
# Osnove razvoja web i mobilnih aplikacija

Laboratorijske vježbe

## LV4
## Uvod u CSS raspored

**Fakultet elektrotehnike računarstva i informacijskih tehnologija Osijek**
Kneza Trpimira 2b
**www.ferit.unios.hr**

# CSS tehnike za raspoređivanje elemenata

Postoje različite CSS tehnike za raspoređivanje HTML elemenata kojima se kontrolira njihova pozicija koja je relativna odnosu prema njihovu po zadanom normalnom toku rasporeda, drugim elementima oko njih, njihovu roditeljskom kontejneru i viewport-u (prozoru). Neke tehnike pozicioniranja su:

- Normalni tok elemenata,
- Svojstvo *display*,
- Float,
- Flexbox,
- Grid... .

Svaka tehnika ima svoju primjenu, prednosti i nedostatke, te nijedna tehnika nije kreirana za samostalno korištena, već se trebaju koristiti različite tehnike za različite zadaće.

# Normalni tok

Normalni tok predstavlja način kako preglednik po zadanom raspoređuje HTML elemente po web stranici, bez vanjskog-dodatno CSS utjecaja kojim bi se kontrolirao tok rasporeda elemenata.

```
<p>Ja volim svog kućnog ljubimca</p>
<ul>
    <li>Kupujem mu hranu</li>
    <li>Dresiram ga</li>
    <li>Veseli me</li>
</ul>
<p>Sretan sam!</p>
```

HTML elementi su prikazani istim redoslijedom kako se pojavljuju unutar izvornog kôda i to tako da su elementi "složeni" jedan na drugi. Svi elementi su prikazani modelom kutije.

Elementi koji su prikazani jedan ispod drugoga su opisani kao *block* elementi, dok postoje *inline* elementi koji se slažu jedan pored drugoga, kao riječi unutar paragrafa. *Block* elementi će biti razmaknuti jedan od drugoga postavljenom marginom i ako se dvije margine dodiruju, razmak između dva elementa će tvoriti veća margina.

Po zadanom, sadržaj *block* elementa je 100% širine svog roditeljskog elementa, a visok je koliko mu sadržaj po visini zauzima mjesta. *Inline* elementi su visoki i široki koliko i njihov sadržaj. Nije moguće postaviti visinu i širinu *inline* elementu, oni se samo nalaze unutar sadržaja roditeljskog *block* elementa. Ako postoji potreba za upravljanjem veličinom *inline* elementa, potrebno je postaviti ponašanje takvog elementa u *block* ili *inline-block*.

Glavni načini postizanja rasporeda stranice u CSS-u omogućeno je pomoću vrijednosti svojstva *display*. Ovo svojstvo omogućava promjenu kako se nešto prikazuje. Sve u normalnom toku ima vrijednost svojstva *display*, npr. *display: block* ili *display: inline*. Za svaki element se može promijeniti *display* svojstvo. Npr. element <li> po zadanom ima vrijednost svojstva *display: block*, to znači kako se <li> elementi unutar web stranice prikazuju jedan ispod drugoga. Ako se promijeni vrijednost svojstva display u *inline*, svaki <li> element će se prikazati jedan pored drugoga, kao što bi se riječi našle jedna pored druge unutar rečenice.

Kada se koristi CSS za kreiranje rasporeda, elementi se miču iz normalnog toka , iako za mnoge elemente na web stranici normalni tok će kreirati željeni raspored. Zato je vrlo važno kako se počinje sa strukturom HTML dokumenta, jer se tada može koristiti normalni tok rasporeda elemenata, umjesto da se bori protiv tog rasporeda. Načini kojima se mogu rasporediti elemenata su:

- Svojstvo *display*: standardne vrijednosti kao što su *block*, *inline*, *inline-block*, mogu promijeniti kako se ponaša element u normalnom toku.
- Svojstvo *float*: primjenjivanjem *float* vrijednosti nad *block* elementom omogućava elementu pozicioniranje na lijevu ili desnu stranu kontejnera.

## Plutanje (engl. Floats)

Svojstvo *float* prvenstveno je bilo namijenjeno za pozicioniranje slike unutar teksta, ali uvidjelo se kako se s ovim svojstvom može kreirati stupčasti raspored web stranice.

