

----- FLEXBOX -----

MORAM, PRVO DEFINISATI flex CONTAINER (PARENT ELEMENT) (ILI FLEKSIBILNI CONTAINER)

```
<ul>
| <li></li>
| <li></li>
| <li></li>
| <li></li>
</ul>
```

```
ul {
|   display: flex; /* ili inline-flex */
| }
```

STILIZUJEM DALJE

```
li {
| width: 100px;
| height: 100px;
| background-color: #8cacea;
| margin: 8px;
| list-style-type: none;
| }
```

```
ul {
|   display: flex; /* ili inline-flex */
|   border: red solid 2px;
|   list-style: none;
| }
```

RENDER-OVANO:



ul SADA JESTE FLEX CONTAINER, DOK SU SADA li ELEMENTI, USTVARI FLEX ITEMI

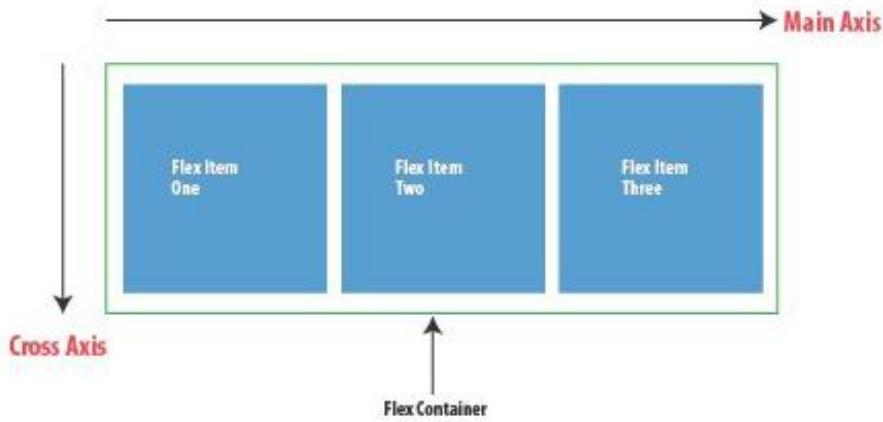
- 1) FLEX CONTAINER: PARENT ELEMENTI, KOJI SU STILIZOVANI SA `display: flex` || `inline-flex`
- 2) FLEX ITEMI: CHILD ELEMENTI, UNUTAR FLEX CONTAINER-A

AKO DEFINISEM DA NEKI ELEMENTI IMA PROPERTI `display`, SA VREDNOSCU `inline-flex`; TO ZNACI DA JE TAJ ELEMENT, USTVARI FLEX CONTAINER, KOJI JE DISPLAYED `inline`

6 ALIGMENT PROPERTIJA, KOJI SE KORISTE NA FLEX CONTAINER-U

→→ flex-direction

PROPERTI KONTROLISE PRAVAC, KOJIM SU FLEX ITEMI POLOZENI DUZ
MAIN AXIS (GLAVNE OSE) ILI HORIZONTALNE



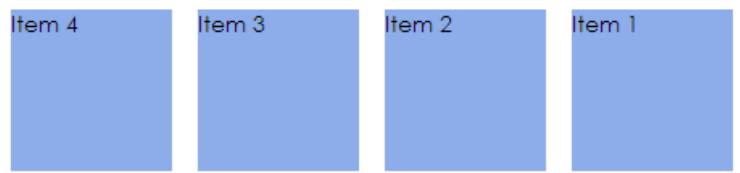
(Left to Right, Top to Bottom)

MAIN AXIS JE HORIZONTALNA (USTVARI, UMESTO HORIZONTALNA, PRAVILNO JU JE ZVATI LEFT-TO-RIGHT), DOK JE CROSS AXIS VERTIKALNA (USTVARI, UMESTO VERTIKALNA, PRAVILNIJE RECI TOP-TO-BOTTOM)

VREDNOSTI, PROPERTIJA: row (DEFAULT) column row-reverse column-reverse

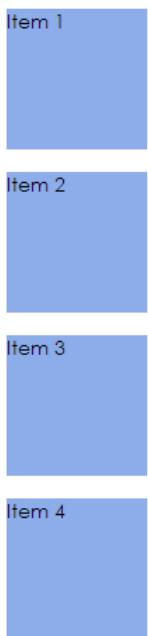
```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
```

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  flex-direction: row-reverse;
}
```



AKO SE DODELI column VREDNOST, ELEMENTI CE BITI POLOZENI DUZ CROSS AXIS

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  flex-direction: column;
}
```



→→ flex-wrap

PROPERTI, KONTROLISE WRAPPING (WRAPPING: KADA SE CONTAINER SMANJI (SIRINU); FLEX ITEMI NECE BITI OVERFLOWED IZ FLEX CONTAINER-A, AKO SE POMENUTOM PROPERTIJU DODELI wrap ili wrap-reverse)

VREDNOSTI PROPERTIJA: no-wrap (DEFAULT) wrap wrap-reverse

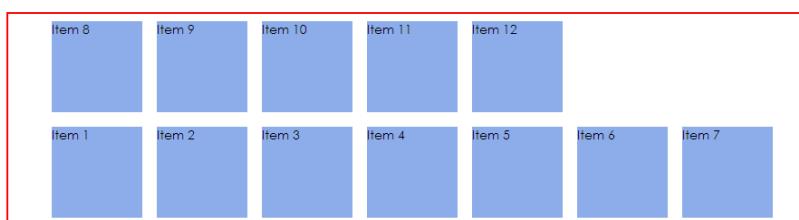
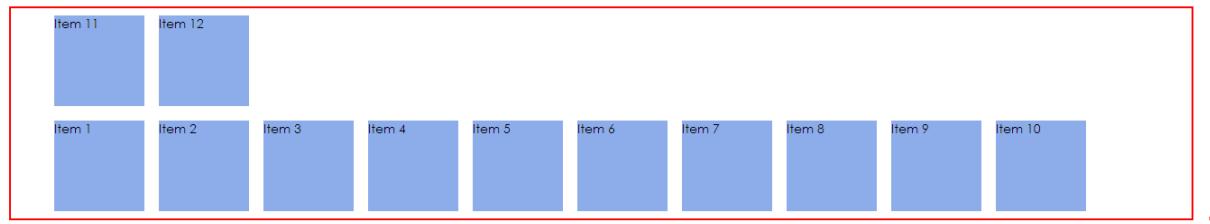
```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
  <li>Item 6</li>
  <li>Item 7</li>
  <li>Item 8</li>
  <li>Item 9</li>
  <li>Item 10</li>
  <li>Item 11</li>
  <li>Item 12</li>
</ul>
```

```
ul {
  display: flex;

  border: red solid 2px;
  list-style: none;

  flex-direction: row;

  flex-wrap: wrap-reverse;
}
```



→ I TAKO DALJE SMANJIVANJEM SIRINE, FLEX

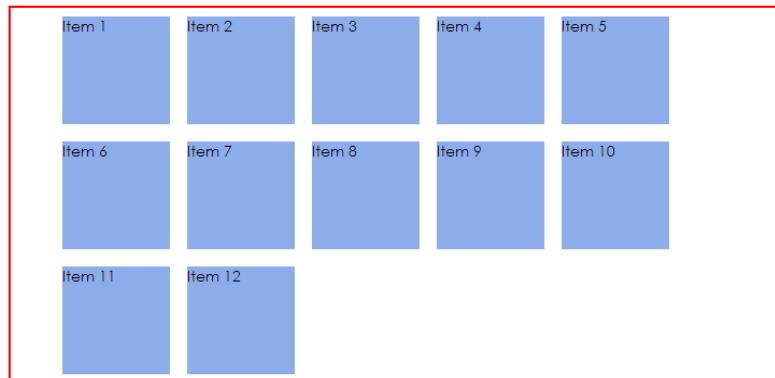
ITEMI CE WRAPPOVATI, SVE DOK SVI NE BUDU U KOLONI (SVAKI RED PO JEDAN CLAN)

→→ flex-flow

(SHORTHAND)

SHORTHAND PROPERTI ZA flex-direction | flex-wrap

```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
     flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
}
```



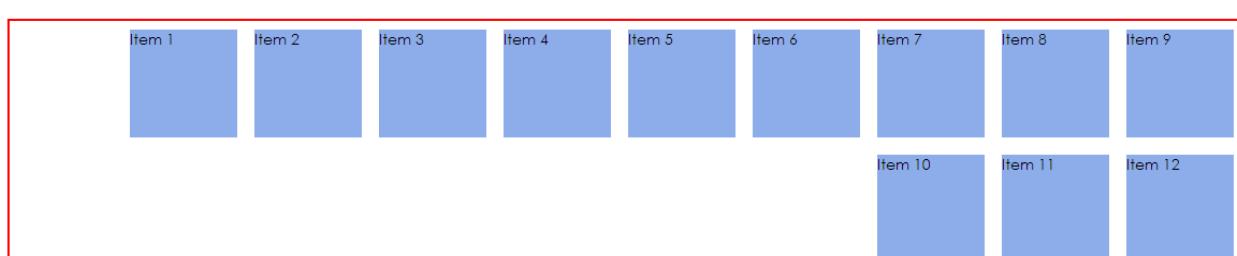
→→ justify-content

POMENUTI PROPERTI DEFINISE, NA KOJI NACIN (ODNOSNO KAKO) SU FLEX ITEMI POLOZENI, DUZ MAIN AXIS-A

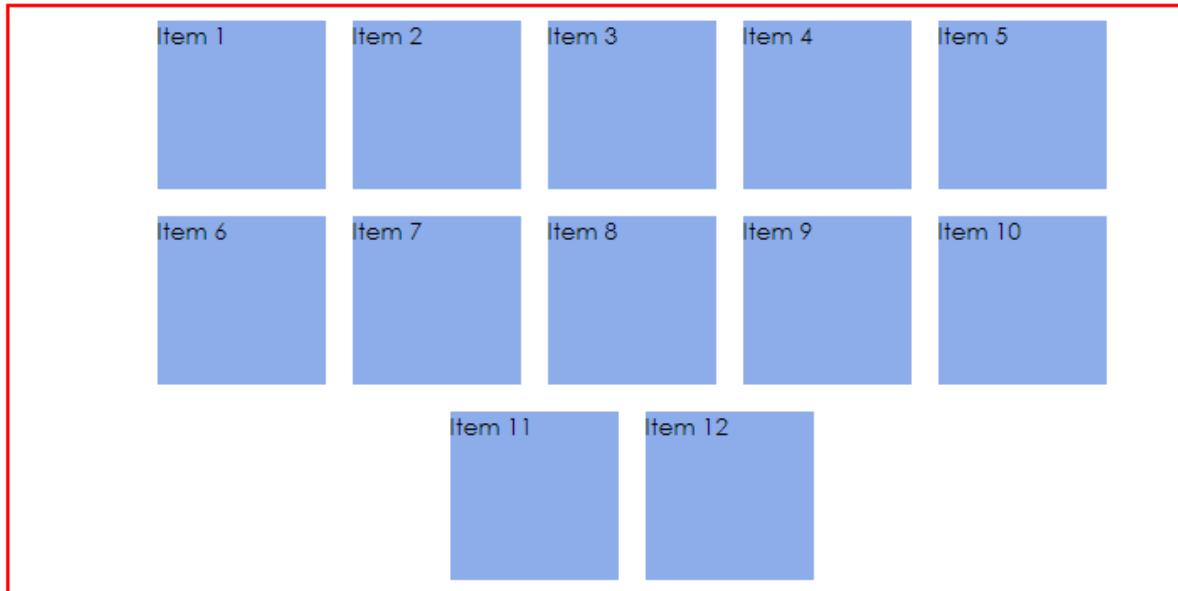
MOZE ME PODSECATI NA text-align PROPERTI

