga, svaki od div elemenata koji predstavlja jednu kolonu ima definisanu širinu.*

*Nešto ranije je rečeno da kontejner koji sadrži isključivo floatovane elemente nema svoju visinu, s obzirom da su njegovi svi potomci izbačeni iz normalnog toka dokumenta. Takav slučaj je i u prikazanom primeru sa elementom center. To stvara različite probleme u primeru prikazanog layouta. S obzirom da center element ne poseduje svoju visinu, on će biti preklopljen sa footerom. Takođe, boja pozadine koja je postavljena na center elementu neće biti vidljiva zbog nepostojanja visine. Opisani problemi mogu se rešiti na više načina. Postavljanje vrednosti overflow svojstva na bilo šta osim visible ima za efekat da se element proširi taman toliko kako bi obuhvatio sve svoje elemente, pa čak i kada su oni floatovani. Ovo rešenje se zasniva na sporednom efektu svojstva overflow.*

*Nešto bolje rešenje jeste definisanje elementa odmah ispod kontejnerskog elementa koji je kolapsirao. To se elegantno postiže na sledeći način:*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | .clear-fix::after {  visibility: hidden;  display: block;  content: "";  clear: both;  height: 0;  } |

U primeru je iskorišćen jedan pseudoelement, korišćenjem selektora ::after. Na taj način je nakon elementa sa klasom clear-fix, dodat jedan element, koji se zapravo neće pojaviti u HTML strukturi stranice (osobina pseudoelemenata). Na takvom elementu je definisano svojstvo clear sa vrednošću both, što ima za efekat sređivanje problema nastalih kao posledica floatovanja.