Svojstvo *float* se učestalo počelo koristiti za kreiranje rasporeda za cijelu web stranicu koja se sastoji od višestrukih stupaca informacija koji plutaju jedan pored drugoga (prirodno bi ti stupci bili složeni jedan ispod drugoga). Važno je za napomenuti kako postoje novije i bolje tehnike rasporeda elemenata od primjene svojstva *float* i kako se svojstvo *float* može smatrati zastarjelom tehnikom pozicioniranja, odnosno da ga se treba koristiti samo za ono za što je bio namijenjen, pozicioniranje slika unutar teksta.

Ako se element postavi da pluta, mijenja se ponašanje toga elementa i *block* elementa koji ga slijedi u normalnom toku. Element se isključuje iz normalnog toka i miče se lijevo ili desno, a preostali sadržaj pluta oko elementa. Neke vrijednosti *float* svojstva:

- left: element se miče u lijevo,
- right: element se miče desno,
- none: isključuje se plutanje, ovo je po zadanom vrijednost.

Primjer:

```
<h1>Jednostavni float primjer</h1>
<div class="kutija">Float</div>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Repellat
suscipit natus itaque asperiores reiciendis rem voluptate sint
aliquam alias ipsum deserunt minima autem id porro tempore totam
minus beatae dolorum at ea aut, saepe commodi eum cum? Nostrum nihil
minus, illo quia, nulla maxime temporibus similique voluptates nobis
ipsa voluptas!</p>
```

```css
body {
    width: 600px;
    margin: 0 auto;
}
.kutija {
    float: left;
    width: 50px;
    height: 50px;
    margin-right: 30px;
    background-color: aqua;
}
```

Objašnjenje:

Element nad kojim se primijenilo svojstvo *float* (u ovom slučaju ) je izvađen iz normalnog toka rasporeda dokumenta i uglavljen je u lijevu stranu svog roditeljskog kontejnera (u ovom slučaju <body>). Bilo koji sadržaj koji dolazi ispod elementa koji pluta u normalnom toku rasporeda bit će omotano oko plutajućeg elementa, ispunjavajući prostor s desne strane i vrha samog plutajućeg elementa. Postavljanjem elementa da pluta s desne strane imat će isti efekt, samo obrnuto od prethodno objašnjenog.

Plutajućem elementu se mogu postaviti margine kako bi odgurnuo tekst kako je prikazano prethodnim primjerom, dok tekstu se ne mogu staviti margine kako bi se tekst odgurnuo od plutajućeg elementa. To je zato što je plutajući element isključen iz normalnog toka i kutije preostalih elemenata se pružaju iza plutajućeg elementa.

Primjer:

```html
<h1>Jednostavni float primjer</h1>
<div class="kutija">Float</div>
<p class="pruza">Lorem ipsum dolor sit amet consectetur adipisicing elit.
Quaerat voluptates quidem id repellat voluptatem. Sit totam facere iusto
sapiente eos.</p>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Repellat
suscipit natus itaque asperiores reiciendis rem voluptate sint aliquam
alias ipsum deserunt minima autem id porro tempore totam minus beatae
dolorum at ea aut, saepe commodi eum cum? Nostrum nihil minus, illo quia,
nulla maxime temporibus similique voluptates nobis ipsa voluptas!</p>
```

```css
.pruza {
    background-color: rgb(79, 185, 227);
    padding: 10px;
    color: #fff;
}
```

## Uklanjanje plutanja

Primjerom je prikazano kako se element koji pluta uklanja iz normalnog toka i kako se drugi elementi pružati iza plutajućeg elementa. Stoga ako se želi sljedeći elementi osloboditi od utjecaja plutajućeg elementa, potrebno je koristiti svojstvo *clear*.

Primjer:

```html
<h1>Jednostavni float primjer</h1>
<div class="kutija">Float</div>
<p class="pruza">Lorem ipsum dolor sit amet consectetur adipisicing elit. Quaerat voluptates quidem id repellat voluptatem. Sit totam facere iusto sapiente eos.</p>
<p class="osloboden">Lorem ipsum dolor sit amet consectetur adipisicing elit. Repellat suscipit natus itaque asperiores reiciendis rem voluptate sint aliquam alias ipsum deserunt minima autem id porro tempore totam minus beatae dolorum at ea aut, saepe commodi eum cum? Nostrum nihil minus, illo quia, nulla maxime temporibus similique voluptates nobis ipsa voluptas!</p>
```

```css
.osloboden {
    clear: left;
}
```

Drugi paragraf se oslobodio plutajućeg elementa i vise mu se ne nalazi s desne strane, već kao *block* element je prešao u novu liniju. Svojstvo *clear* prima sljedeće vrijednosti:

- left: oslobađa od element koji pluta u lijevu stranu,
- right: oslobađa od elementa koji pluta u desnu stranu,
- both: oslobađa od plutajućeg elementa, bilo lijeva ili desna strana.