VREDNOSTI PROPERTIJA: flex-start (DEFAULT) flex-end center space-around
 space-between
 space-evenly

```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
     flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
    justify-content: flex-end;  
}
```



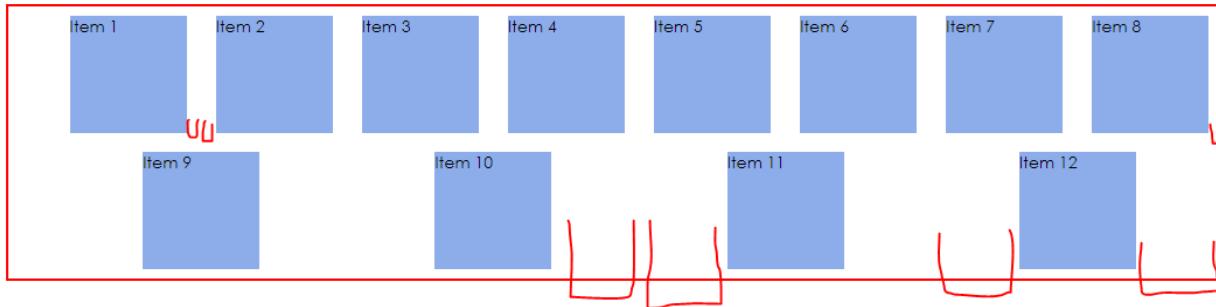
```
ul {  
  display: flex;  
  
  border: red solid 2px;  
  list-style: none;  
  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  
  flex-flow: row wrap;  
  
  justify-content: center;  
}
```



```
ul {  
  display: flex;  
  
  border: red solid 2px;  
  list-style: none;  
  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  
  flex-flow: row wrap;  
  
  justify-content: space-between;  
}
```

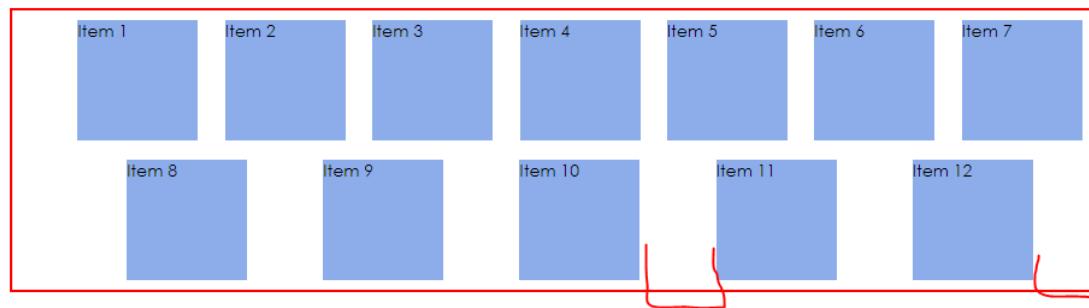


```
ul {  
  display: flex;  
  
  border: red solid 2px;  
  list-style: none;  
  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  
  flex-flow: row wrap;  
  
  justify-content: space-around;  
}
```



```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: space-evenly;
}
```

ITEM-OVI SU DISTRIBUTED, TAKO DA JE SVAKI PROSTOR IZMEDJU DVA SUSEDNA ITEMA JEDNAK, ALI I PROSTOR OD ITEMA DO IVICA



→→ align-items

DEFINISE, NA KOJI NACIN SU FLEX ITEMI POLOZENI DUZ CROSS AXIS, I U TOME JE RAZLIKA, IZMEDJU OVOG I PREDHODNOG PROPERTIJA

VREDNOSTI PROPERTIJA: flex-start flex-end center stretch (DEFAULT) baseline

KAKO BI NAJBOLJE RAZUMEO OVAJ PROPERTI, PRVO CU DODATI, NEKE STILOVE, LIST ITEM-IMA, ODNOSNO FLEX ITEM-IMA, DIREKTNO (MEDJUTIM, NEGDE CE, SVAKI TRECI ITEM, BITI DRUGACIJE STILIZOVAN, A NEGDE, SVAKI DRUGI ELEMENT)

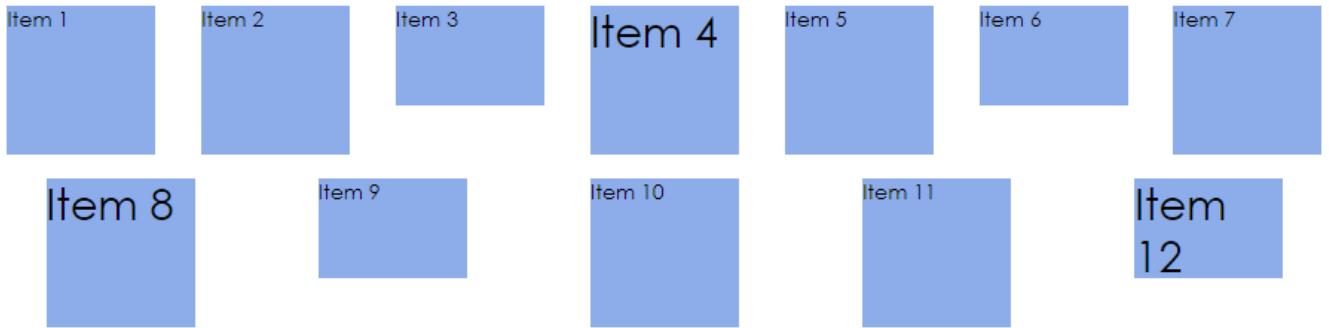
DEFINISACU DA SVAKI TRECI FLEX ITEM IMA VISINU 4.2rem

DEFINISACU DA SVAKI CETVRTI FLEX ITEM IMA TEKST VELICINE 1.8rem

```
ul li:nth-child(3n) {
  height: 4.2rem;
}

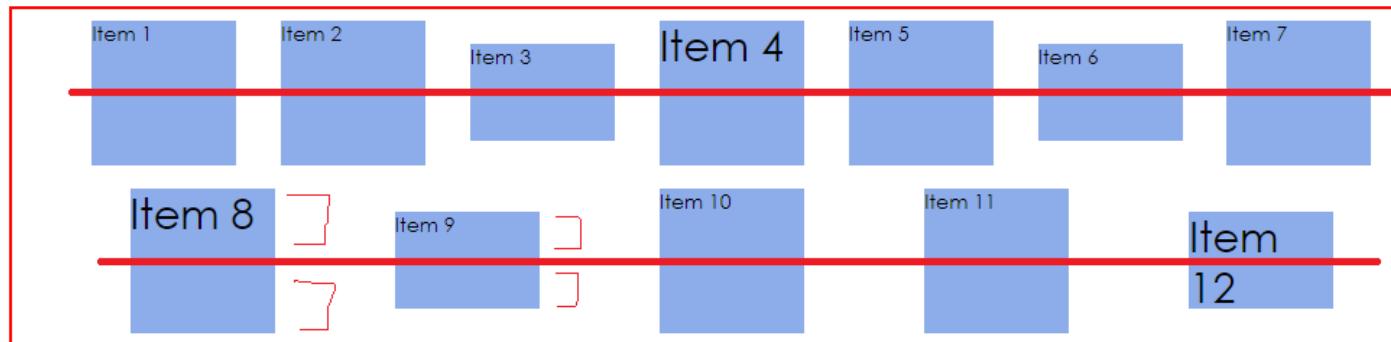
ul li:nth-child(4n) {
  font-size: 1.8rem;
}
```

(rem JE VREDNOST (JEDINICA), KOJA JE RELATIVNA U ODNOSU NA ROOT ELEMENT (BODY, HTML), OVIM CU SE DODATNO POZABAVITI U BUDUCNOSTI)

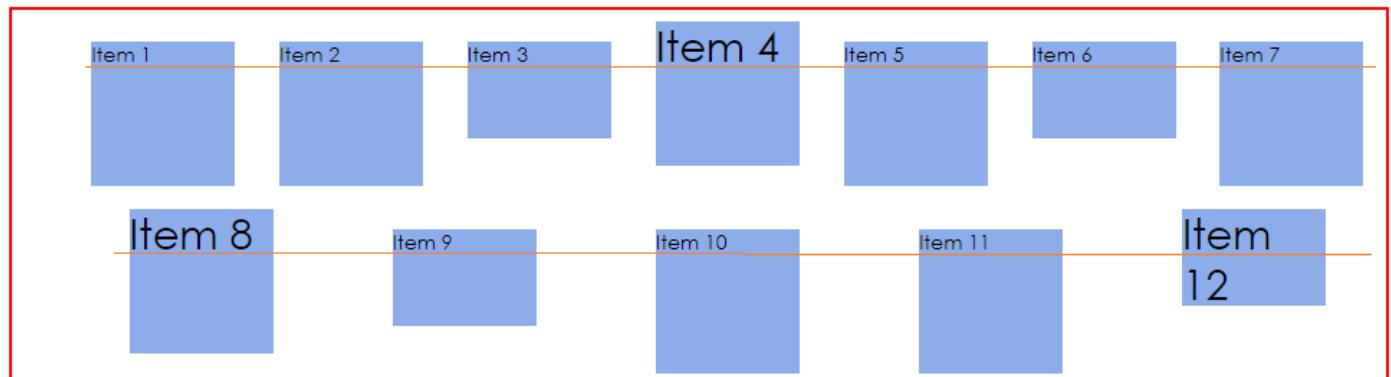


PRVO CU DEFINISATI DA align-items IMA VREDNOST center, PA ONDA baseline JER MISLIM DA CU SA TIM VREDNOSTIMA LAKSE SHVATITI ULOGU OVOG PROPERTIJA, KADA POGLEDAM ELEMENTE, NAKON UPOTREBE, POMENUTOG PROPERTIJA

```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: space-around;  
  
    align-items: center;  
}
```



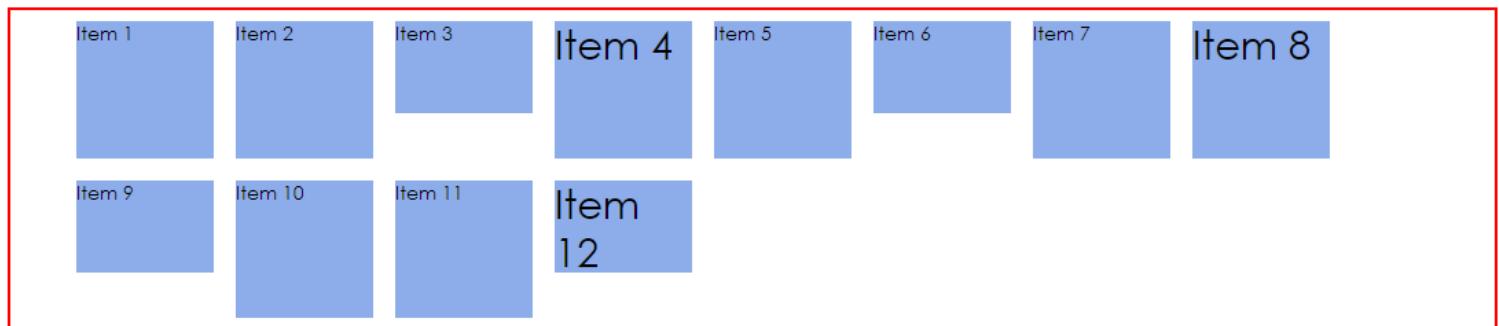
```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: space-around;  
  
    align-items: baseline;  
}
```