# Flexbox

Flexbox je skraćeno od (engl. *Flexible Box Layout*) – fleksibilni raspored kutija. Kreiran je kako bi olakšao postavljanje elemenata unutar jedne dimenzije, u red ili stupac. Kako bi se primijenio flexbox nad određenim elementima, potrebno je postaviti unutar roditeljskog elementa svojstvo *display* na vrijednost *flex*. Na ovaj način će sva djeca elementi postati flex elementi.

Kroz jedan jednostavni primjer koji se sastoji od zaglavlja i tri članka unutar jednog odjeljka, pokazat će se primjena flexbox-a.

Primjer:

```
<header><h1>Jednostavni Flexbox primjer</h1></header>
<section>
    <article>
        <h2>Prvi članak</h2>
<p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Velit
consectetur blanditiis non harum animi natus temporibus ut quo optio
sit.</p>
    </article>
    <article>
        <h2>Drugi članak</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eveniet
pariatur veniam iure vitae illo facere maxime necessitatibus rerum
doloremque minus.</p>
    </article>
    <article>
        <h2>Treći članak</h2>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Architecto
nemo fuga, sequi necessitatibus minima vero incidunt unde possimus
distinctio dolores.</p>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam
quae deserunt provident maiores nobis eaque harum fuga sequi quam
neque.</p>
</article>
```

```
body {
    margin: 0;
    font-family: sans-serif;
}
header {
    background: orange;
    height: 100px;
}
h1 {
    text-align: center;
    color: white;
    line-height: 100px;
    margin: 0;
}
article {
    padding: 10px;
    margin: 10px;
    background: aqua;
 }
```

## Definiranje elemenata za fleksibilne kutije

Prilikom kreiranja rasporeda potrebno je definirati elemente koji će postati fleksibilne kutije. To se postiže tako da se unutar roditeljskog elementa postavi vrijednost svojstva *display* na *flex*.

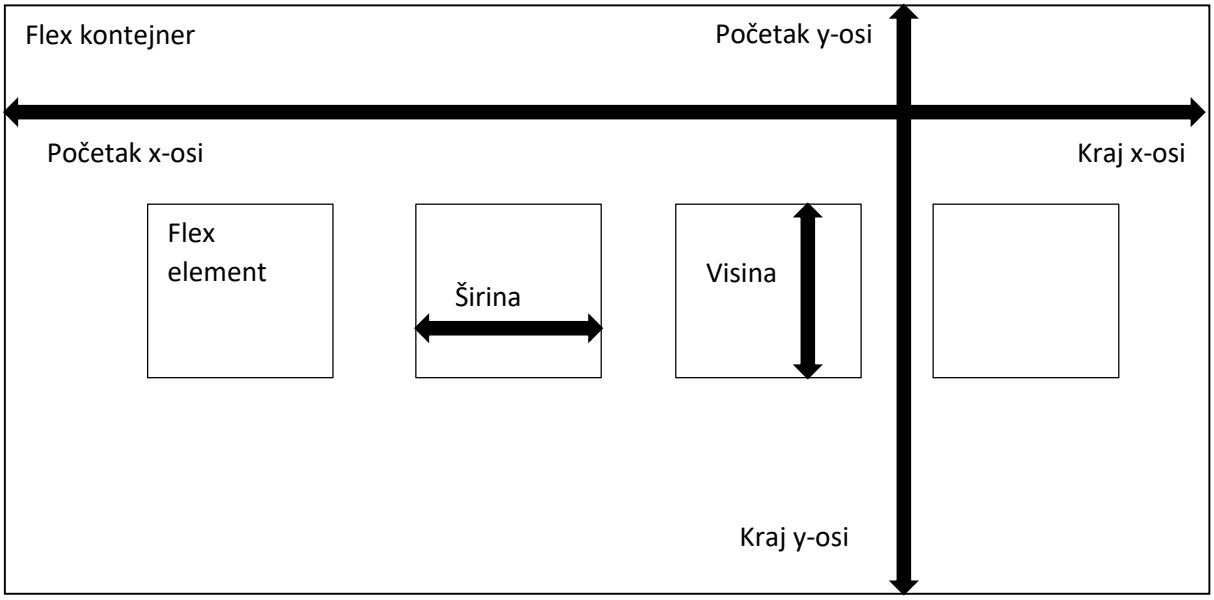
Roditeljski element u ovom primjeru je element <section> koji sadrži tri <article> elementa.

```
section {
    display: flex;
}
```

Normalnim tokom, <article> elementi bi se složili jedan ispod drugoga jer su to *block* elementi, no kako se sada koristi vrijednost *flex* za sve elemente koji se nalaze unutar roditeljskog elementa <section>, svi <article> elementi su postali *flex* elementi koji su složeni unutar jednog reda, jedan pokraj drugoga.

Koristeći vrijednost *inline-flex* moguće je rasporediti na isti način *inline* elemente.

## Flex model




## Redovi i stupci

Flexbox ima svojstvo *flex-direction* kojim se određuje u kojem će smjeru bit postavljeni flex elementi, da li u redove po x-osi (vrijednost *row*) ili u stupce po y-osi (vrijednost *column*). Po zadanom je vrijednost svojstva *flex-direction*, *row*.

## Prelijevanje

Jedan od problema koji se može pojaviti kada se koriste fiksne vrijednosti za širinu i visinu rasporeda jest prelijevanje djece flex elemenata iz njihovih kontejnera, što će na kraju iskriviti raspored. Jedan od načina kako se to može popraviti jest korištenje svojstva *flex-wrap* kojem se postavi vrijednost na *wrap*.

Postoji i skraćeno svojstvo *flex-flow* kojim se zamjenjuju svojstva *flex-direction* i *flex-wrap*.

```
flex-direction: row;
flex-wrap: wrap;

flex-flow: row wrap;
```

## Fleksibilna promjena veličine flex elementa

U Flexboxu postoji način kontrole koliko će svaki flex element zauzeti proporcionalno praznog prostora. To se ostvaruje pomoću svojstva *flex* kojemu se dodjeljuje bez jedinična cjelobrojna numerička vrijednost.

```
article {
    flex: 1;
}
```

Ovim primjerom je pokazano koliko će svi <article> element zauzeti slobodnog prostora po x-osi, svaki element će zauzeti jednak dio prostora nakon pridruživanja margine i ispunjena, jer se svakom flex elementu dodjeljuje proporcionalna vrijednost jedan.

```
article:nth-of-type(3){
    flex:2;
}
```

Nakon što se trećem elementu pridružilo svojstvo *flex* s vrijednosti dva, taj element će sada zauzimati dvostruko više prostora naspram preostalih elemenata unutar roditeljskog flex kontejnera. Preciznije, prva dva elementa će zauzimati ¼ slobodnog prostora, dok će zadnji element zauzimati 2/4 slobodnog prostora.

Također je moguće definirati minimalnu veličinu unutar flex svojstva.

```
article {
    flex: 1 200px;
}
article:nth-of-type(3){
    flex:2 200px;
}
```

Gornjim primjerom se svakom flex elementu pridružuje 200px slobodnog prostora, a nakon toga se preostali prostor dijeli prema proporcijskoj veličini.

Upravo se stvarna vrijednost flexbox-a može vidjeti u fleksibilnosti i prilagodljivosti veličini ekrana. Ako se mijenja veličina viewport preglednika ili se doda još jedan <article> element, raspored će i dalje vrlo dobro funkcionira.

Svojstvom *flex-basis* postavlja se inicijalna veličina flex elementa, moguće je koristiti vrijednost izraženu u postocima, te određenim poravnanjem, moguće je stvoriti određeni razmak između elemenata. Koristi se umjesto svojstva *flex*.

## Horizontalno i vertikalno poravnanje

Moguće je koristiti Flexbox svojstva za poravnanje flex elemenata po x i y osi, odnosno vertikalno i horizontalno.