A KAKO BI POKAZAO STA, CE ZAPRAVO BITI UCINJENO NA STRANICI, KADA POMENUTI PROPERTI BUDE IMAO VREDNOST flex-start, ILI VREDNOST flex-end; DEFINISACU DA justify-content PROPERTI IMA VREDNOST flex-start

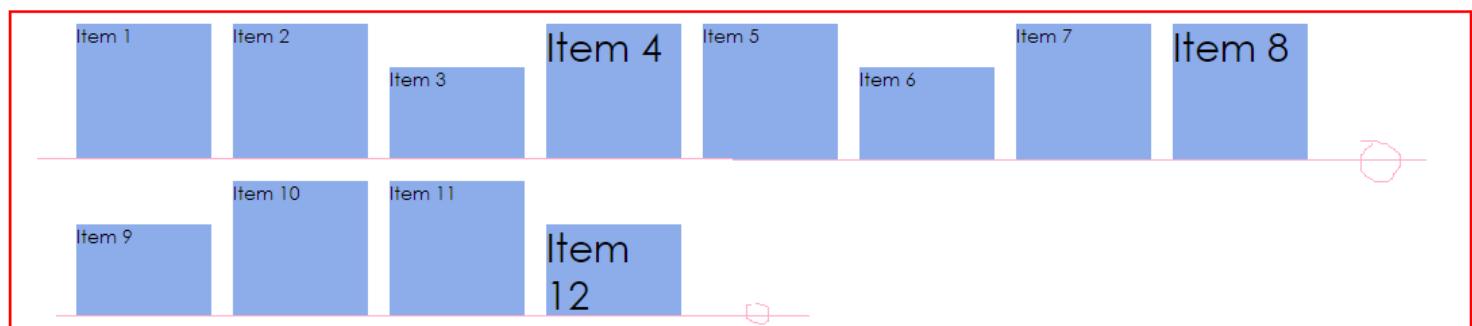
```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: flex-start;  
}
```

KAKO BIH KAO POCKETNU, IMAO SLEDECU SITUACIJU:

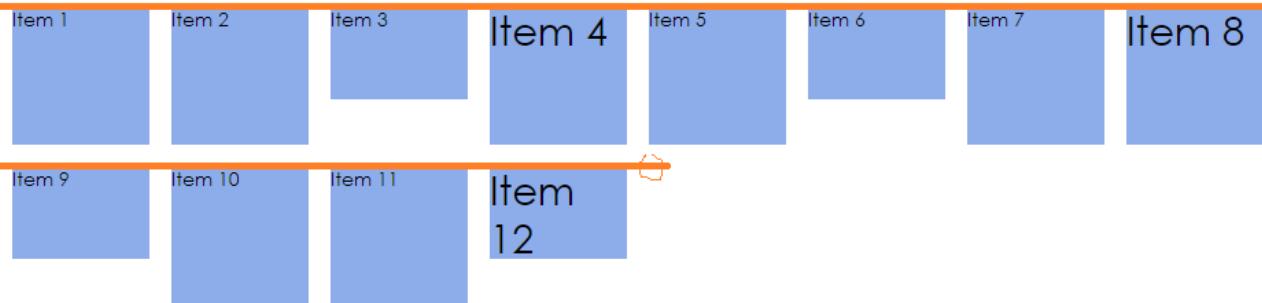


SADA CU DEFINISATI I align-items PROPERTI

```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: flex-start;  
  
    align-items: flex-end;  
}
```



```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: flex-start;  
  
    align-items: flex-start;  
}
```



OSTALO MI JE DA POKAZEM KAKO, JOS JEDNA VREDNOST align-items PROPERTIJA, UTICE NA IZGLED, A TO JE stretch VREDNOST, KOJA JE I DEFAULT VREDNOST

KAKO BI TO NA NAJBOLJI NACIN POKAZAO, OPET CU REDEFINISATI CSS, NA SLEDECI NACIN

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: flex-start;
}

li {
  width: 100px;
  height: 100px;
  background-color: #8cacea;
  margin: 8px;
}

ul li:nth-child(3n) {
  height: 4.2rem;
}

ul li:nth-child(4n) {
  font-size: 1.8rem;
}
```

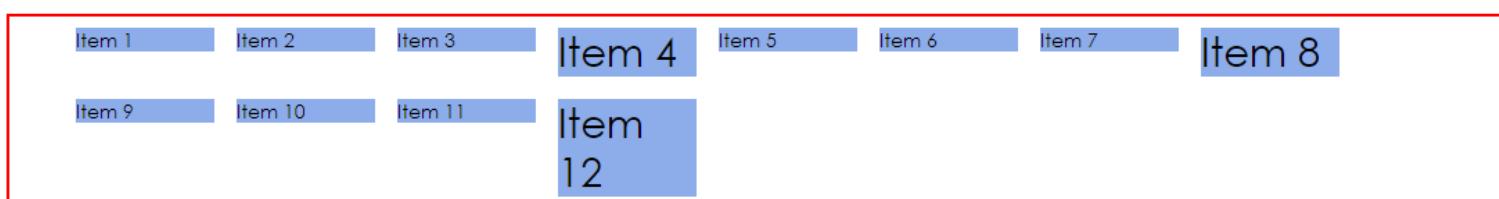


```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: flex-start;
}

li {
  width: 100px;
  background-color: #8cacea;
  margin: 8px;
}

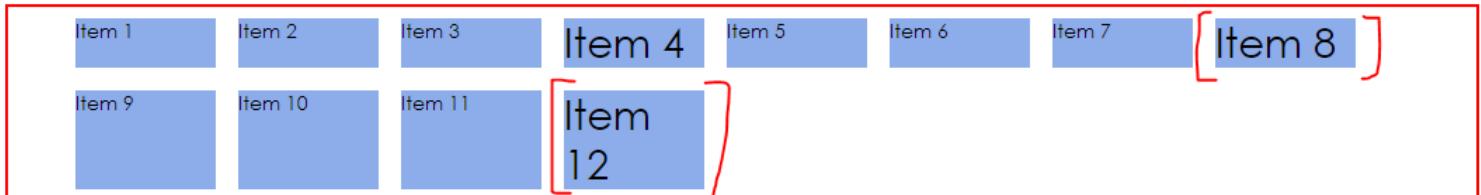
ul li:nth-child(4n) {
  font-size: 1.8rem;
}
```

KAKO BIH IMAO SLEDECU, POCETNU
SITUACIJU:



SADA CU align-items PROPERTIJU DODELITI stretch VREDNOST

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: stretch;
}
```



NA SLICI GORE SAM OZNACIO ONE ELEMENTE, I DO NJIHOVE VISINE SU MORALI DA BUDU STRETCHED ELEMENTI, KOJI SU PORED NJIH U REDOVIMA, JER SU, OZNACENI SU BILI NAJVISI

→ align-content

PROPERTI KOJI, PO DEFINICIJI KONTROLISE, KAKO SU FLEX ITEMI, ALIGN-ED DUZ CROSS AXIS-A, U MULTI-LINE FLEX CONTAINER-U; A MULTI-LINE FLEX CONTAINER, JE ONAJ CONTAINER, KOJEM flex-wrap, MOZE IMATI, VE OSTALE VREDNOSTI, IZUZUZEV no-wrap; ODNOSNO, KADA SE FLEX CONTAINER SASTOJE OD FLEX ITEMA, KOJI SU SMESTENI U VISE REDOVA

VREDNOSTI PROPERTIJA:

flex-start	flex-end	center	stretch (DEFAULT)
			space-between
			space-around

OBJASNJAVA OVOG PROPERTIJA, IMACE SMISLA, SAMO AKO DEFINISEM VISINU FLEX CONTAINER-A; NAIME, OVAKO CE IZGLEDATI STILOVI, PRE NEGO STO BUDEM DEFINISAO align-content PROPERTI

```
ul {
    display: flex;

    border: red solid 2px;
    list-style: none;

    /*flex-direction: row;
    flex-wrap: wrap-reverse;*/
    flex-flow: row wrap;

    justify-content: flex-start;
    align-items: center;
    height: 38rem;
}

li {
    width: 100px;

    background-color: #8cacea;
    margin: 8px;
}

ul li:nth-child(4n) {
    font-size: 1.8rem;
}
```

A OVAKO CE IZGLEDATI FLEX CONTAINER I ITEMI; DAKLE OVAKO CE IZGLEDATI, PRE DEFINISANJA align-content PROPERTIJA

```
Item 1 Item 2 Item 3 Item 4 Item 5
```

```
Item 6 Item 7 Item 8 Item 9 Item 10
```

```
Item 11 Item 12
```

SADA CU DEFINISATI align-content PROPERTI

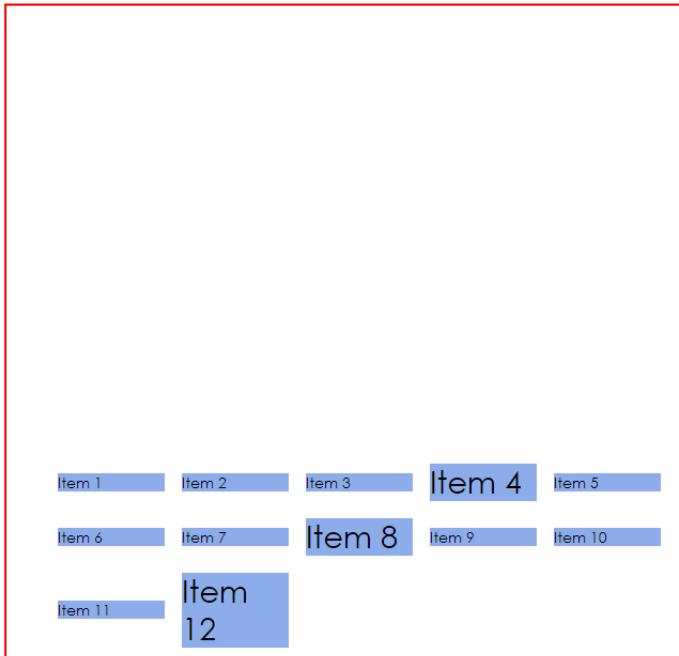
```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
     flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: flex-start;  
  
    align-items: center;  
  
    height: 38rem;  
  
    align-content: flex-start;  
}
```

```
Item 1 Item 2 Item 3 Item 4 Item 5
```

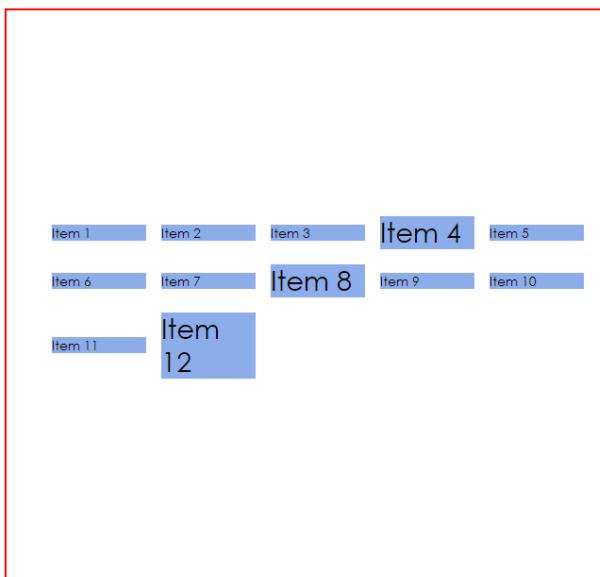
```
Item 6 Item 7 Item 8 Item 9 Item 10
```

```
Item 11 Item 12
```