Svojstvom *justify-content* poravnavaju se flex elementi po x-osi (horizontalno) i prima sljedeća vrijednosti:

- flex-start: ovo je po zadanom vrijednost, svi elementi su poravnati od početka x-osi,
- flex-end: svi elementi su poravnati na kraj x-osi,
- center: svi elementi su poravnati u sredini x-osi,
- space-around: raspoređuje sve elemente jednako po x-osi, s malo praznog prostora na početku i kraju,
- space-between: raspoređuje sve elemente jednako po x-osi, bez praznog prostora na početku i kraju.

Svojstvom *align-items* poravnavaju se flex elementi po y-osi (vertikalno) i prima sljedeće vrijednosti:

- stretch: ovo je po zadanom vrijednost, rasteže sve flex elemente kako bi popunili prostor unutar roditeljskog kontejnera. Ako roditeljski kontejner nema zadanu fiksnu visinu, tada će svi flex elementi postati visoki kao najviši element,
- center: svi elementi su postavljeni u sredini y-osi,
- flex-start: poravnava elemente s početka y-osi,
- flex-end: poravnava elemente na kraj y-osi.

## Redoslijed flex elemenata

Flexbox ima svojstvo kojim se može mijenjati redoslijed flex elemenata, bez mijenjanja izvornog kôda HTML-a. To se postiže svojstvom *order* kojemu se može postaviti cjelobrojna numerička vrijednost s predznakom:

- Svi flex elementi imaju po zadanom postavljenu vrijednost nula za svojstvo *order*.
- Flex elementi s većom vrijednosti svojstva *order* biti će postavljeni poslije elemenata koji imaju nižu vrijednost svojstva *order*.
- Flex elementi koji imaju jednaku vrijednost svojstva *order* biti će postavljeni prema poziciji unutar izvornog kôda.
- Mogu se postaviti negativne vrijednosti kako bi se elementi pojavili prije elemenata s vrijednosti nula.

```
article:nth-of-type(3){
     order: 0;
}
article:nth-of-type(2){
     order: 2;
}
article:nth-of-type(1){
     order: 1;
}
```

## Ugnježđivanje flex kutija

Moguće je kreirati kompleksni raspored pomoću flexbox-a. Svaki flex element može biti flex kontejner, kako bi njegova djeca elementi bili raspoređeni kao flex kutije.

Primjer:

Prethodnom primjeru u treći <article> element pridružena su tri  elementa, a sam <article> element je postavljen kao flex kontejner koji sadrži  elemente koji su postali flex elementi (flex kutije).

```
<article><!-- <h2>Treći članak</h2>-->
<div>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda, in?</div>
<div>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Officiis, consectetur!</div>
<div>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque, quas.</div>
</article>
```

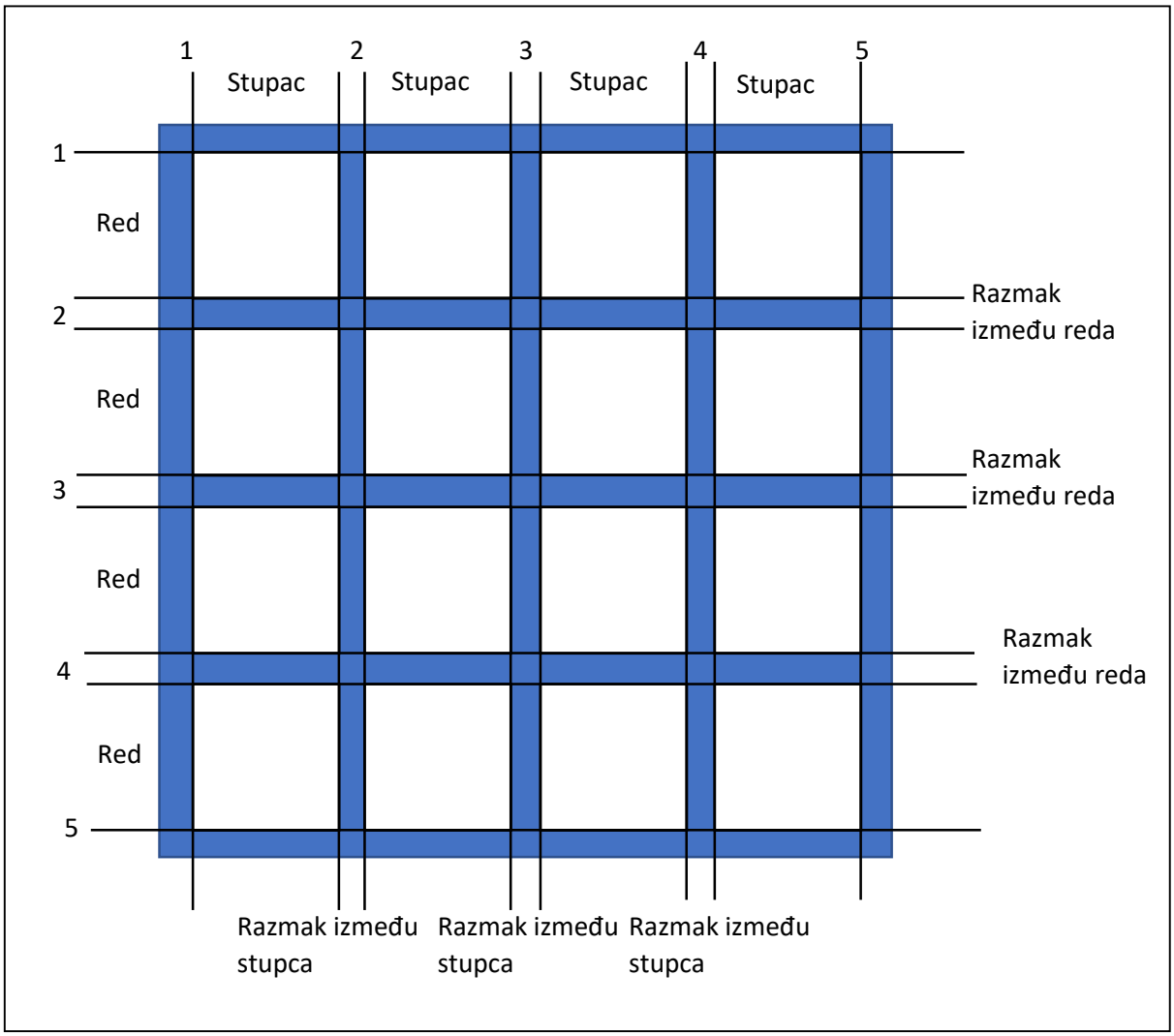
```
article:nth-of-type(3) {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
article:nth-of-type(3) div {
    flex: 1 50px;
}
```

# Grid raspored

Dok je flexbox kreiran za jednu dimenziju, grid je kreiran za dvije dimenzije, postavljajući elemente unutar redova i stupaca. Grid raspored se može uključiti tako da se svojstvu *display* dodjeli vrijednost *grid*.

Grid je kolekcija horizontalnih i vertikalnih linija koje tvore uzorak unutar kojega se postavljaju elementi. Ovim načinom se omogućuje velika konzistentnost prilikom kreiranja web stranice. Grid sadrži stupce i retke koji mogu imati između sebe prazninu (marginu) radi ljepše prezentacije.

## Grid model



## Kreiranje Grid-a

Kada se odluči na željeni Grid raspored koji zahtjeva zamišljeni raspored web stranice, potrebno je pomoću CSS-a kreirati Grid i postaviti elemente na željenu poziciju.

### Definiranje grid-a

Korištenjem jednostavnog primjera, pokazat će se princip rada grid rasporeda. Koristi se naslov i glavni  kontejner koji unutar sebe sadrži sedam  elemenata koji su djeca elementi glavnog kontejnera.

Primjer:

```
<h1>Jednostavni grid primjer</h1>
<div class="kontejner">
    <div>Jedan</div>
    <div>Dva</div>
    <div>Tri</div>
    <div>Četiri</div>
    <div>Pet</div>
    <div>Šest</div>
    <div>Sedam</div>
</div>
```

```
body {
    width: 90%;
    max-width: 900px;
    margin: 2em auto;
    font: .9em/1.2 Arial, Helvetica, sans-serif;
}
.kontejner > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(207, 232, 220);
    border: 2px solid rgb(79, 185, 227);
}
```

Po zadanom, svi elementi će se posložiti prema normalnom toku rasporeda, jedan ispod drugoga.

Kako bi se definirao grid potrebno je svojstvu *display* postaviti vrijednost na *grid*, a tu izmjenu treba primijeniti na roditeljskom kontejneru unutar CSS. Kada se ta izmjena provede, sva djeca elementi će postati grid elementi.

```
.kontejner {
    display: grid;
}
```

Nasuprot flexbox-u, elementi se neće odmah promijeniti. Deklaracijom *display: grid* automatski se kreira grid s jednim stupcem, tako da će se elementi u početku ponašati kao da su dio normalnog toka.