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: center;
  height: 38rem;
  align-content: flex-end;
}
```

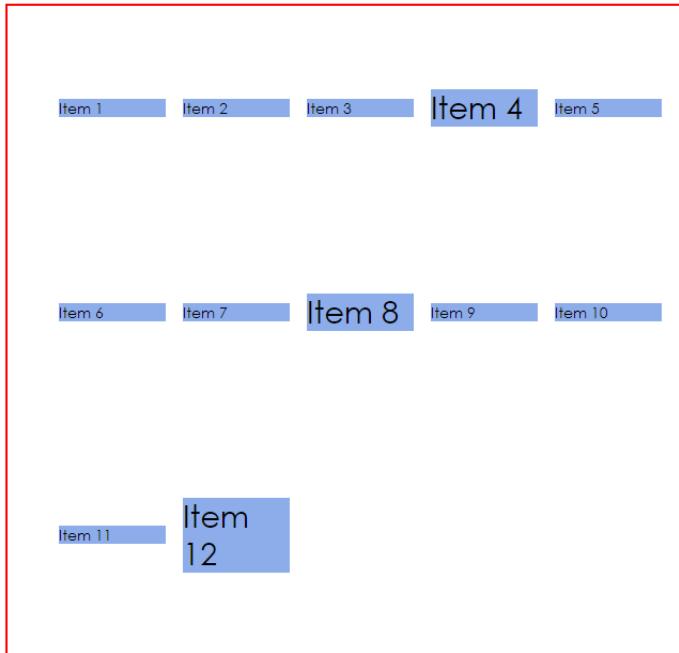


```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: center;
  height: 38rem;
  align-content: center;
}
```

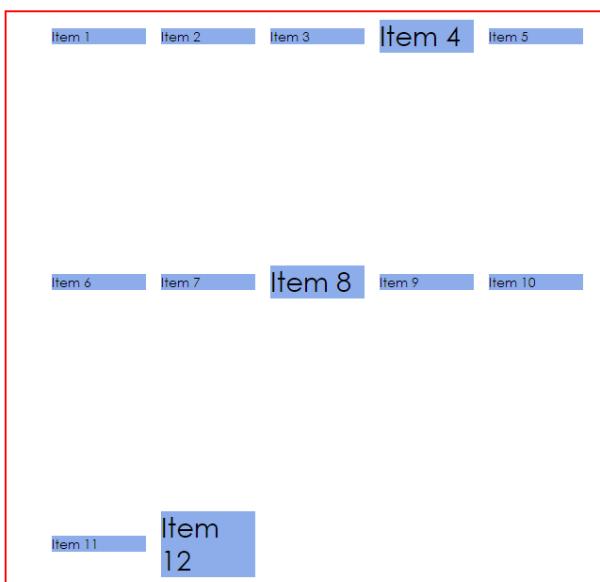


```
ul {  
  display: flex;  
  
  border: red solid 2px;  
  list-style: none;  
  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  
  flex-flow: row wrap;  
  justify-content: flex-start;  
  align-items: center;  
  
  height: 38rem;  
  align-content: stretch;  
}
```

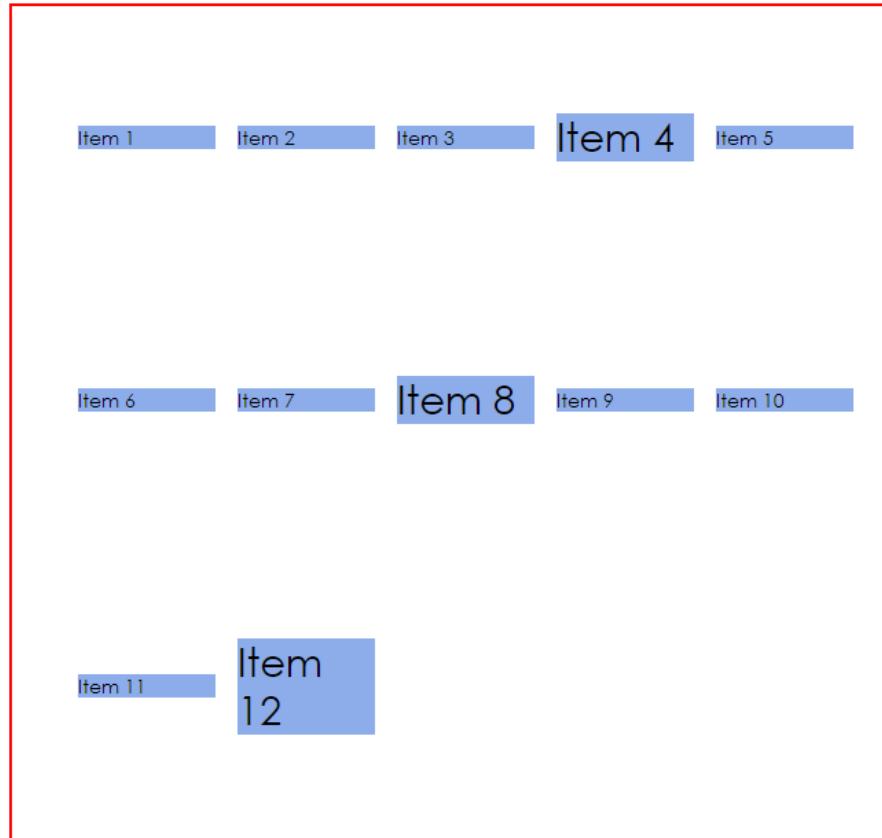
(OVO JE I DEFAULT)



```
ul {  
  display: flex;  
  
  border: red solid 2px;  
  list-style: none;  
  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  
  flex-flow: row wrap;  
  justify-content: flex-start;  
  align-items: center;  
  
  height: 38rem;  
  align-content: space-between;  
}
```



```
ul {  
  display: flex;  
  border: red solid 2px;  
  list-style: none;  
  /*flex-direction: row;  
   flex-wrap: wrap-reverse;*/  
  flex-flow: row wrap;  
  justify-content: flex-start;  
  align-items: center;  
  height: 38rem;  
  align-content: space-around;  
}
```



6 ALIGNMENT PROPERTIES, KOJI SE KORISTE NA FLEX ITEM-U

→→ order

DEFINISE REDOSLED ODREDJENOG/IH FLEX ITEM-A U CONTAINER-U

VREDNOSTI PROPERTIES: INTEGER NUMBER VREDNOSTI (DAKLE, MOGU BITI I NEGATIVNI BROJEVI)

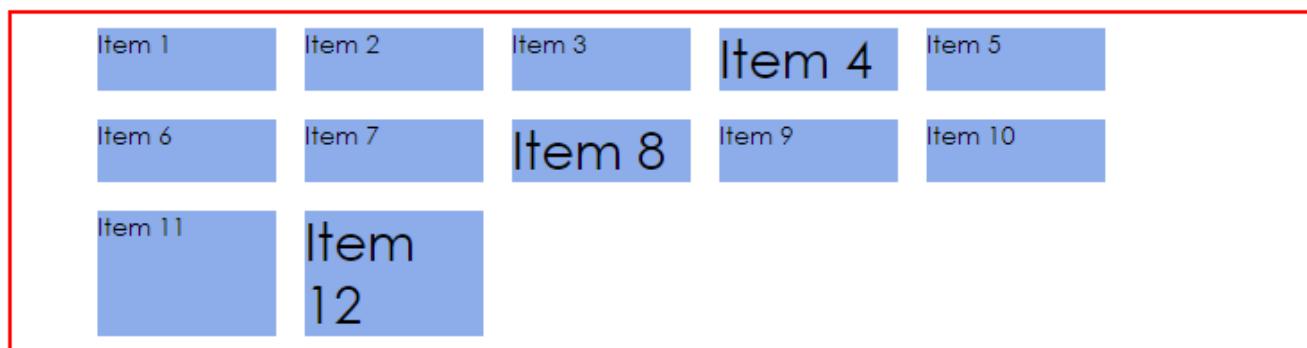
0 (NULA) JESTE DEFAULT VREDNOST

STO ZNACI, DA PRILIKOM DODELJIVANJA VREDNOSTI FLEX ITEMIMA, ITEMI CE REDOSLEDNO BITI SLOZENI OD NAJMANJEG INTEGER BROJA, DO NAJVECEG

NAJBOLJE DA OVAJ ELEMENT, POKAZEM PUTEM PRIMERA

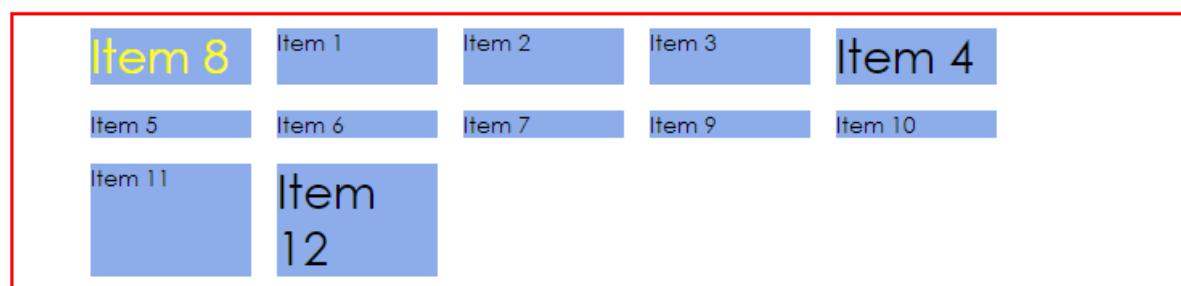
ZA POCETAK CE, MOJI FLEX ITEMI U FLEX CONTAINERU BITI STILIZOVANI OVAKO

```
ul {  
    display: flex;  
    border: red solid 2px;  
    list-style: none;  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
    flex-flow: row wrap;  
    justify-content: flex-start;  
    align-items: stretch;  
    /*height: 38rem;*/  
    align-content: flex-start;  
}  
  