Kako bi raspored počeo ličiti na grid, potrebno je postaviti stupce, a to se postiže navodeći svojstva *grid-template-columns* unutar kontejnerski element. Za vrijednosti navedenog svojstva mogu se koristiti bilo koje veličine za duljinu i postotci.

```
.kontejner {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}
```

### Fleksibilni grid s *fr* jedinicom

Uz kreiranje grid-a pomoću veličine za duljinu i postotke, može se koristiti jedinica *fr* kojom se kreira fleksibilna promjena veličine stupaca i redova. Ovom jedinica moguće je postići podjelu preostalog prostora unutar grid kontejnera.

```
.kontejner {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
}
```

*fr* jedinica proporcionalno raspoređuje slobodni prostor unutar kontejnerskog elementa, stoga moguće je dodijeliti različite vrijednosti *fr* jedinice različitim elementima. Ako neki element ima veću vrijednost *fr* jedinice, taj element će zauzima više prostora.

```
.kontejner {
    display: grid;
    grid-template-columns: 3fr 1fr 1fr;
}
```

Prvom stupcu je dodijeljena vrijednost *3fr*, dok ostalim stupcima je dodijeljena vrijednost *1fr*, te će prostor biti proporcionalno raspoređen, gdje će stupac s *3fr* zauzeti najviše mjesta. Moguće je miješati

*fr* jedinice s fiksnim duljinama, gdje će se u tome slučaju prazni prostor za fiksne veličine prvo zauzeti, a nakon toga će se pomoću *fr* jedinica raspodijeliti preostali prostor.

```
.kontejner {
    display: grid;
    grid-template-columns: 50% 1fr 1fr;
}
```

### Razmaci između elemenata

Kako bi se kreirali razmaci između elemenata koji tvore stupce potrebno je koristiti svojstvo *grid-column-gap*, a za kreiranje razmaka između redova potrebno je koristiti svojstvo *grid-row-gap*. Postoji svojstvo *grid-gap* kojim se postavlja razmak i za stupce i za redove. Vrijednosti koje mogu primiti ova svojstva mogu biti bilo koje jedinice za duljinu, postotak ali nikako *fr* jedinica.

```
.kontejner {
    display: grid;
    grid-template-columns: 50% 1fr 1fr;
    grid-gap: 20px;
}
```

### Ponavljajuća lista zapisa

Drugi naziva za stupac ili redak je zapis. Moguće je ponavljati sve elemente zapisa, ili dio korištenjem funkcije *repeat()*. Funkcija *repeat()* prima dva argumenta, prvi argument predstavlja broj ponavljanja, dok drugi argument predstavlja podjelu praznog prostora za pojedini stupac ili red.

```
.kontejner {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
}
```

### Implicitni i eksplicitni grid

Do sada se kroz primjere pokazalo kreiranje stupaca, ali unatoč tome, bili su kreirani i redovi koji su sadržavali sadržaj. Ovo je bio primjer eksplicitnog i implicitnog grid-a. Eksplicitni grid se kreira pomoću svojstva *grid-template-columns* ili *grid-template-rows*. Implicitni grid se kreira kada se sadržaj stavi izvan grid-a, kao u prethodnom primjeru za stupce. Kreirana su tri stupca, ali bilo je sedam elemenata, pa su se preostali elementi razmjestili u svoje redove.

Po zadanom, zapisi kreirani implicitnim grid-om su *auto* veličine, to znači da su dovoljno veliki kako bi primili svoj sadržaj. Ako se želi zapisu zadati veličina mogu se koristiti svojstva *grid-auto-rows* i *grid-auto-columns*.

```
.kontejner {
     display: grid;
     grid-template-columns: repeat(3, 1fr);
     grid-auto-rows: 100px;
     grid-gap: 20px;
}
```

## Funkcija minmax()

Redovi koji su postavljeni na 100px visine, neće biti od velike koristi ako se u njih stavi sadržaj koji je viši od 100px, što bi u tome slučaju izazvalo prelijevanje. Bilo bi dobro kada bi se podesila minimalna visina redova uz mogućnost proširivanja kada se stavi sadržaj unutar njih. Situacija s web sadržajem je takva da se ne može sa sigurnošću znati koliko će nešto biti visoko, jer dodatnim sadržajem ili promjenom fonta može se dovesti do problema s dizajnom.