li {  
    width: 100px;  
    background-color: #8cacea;  
    margin: 8px;  
}  
  
ul li:nth-child(4n) {  
    font-size: 1.8rem;  
}
```



A SADA CU, OSMOM FLEX ITEMU ZADATI, NOVU BOJU FONTA, A TAKODJE CU MU DEFINISATI I order PROPERTI

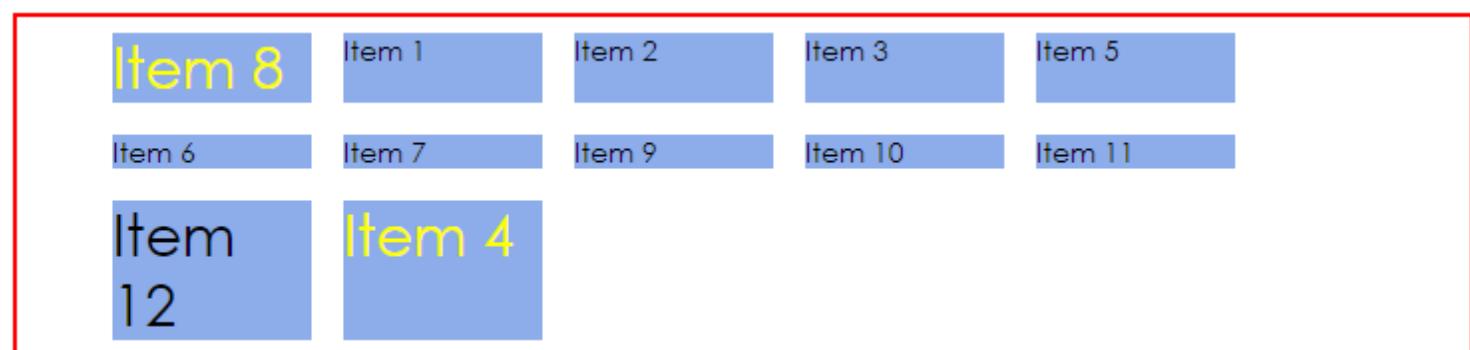
```
ul li:nth-child(8) {  
    color: yellow;  
    order: -1;  
}
```



ZAPAMTI, DA JE order VREDNOST, SVIH OSTALIH ITEM-A, USTVARI 0(NULA), DOK JE OSMOG, USTVARI -1

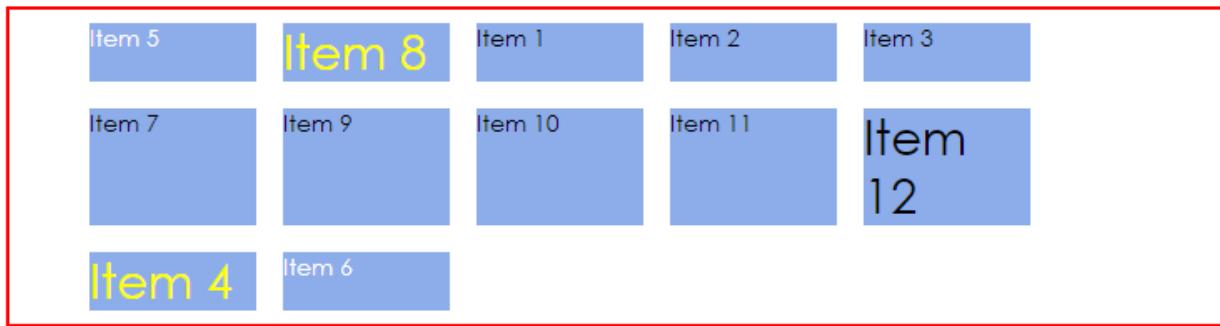
SADA CU DEFINISATI DA VREDNOST order PROPERTIJA, CETVRTOG FLEX ITEM-A, BUDE 1

```
ul li:nth-child(4){  
    color: yellow;  
    order: 1;  
}
```



SADA CU DEFINISATI DA VREDNOST order PROPERTIJA, PETOG FLEX ITEM-A, BUDE -58; A DA VREDNOST order PROPERTIJA, SESTOG FLEX ITEM-A BUDE 68

```
ul li:nth-child(5) {  
    color: white;  
    order: -58;  
}  
  
ul li:nth-child(6) {  
    color: white;  
    order: 68;  
}
```



→→ flex-grow I flex-shrink

flex-grow DEFINISE SPOSOBNOST DA FLEX-ITEM RASTE (PO SIRINI), AKO JE TO MOGUĆE ; ODNOSNO POMENUTIM PROPERTIJEM SE DEFINISE, KOLIKO TREBA DA RASTE FLEX-ITEM; DOK SE flex-shrink PROPERTIJEM DEFINISE SUPROTNO

VREDNOSTI PROPERTIJA flex-grow: UNITLESS VREDNOSTI (DEFAULT JE NULA; NE MOGU NEGATIVNI)

VREDNOSTI PROPERTIJA flex-shrink: UNITLESS VREDNOSTI (DEFAULT JE 1; NE MOGU NEGATIVNI)

NA PRIMER: AKO SVI FLEX ITEMI IMAJU VREDNOST 1, PODESENU ZA POMENUTI PROPERTI, OSTATAK PROSTORA U REDU, BICE JEDNAKO DISTRIBUIRANO IZMEDJU, POMENUTIH FLEX ITEMA; A AKO JEDAN OD NJIH IMA VREDNOST DVA, OSTATAK PROSTORA U REDU BI BIO DISTRIBUIRAN TOM JEDINOM ITEMU, TAKO DA POMENUTI IMA DUPLOVISE TOG PROSTORA OD OSTALIH ITEM-A (“ILI CE BAR POKUSATI DA IMA”)

SMANICU BROJ FLEX ITEMA, KAKO BI LAKSE OBJASNIO POMENUTI PROPERTI (OSTAVICU SAMO JEDAN)

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
  <li>Item 6</li>
  <li>Item 7</li>
  <li>Item 8</li>
  <li>Item 9</li>
  <li>Item 10</li>
  <li>Item 11</li>
  <li>Item 12</li>
</ul>
```

→→

```
<ul>
  <li>Item 1</li>
</ul>
```

Item 1

SADA CU ZADATI DA order PROPERTI, JEDINOGL ITEM-A, IMA VREDNOST 1

```
li {
  width: 100px;
  background-color: #8cacea;
  margin: 8px;

  flex-grow: 1;

}
```

Item 1

FLEX-ITEM SADA "RASTE" DA ZAUZME SVE RASPOLOŽIV PROSTOR. PREKIDAČ JE UKLJUČEN!

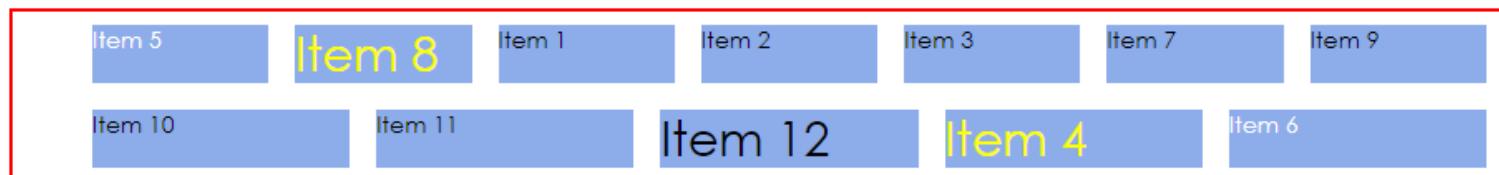
AKO POKUŠAM DA PROMENIM VELIČINU PRETRAŽIVAČA, FLEKSIBILNA STAVKA BI SE TAKOĐE "SMANJILA" DA BI SE PRILAGODILA ŠIRINI NOVOG EKRANA.

ZAŠTO?

PODRAZUMEVANO, flex-shrink JE PODEŠENA NA 1; TO ZNAČI DA JE UKLJUČEN I PREKIDAČ FLEKSIBILNOG SMANJIVANJA!

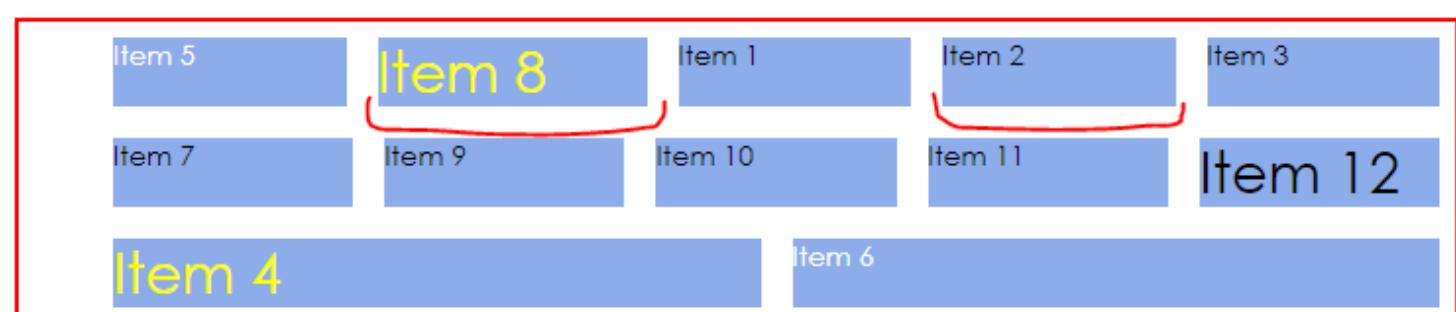
SADA CU DA VRATIM SVE ODUZETE FLEX ITEM-E

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
  <li>Item 6</li>
  <li>Item 7</li>
  <li>Item 8</li>
  <li>Item 9</li>
  <li>Item 10</li>
  <li>Item 11</li>
  <li>Item 12</li>
</ul>
```



PA CU DA ZADAM DA SVI FLEX ITEMI IMAJU PODESENU JEDINICU, ZA POMENUTI PROPERTI, A SAMO CE, OSMI FLEX ITEM IMATI DVA KAO VREDNOST, PA CU SMANJIVATI I POVECAVATI SIRINU WINDOW-A BROWSER-A, KAKO BIH VIDEO, KAK OSE FLEX ITEM-I PONASAJU

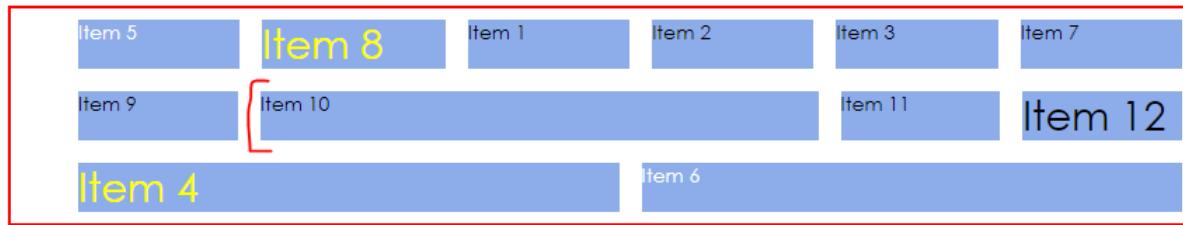
```
ul li:nth-child(8) {
  color: yellow;
  order: -1;
  flex-grow: 2;
}
```



DAKLE, UPOREDNUJEM, KOJA JE SIRINA OSMOG ITEMU I OSTALIH U NJEGOVOM REDU, I KAO STO VIDIM, OSMI ITEM IMA NESTO VECU SIRINU, TOKOM OVOG "FLEXOVANJA"

SADA CU DA ZADAM, DESETOM FLEX ITEMU, NOVU SIRINU

```
ul li:nth-child(10) {  
    width: 24rem;  
}
```



SADA CU ISTOM ITEMU, DEFINISATI flex-shrink PROPERTI, KOJI NECE IMATI 1, KOJA JE DEFAULT VREDNOST, KOJA MU DAJE MOGUCNOST DA SE SHRINK-UJE; VEC CU MU ONEMOGUCITI DA SE SHRINK-UJE, TAKO STO CU DODELITI NULU KAO VREDNOST, POMENUTOM PROPERTIJU



I KAO STO VIDIM SA GORNJE SLIKE, PRILIKOM OVOG SMANJENJA BROWSER-OVOG WINDOW-A, DOGODIO SE OVERFLOW, JER DESTEI FLEX ITEM, NE MOZE VISE DA SE SMANJUJE

DAKLE OPET PONAVLJAM DEFAULT VREDNOST ZA flex-grow JESTE 0; A DEFAULT VREDNOST ZA flex-shrink JESTE 1

→→ flex-basis

POMENUTI PROPERTI DEFINISE INICIJALNU VELICINU ELEMENTA (KADA TO KAZEM, MISLIM NA ONU VELICINU, PRE NEGO STO SU JE flex-grow ILI flex-shrink PODESILI DA STAJE CELOM SIRINOM FLEX CONTAINER-A, ILI NE), PRE NEGO STO JE OSTATAK PROSTORA DISTRIBUIRAN; TO MOZE BITI DUZINA (NA PRIMER, U OVAKVIM JEDINICAMA (FORMATU) : PROCENTI, em-OVI, rem-OVI, pixel-l, I TAKO DALJE)

DEFAULT VREDNOST, POMENUTOG PROPERTIJA JESTE : auto

ZAPAMTI DA AKO ZELIS DA PODESIS POMENUTI PROPERTI NA ZERO BASED VREDNOST, NEMOJ PISATI SAMU NULU; VEC PISI I JEDINICU; NA PRIMER: 0px

SIRINA FLEX ITEM-A SE IZRAČUNAVA "AUTOMATSKI" NA OSNOVU VELIČINE SADRŽINE (TEKST I OSTALO NESTED U FLEX ITEM-U) (I OČIGLEDNO, PLUS PODESENI padding)

MODIFIKOVACU, OPET MOJ PRIMER, TAK OSTO CU SAMO OSTAVITI, JEDAN LIST ITEM

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
  <li>Item 6</li>
  <li>Item 7</li>
  <li>Item 8</li>
  <li>Item 9</li>
  <li>Item 10</li>
  <li>Item 11</li>
  <li>Item 12</li>
</ul>
```



```
      <ul>
        <li>Item 1 of the simple list</li>
      </ul>
```