Funkcija minmax() omogućava postavljanje minimalne i maksimalne veličine zapisa. Funkcija prima dva argumenta, za redove prvi argument predstavlja minimalnu visinu, dok drugi element predstavlja maksimalnu visinu.

```
.kontejner {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-gap: 20px;
}
```

## Linijsko pozicioniranje elemenata

Grid uvijek ima linije koje počinju od broja jedan. Prva linija stupca uvijek počinje na krajnjoj lijevoj strani, dok prva linija reda počinje odozgo. Pozicioniranje elemenata unutar pojedine ćelije grida omogućava se pomoću svojstva:

- grid-column-start,
- grid-column-end,
- grid-row-strat,
- grid-row-end.

Također se mogu koristiti skraćena svojstva:

- grid-column,
- grid-row.

Sva ova svojstva mogu imati broj linije za vrijednost. Početak i kraj linije se navodi s početnim brojem linije od koje se element širi, razdvojeno s znakom (/) iza kojeg slijedi krajnji broj do kojeg se element širi.

Kada se kreira grid, svi elementi će biti automatski postavljeni unutar ćelija redoslijedom kako su postavljeni u izvornom kôdu HTML-a. Ako želimo napraviti promjenu rasporeda pojedinih elemenata, nije potrebno odlaziti u izvorni kôd HTML-a, već se primijeniti preraspodjela pomoću linijskog pozicioniranja.

```
<h1>Jednostavni grid primjer</h1>
<div class="kontejner">
    <div class="kut1">Jedan</div>
    <div class="kut2">Dva</div>
    <div class="kut3">Tri</div>
    <div class="kut4">Četiri</div>
    <div class="kut5">Pet</div>
    <div class="kut6">Šest</div>
    <div class="kut7">Sedam</div>
</div>
```

```css
.kut7 {
    grid-column: 1 / 4;
    grid-row: 1;
}
.kut5 {
    grid-column: 1 / 4;
    grid-row: 2;
}
.kut6 {
    grid-column: 1 / 3;
    grid-row: 3 / 6;
}
.kut1 {
    grid-column: 3;
    grid-row: 3 / 6;
}
.kut2 {
    grid-column: 1 / 3;
    grid-row: 6 / 9;
}
.kut4 {
    grid-column: 3;
    grid-row: 6 / 9;
}
.kut3 {
    grid-column: 1 / 4;
    grid-row: 9;
}
```

Pozicioniranje sa svojstvom grid-template-areas

Alternativni način pozicioniranje grid elemenata postiže se pomoću svojstva *grid-template-areas* i dodjeljivanjem imena pojedinim elementima koji su dio rasporeda.

```
.kontejner {
    display: grid;
    grid-template-areas:
    "header header header"
    "nav nav nav"
    "article1 article1 aside1"
    "article1 article1 aside1"
    "article2 article2 aside2"
    "article2 article2 aside2"
    "footer footer footer";
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-gap: 20px;
}
```

```
.kut7 {
    grid-area: header;
}
.kut5 {
    grid-area: nav;
}
.kut6 {
    grid-area: article1;
}
.kut1 {
    grid-area: aside1;
}
.kut2 {
    grid-area: article2;
}
.kut4 {
    grid-area: aside2;
}
.kut3 {
    grid-area: footer;
}
```

Pravila za *grid-template-areas* su sljedeća:

- Potrebno je popuniti svaku ćeliju grid-a,
- Za obuhvaćanje više ćelija, potrebno je ponoviti ime,
- Ako se želi ostaviti ćeliju praznom, na tome mjestu se koristiti (.),
- Ćelije moraju biti kvadratne, ne može se dobiti oblik slova L,
- Ćelije se ne mogu ponoviti na nekom drugom mjestu unutar grid-a.

# Zadaci

1. Korištenjem svojstva *float* i *clear*, kreirati raspored prema slici.


2. Koristeći flexbox raspored, napraviti odgovarajući raspored HTML elemenata za web stranicu.
3. Koristeći grid raspored, napraviti odgovarajući raspored HTML elemenata za web stranicu.