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: stretch;
  /*height: 38rem;*/
  align-content: flex-start;
}

li {
  min-height: 8rem;
  background-color: #8cacea;
  margin: 8px;
}
```

Item 1 of the simple list

DAKLE, KAO STO SAM I REKAO, SADA JE DEFAULT VREDNOST, POMENUTOG PROPERTIJA, UPRAVO: auto

ZADACU TU VREDNOST, IAKO ZNAM DA JE POMENUTA, ZAISTA DEFAULT VREDNOST

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex-basis: auto;  
}
```

Item 1 of the simple list

SADA CU DEFINISATI, DRUGU VREDNOST, POMENUTOG PROPERTIJA

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex-basis: 18rem;  
}
```

Item 1 of the simple list

NARAVNO, JASNO JE DA AKO SMANJIM PROZOR BROWSCERA, SMANJIVACE SE I FLEX ITEM, KAD SE IVICA (FLEX CONTAINER-A) PRIBLIZI DO NJEGA (“ODNOSNO PREKO NJEGA”)

A MOGAO SAM DEFINISATI I NESTO MANJU VELICINU flex-basis A

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex-basis: 8rem;  
}
```

Item 1 of the
simple list

KAO STO VIDIM SA SLIKE GORE, TEKST JE SADA ZBOG SMANJENJA flex-basis VELICINE, SMESTEN U DVA REDA

→→ flex (SHORTHAND)

OVAJ PROPERTI MI DOZVOLJAVA DA, ODJEDNOM PODESIM VREDNOSTI, ZA flex-grow, flex-shrink I flex-basis

PREPORUCUJE SE KORISCENJE OVOG PROPERTIJA, UMESTO KORISCENJA, POMENUTIH INDIVIDUALNIH PROPERTIJA, ZATO STO IH OVAJ PROPERTI PODESAVA INTELLIGENTLY, KAKO KAZU ZAPAMTI, REDOSLED POSTAVLJANJA VREDNOSTI, POMENUTOG PROPERTIJA; AKRONIM GSB MOZE U TOME POMOCI

AKO PODESIM flex-grow, I flex-shrink VREDNOSTI, A ZABORAVIM DA PODESIM flex-basis; ONDA BI flex-basis DEAFULT-OVALO DO 0 (NULE)

POMENUTO SE NAZIVA APSOLUTNI FLEX (ABSOLUTE FLEX)

A AKO PODESIM, SAMO flex-basis VREDNOST, KOJA NARAVNO IMA I, ODGOVARAJUCU JEDINICU (UNIT) DUZINE; ONDA SE TO NAZIVA RELATIVNI FLEX (RELATIVE FLEX)

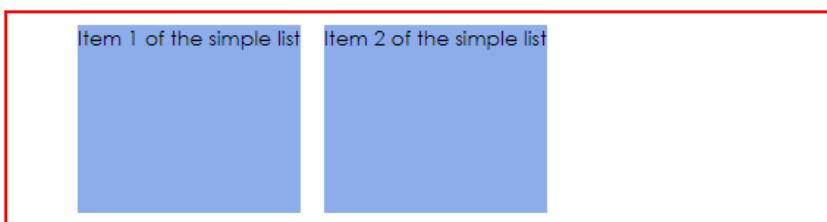
DAKLE, U DRUGOM SLUCAJU BI BILO PODESENO, SAMO flex-basis

O ONOME STA JE SVRHA APSOLUTNOG I RELATIVNOG FLEX-A, GOVORICU KASNIJE

SADA CU POGLEDATI, NEKE KORISNE, flex SHORTHAND VREDNOSTI; ALI PRE TOGA CU DODATI, JOS JEDAN FLEX ITEM, U MOJ FLEX CONTAINER

```
<ul>
  <li>Item 1 of the simple list</li>
  <li>Item 2 of the simple list</li>
</ul>
```

```
li {
  min-height: 8rem;
  background-color: #8cacea;
  margin: 8px;
}
```



a) 0 1 auto

POMENUTE VREDNOSTI SU DEFAULT, I TO BI BILO ISTO, KAO DA SAM NAPISAO SLEDECE:

flex: default

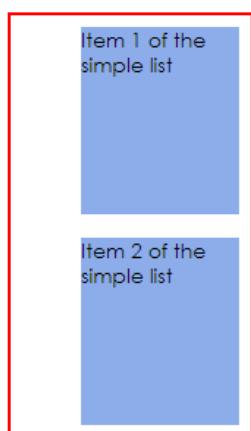
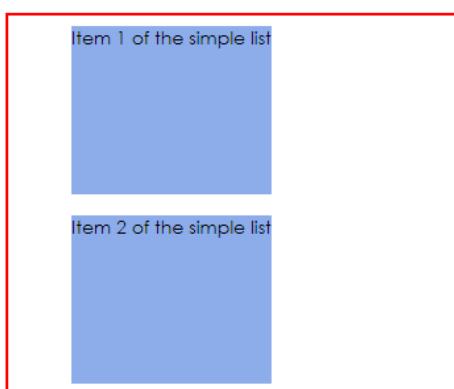
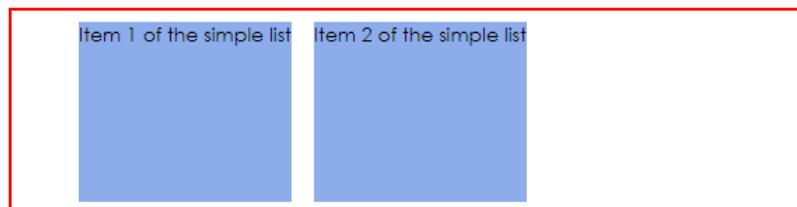
POSTO JE flex-basis PODESENO NA auto, TO ZNACI DA CE INICIJALNA VELICINA FLEX ITEM-A, BITI AUTOMATSKI ODREDJENA, U ODNOSU NA VELICINU SADRZINE (TEKSTA U OVOM SLUCAJU)

flex-grow JE PODESENO NA 0, STO ZNACI DA FLEX ITEM NECE RASTI, PO SIRINI

flex-shrink JE PODESENO NA 1, STO ZNACI DA SE FLEX ITEM NECE SMANJIVATI PO SIRINI, KADA SMANJIM WINDOW BROWSCERA; ALI NE I KADA BROWSEROW WINDOW "NARUSI" INICIJALNU SIRINU FLEX ITEM-A; STO ZNACI DA POMENUTU VREDNOST MOGU INTERPRETIRATI I NA SLEDECİ NACIN:

"SHRINK-UJ FLEX ITEM, SAMO ONADA, KADA JE TO NEOPHODNO"

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex: 0 1 auto;  
}
```



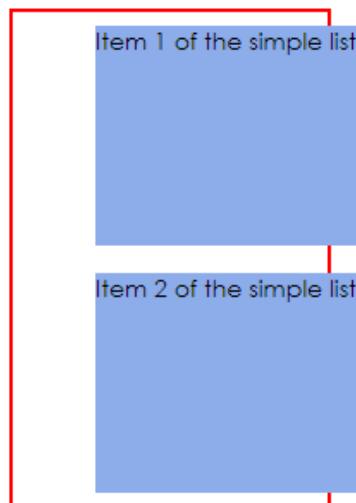
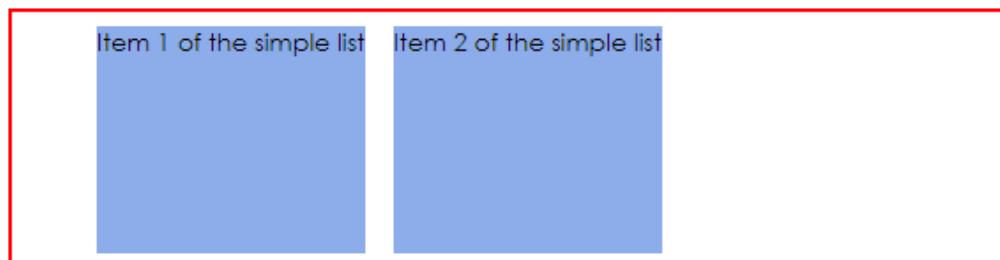
b) 0 0 auto

POMENUTA VREDNOST, JE ISTA, KAO DA SAM NAPISAO SLEDECE:

flex: none

U OVOM SLUCAJU, I GROW I SHRINK SU ISKLJUCENI

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex: 0 0 auto;  
}
```



(NE ZABORAVI DA SAM PODESIO DA JE VREDNOST flex-wrap PROPERTIJA, FLEX CONTAINER-A, PODESENA NA wrap, DA NIJE DOGODIO BI SE DRUGACIJI OVERFLOW)

c) 1 1 auto

POMENUTA VREDNOST, JE ISTA, KAO DA SAM NAPISAO SLEDECE:

flex: auto

U OVOM SLUCAJU SU I GROW I SHRINK UKLJUCENI

```
li {  
    min-height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
  
    flex: 1 1 auto;  
}
```

Item 1 of the simple list

Item 2 of the simple list

Item 1 of the simple list

Item 2 of the simple list

Item 1 of the simple list

Item 2 of the simple list

Item 1 of the simple list

Item 2 of the simple list

POMENUTU VREDNOST MOGU INTERPRETIRATI I NA SLEDECI NACIN:

“AUTOMATSKI COMPUTE-UJ INICIJALNU SIRINU, ALI RASTI DA BI MOGAO DA ZAUZMES DOSTUPAN PREOSTALI PROSTOR, A SHRINK-UJE SE KADA JE TO NEOPHONDO”

d) pozitivan broj

POMENUTA VREDNOST, JE ISTA, KAO DA SAM NAPISAO SLEDECE:

flex: (pozitivan broj) 1 0

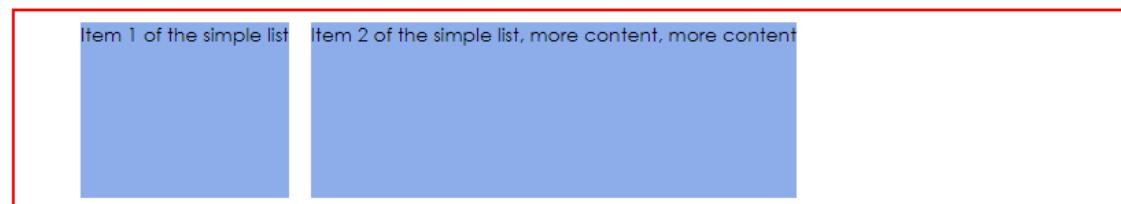
DAKLE, U OVOM SLUCAJU BI VELICINA flex-basis BILA NULA

KAKO BI NA NAJBOLJI NACIN POKAZAO OSOBINE POMENUTE VREDNOSTI, POMENUTOG PROPERTIJA, DEFINISACU DA, DRUGI FLEX ITEM IMA NESTO VISE, NESTED TEKSTA

```
<ul>
  <li>Item 1 of the simple list</li>
  <li>Item 2 of the simple list, more content, more content</li>
</ul>
```

```
ul {
  display: flex;
  border: red solid 2px;
  list-style: none;
  /*flex-direction: row;
  flex-wrap: wrap-reverse;*/
  flex-flow: row wrap;
  justify-content: flex-start;
  align-items: stretch;
  /*height: 38rem;*/
  align-content: flex-start;
}

li {
  min-height: 8rem;
  background-color: #8cacea;
  margin: 8px;
}
```



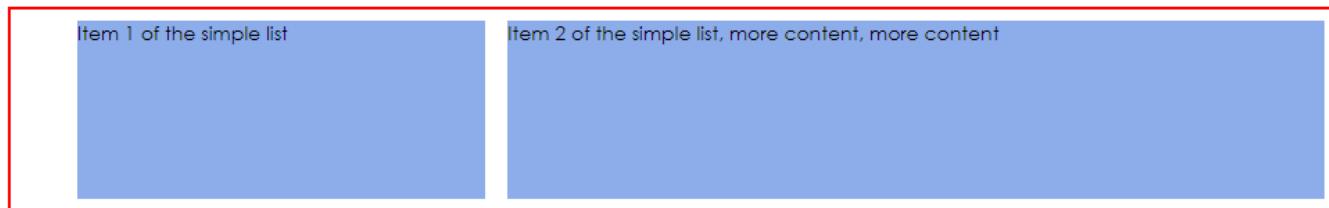
SADA CU SVIM FLEX ITEMIMA DEFINISATI flex SHORTHAND, KOJI CE IMATI VREDNOSTI 1, flex-grow PROPERTIJA, ZATIM VREDNOST 1, flex-shrink PROPERTIJA I auto ZA flex-basis PROPERTI

```
li {
  min-height: 8rem;
  background-color: #8cacea;
  margin: 8px;
  flex: 1 1 0;
}
```



ZATIM CU, SADA DEFINISATI DA flex-grow PROPERTI, SAMO DRUGOG FLEX ITEM-A, IMA VREDNOST 2; STO MOGU UINITI OPET UZ POMOC flex SHORTHAND-A

```
ul li:nth-child(2){  
    flex: 2;  
}
```



JASNO MI JE DA JE DA SE:

OBA PROŠIRUJU DA POPUNE RASPOLOŽIVI PROSTOR, ALI U NEKOJ MERI

EVO KAKO TO FUNKCIONIŠE

OVAJ DRUGI PREUZIMA 2/3 RASPOLOŽIVOG (PREOSTALOG) PROSTORA, DOK JE PRVI ZAUZEZO 1/3
ZNAŠ KAKO SAM STIGAO DO TOGA?

OSNOVNI ODNOS MATEMATIKE: INDIVIDUALNI ODNOS / UKUPNI ODNOS

→ align-self

align-items PROPERTI FLEX CONTAINER-A, OMOGUCAVA KOLEKTIVNI ALIGNMENT DUZ CROSS AXIS-U, SVIH FLEX ITEM-A, NEKOG FLEX CONTAINER-A, DOK align-self PROPERTI, FLEX ITEM-A, OMOGUCAVA ALIGNMENT, INDIVIDUALNIH FLEX ITEM-A

VREDNOSTI PROPERTIJA: auto(DEFAULT) flex-start flex-end center baseline stretch

ZA POCETAK, NEKA U MOM PRIMERU BUDE PRISUTNA SLEDECA STILIZACIJA:

```
ul {  
    display: flex;  
  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
  
    flex-flow: row wrap;  
  
    justify-content: flex-start;  
  
    /*height: 38rem;*/  
  
    min-height: 18rem;  
}  
  
li {  
  
    height: 8rem;  
    background-color: #8cacea;  
    margin: 8px;  
}
```

Item 1 of the simple list Item 2 of the simple list, more content, more content

SADA ZA OBA FLEX ITEM-A DEFINISATI, RAZLICITE VREDNOSTI align-self PROPERTIJA

```
ul li:first-child {  
    align-self: flex-end;  
}  
  
ul li:last-child {  
    align-self: flex-start;  
}
```

Item 2 of the simple list, more content, more content
Item 1 of the simple list

```
ul li:first-child {  
    align-self: baseline;  
}  
  
ul li:last-child {  
    align-self: center;  
}
```

Item 1 of the simple list Item 2 of the simple list, more content, more content

KAKO BIH POKAZAO, STA CE SE POSTICI SLEDECIM VREDNOSTIMA PROPERTIJA, OPET CU MODIFIKOVATI CSS CODE, MOG PRIMERA

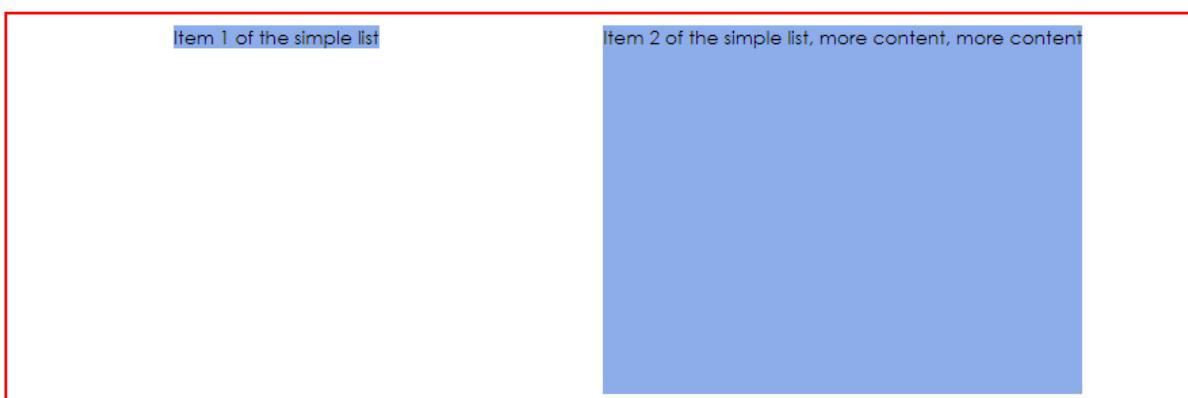
```
ul {  
    display: flex;  
    border: red solid 2px;  
    list-style: none;  
  
    /*flex-direction: row;  
    flex-wrap: wrap-reverse;*/  
    flex-flow: row wrap;  
    justify-content: space-around;  
    /*height: 38rem;*/  
    min-height: 18rem;  
    align-items: flex-start;  
}  
  
li {  
  
    background-color: #8cacea;  
    margin: 8px;  
}
```

KAKO BIH IMAO SLEDECU POCETNU SITUACIJU



SADA CU DEFINISATI, SVAKOM OD FLEX ITEMA RAZLICITE VREDNOST align-self PROPERTIJA

```
ul li:first-child {  
    align-self: auto;  
}  
  
ul li:last-child {  
    align-self: stretch;  
}
```



auto POSTAVLJA VREDNOST IZABRANOG FLEX ITEM-A NA VREDNOST PORAVNANJA RODITELJA ILI SE ISTEZA AKO ELEMENT NEMA RODITELJA

NESTO DETALJNIJE O RELATIVNOM I APSOLUTNOM FLEX ITEM-U

KOJA JE RAZLKA IZMEDJU RELATIVNOG (PODESEN MU JE SAMO flex-basis, DOK flex-grow I flex-shrink NISU PODESENI) I APSOLUTNOG FLEX ITEM-A (SAMO flex-basis NIJE PODESENO)?

GLAVNA RAZLIKA SE OGLEDA U RAZMAKA (SPACING-U), I KAKO SU POMENUTI IZRACUNATI (COMPUTED)

SPACING UNUTAR RELATIVNOG FLEX ITEM-A JE IZRACUNAT, U ODNOSU NA VELICINU SADRZINE (CESTO TEKSTA); APSOLUTNI FLEX-ITEM, ISKLJUCIVO ZASNOVAN (BASED SOLELY) NA "FLEX"-U, A NE NA SADRZINI

KREIRACU NOVI MARKUP

```
<ul>
  <li>
    Ovo je, samo neki random tekst, koji treba da podupre (buttress)
    smisao (point) onoga sto se objasnjava.
    Jos malo random teksta da butress the point.
  </li>
  <li>Ovo je samo kraci random tekst.</li>
</ul>
```

STILIZOVACU GA NA SLEDECİ NACIN

```
ul {
  border: red solid 1px;
  padding: 0;
  list-style: none;
  background-color: #e8e8e9;

  display: flex;      /*flexbox aktiviran*/
}

ul li {
  border: red solid 2px;
  background-color: #8cacea;
  margin: 1em;

  flex: auto;        /*Zaplamti da je ovo isto kao da sa napisao 1 1 auto*/
}
```

FLEX CONTAINER I NJEGOVI ITEMI CE IZGLEDATI OVAKO

Ovo je, samo neki random tekst, koji treba da podupre (buttress) smisao (point) onoga sto se objasnjava. Jos malo random teksta da butress the point.

Ovo je samo kraci random tekst.

DAKLE, PONAVLJAM OPET:

flex: 1 1 auto

JE ISTO, KAO I

flex: auto

STO ZNACI DA SU INICIJALNE ŠIRINE FLEX ITEM-A, AUTOMATSKI COMPUTED (flex-basis: auto), I ŠIRINE RASTU, KAKO BI UPOTPUNILE (TO FIT), DOSTUPNI PROSTOR (flex-grow: 1)

KADA FLEX ITEM-I, IMAJU AUTOMATSKI IZRACUNATE ŠIRINE, TO ZNACI DA SE NJIHOVE ŠIRINE BAZIRAJU NA, NJIHOVOJ SADRZINI

FLEX ITEM-I, U GORE NAVEDENOM PRIMERU NEMAJU SADRŽAJE ISTE VELIČINE; STOGA, VELIČINE FLEX-ITEM-A BI BILE NEJEDNAKE

POŠTO POJEDINAČNE ŠIRINE NISU BILE JEDNAKE NA PRVOM MESTU (ZASNOVANO JE NA SADRŽAJU); KADA ITEM-I RASTU, ŠIRINE OSTAJU NEJEDNAKE

FLEX ITEM-I IZ GORNJEG PRIMERA JESU **RELATIVNI FLEX ITEM-I**

ONO STO ZELIM DA URADIM, JESTE DA TE FLEX ITEM-E, UCINIM APSOLUTNIM; STO ZNACI DA TADA, NJIHOVE ŠIRINE NE TREBA DA ZAVISE OD VELICINE SADRZINE (NAJCESCE TEKSTA) VEC OD "FLEX-A"

JEDAN "ONE-LINER" CE NAPRAVITI MAGIJU ("ONE-LINER" DOES THE MAGIC)

```
ul li {  
    border: red solid 2px;  
    background-color: #8cacea;  
    margin: 1em;  
  
    flex: 1; /*ovo je isto kao da sam napisao 1 1 0 (U PROSLOJ LEKCIJI  
    TO JE NAZVANO SLEDECIM:  
    flex: "pozitivan broj")*/  
}
```

Ovo je, samo neki random tekst, koj treba da podupre (buttress) smisao (point) onoga sto se objasnjava. Jos malo random teksta da buttress the point.

Ovo je samo kraci random tekst.

DA LI VIDIS DA OBA FLEX ITEMA IMAJU ISTE ŠIRINE OVOG PUTA?

POČETNE ŠIRINE FLEX ITEM-A, SU flex-basis: 0, A ZATIM "RASTU" KAKO BI ODGOVARALE RASPOLOŽIVOM PROSTORU

KADA POSTOJE DVA ILI VIŠE FLEX ITEM-A, KOJI IMAJU VREDNOSTI BAZIRANE NA NULI, ONI DIJELE RAZMAK DOSTUPAN NA OSNOVU flex-grow VRIJEDNOSTI

O TOME JE BILO RECI I RANIJE

SADA SE ŠIRINE NE RAČUNAJU NA OSNOVU VELIČINE SADRŽAJA; ŠIRINE SE ZASNIVAJU NA NAVEDENOJ VRIJEDNOSTI FLEX-A

DAKLE, SHVATIO SI TO. JEL TAKO?

**APSOLUTNI FLEX ITEM-I, IMAJU SVOJE ŠIRINE ZASNOVANE ISKLJUČIVO NA FLEX-U, DOK
RELATIVNI FLEX ITEM-I IMAJU ŠIRINU ZASNOVANU NA VELIČINI SADRŽAJA**

AUTO-MARGIN ALIGNMENT

CUVAJ SE margin: auto ALIGNMENT-A NA FLEX ITEM-IMA

KADA KORISTIM margin: auto, NA FLEX ITEM-IMA, STVARI MOGU IZGLEDATI PRILIČNO ČUDNO MORAM RAZUMJETI ŠTA SE DEŠAVA; TO MOŽE REZULTIRATI NEOČEKIVANIM REZULTATIMA, ALI ĆU TO OBJASNITI

KADA KORISTIM margin: auto NA FLEX ITEM-U, PRAVAC, ODNOSNO SMER (LIJEVO, DESNO ILI OBOJE) KOJI IMA VREDNOST auto, CE ZAUZETI, SVE RASPOLOŽIVE PRAZNE PROSTORE

TO JE TEŠKO SHVATITI; EVO ŠTA MISLIM:

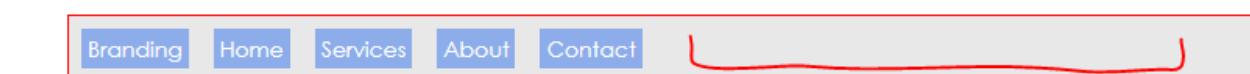
RAZMISLI O LINIJI ZA NAVIGACIJU KOJA MARKED-UP I STYLED ISPOD:

```
ul {  
    border: red solid 1px;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
}  
  
li {  
    background-color: #8cacea;  
    margin: 8px;  
    padding: 4px;  
    color: #fff;  
  
    flex: 0 0 auto;  
}
```



POSTOJI NEKOLIKO STVARI, KOJE SE TREBAJU ZABELEZITI:

- 1) flex-grow, JE PODESEN NA NULU; I TO OBJASNJAVA ZASTO FLEX ITEMI NE RASTU
- 2) FLEX ITEMI SU ALIGNED UZ POCETAK MAIN AXIS-A, JER JE TAKO PODRAZUMEVANO (DEFAULT BEHAVIOUR), ODNOSNO justify-content FLEX CONTAINER-A, PO DEFAULT-U IMA flex-start VREDNOST
- 3) ZAHVALJUJUCI (OWING) TOME STO SU FLEX ITEM-I, ALIGNED UZ POCETAK MAIN AXIS-A, ODREĐENI DODATNI PROSTOR (EXTRA SPACE), JE OSTAVLJEN NA DESNO J STRANI FLEX CONTAINER-A



SADA CU DEFINISATI margin-right: auto , ZA PRVI FLEX ITEM (Branding); PA CU VIDETI STA CE SE, Onda dogoditi

```
ul li:nth-child(1){  
    margin-right: auto;  
}
```

Branding

Home

Services

About

Contact

STA SE DOGODILO?

PA, POSTOJECI EXTRA PROSTOR, JE SADA DITRIBUIRAN, SAMO DESNO OD PRVOG FLEX ITEM-A,

SEĆAŠ LI SE ŠTA JE RECENO RANIJE? KADA KORISTITIM margin: auto NA FLEX ITEM-U, SMER (LIJEVO, DESNO ILI OBOJE) KOJI IMA VREDNOST auto ĆE PREUZETI SAV RASPOLOŽIVI PRAZAN PROSTOR; I TO SE NE ZAVRŠAVA SAMO SA JEDNOM STRANOM; ŠTA AKO ŽELIM margin-OV, auto ALIGNMENT, SA OBE STRANE FLEX ITEM-A?

```
ul li:nth-child(1){  
    margin-right: auto;  
    margin-left: auto;  
}
```

Branding

Home

Services

About

Contact

SADA SE PROSTOR DISTRIBUTUIRAO SA OBE STRANE, PRVOG FLEX-ITEM-A

DAKLE, DA LI POSTOJI KOMPROMIS (TRADE-OFF) U VEZI COOL AUTO-MARGIN ALIGNMENT-A?
IZGLEDA DA POSTOJI JEDAN; TO MOŽE BITI IZVOR FRUSTRACIJE AKO NE OBRATIM PAŽNU

KADA KORISTIM AUTO-MARIGN ALIGNMENT NA FLEX ITEM-U, justify-content PROPERTI, VIŠE NE FUNKCIONIŠE NA FLEX ITEM-IMA; JEDNOSTAVNO NE RADI

NA PRIMJER, POSTAVLJANJEM DRUGAČIJE OPCIJE PORAVNANJA NA GORE NAVEDENE FLEX ITEM-E, POMOĆU SVOJSTVA justify-content A, NEMA UTICAJA NA IZGLED

```
li {  
    background-color: #8cacea;  
    margin: 8px;  
    padding: 4px;  
    color: #fff;  
  
    flex: 0 0 auto;  
    justify-content: flex-end;  
}
```

Branding

Home

Services

About

Contact

SLUČAJEVI PRAKTIČNE UPOTREBE

NAVIGATIONS SU VEOMA VELIKI DEO SVAKOG VEB SAJTA ILI APLIKACIJE; GOTOVO SVAKA VEB STRANICA NA PLANETI IMA NEKU VRSTU NAVIGACIONOG SISTEMA

POGLEDAJ OVE POPULARNE SAJTOVE I KAKAV IMAHU APPROACH, SVOJIM NAVIGACIONIM SISTEMIMA; DA LI VIDIS KAKO TI FLEXBOX MOŽE POMOĆI DA IZGRADIS OVE LAYOUT-E EFIKASNIJE? POGLEDAJ BLIŽE KAKO BI VIDEO KAKO SVOJSTVO AUTO-MARGIN-A MOŽE BITI VRLO ZGODNO

1. Bootstrapped Navigation



2. AirBnB desktop Navigation



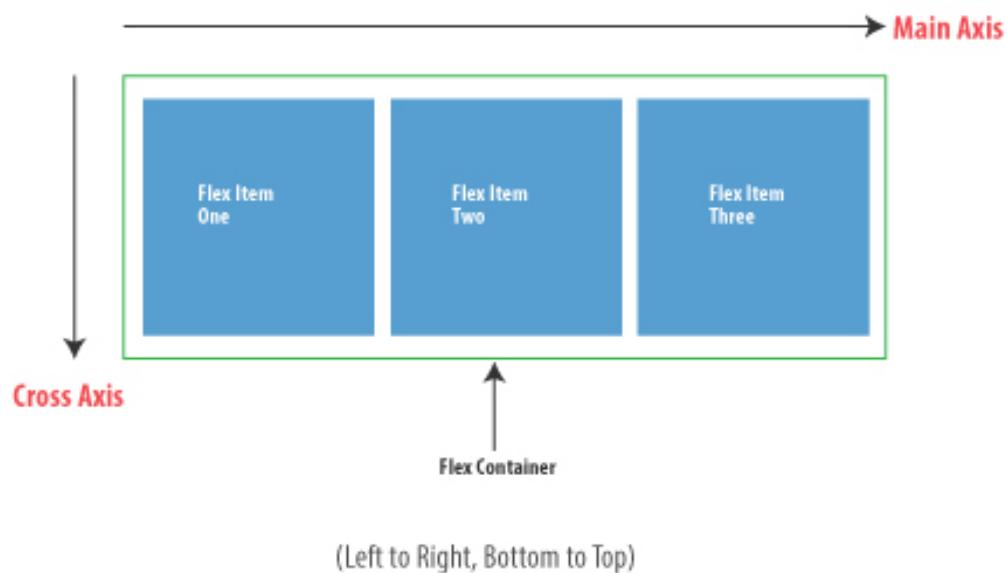
3. Twitter desktop Navigation



STA SE DOGADJA, KADA PROMENIM flex-direction

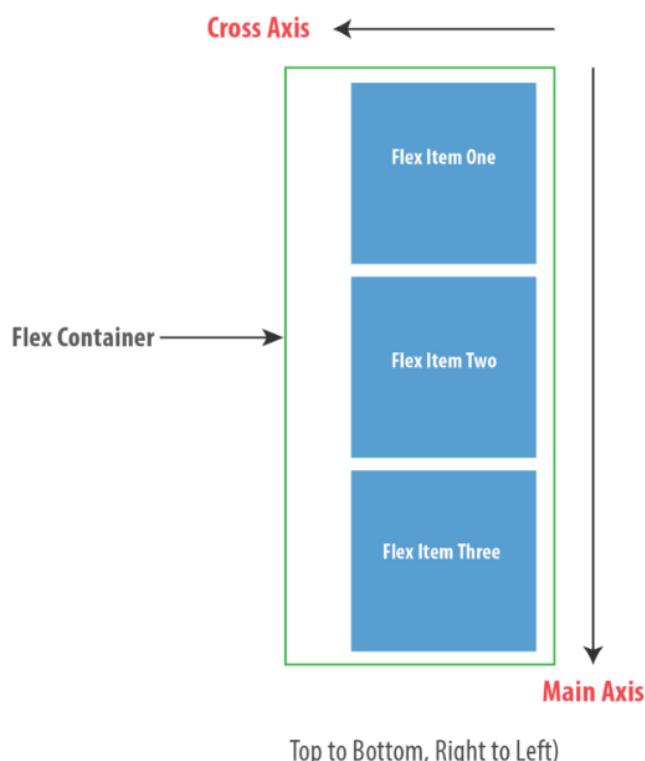
KADA SAM ZAPOČEO SA UČENJEM FLEXBOX MODELA, OVAJ DEO JE BIO NAJVISE ZBUNJUJUĆI KLADIM SE DA, PUNO NOVOPRIDOŠLIH OSOBA U SVIJETU FLEXBOX-A, ISTO TAKO MISLI

SEĆAS LI SE KADA SAM GOVORIO O DEFAULT MAIN I CROSS OSI U PRAVCU; DA SU ONE INTERPRETIRANE, KAO "OD LIJEVE KA DESNOJ" (MAIN) I "ODOZGO PREMA DOLE" (CROSS)



PA, TO MOŽES I DA PROMENIS

ŠTO JE UPRAVO ONO ŠTO SE DEŠAVA KADA KORISTIM `flex-direction: column` (ZAPAMTITE DA JE PODRAZUMEVANA VRIJEDNOST `flex-direction: row`); KADA OVO URADITE, GLAVNA I UNAKRSNA OSA SE MENJAJU KAO ŠTO JE PRIKAZANO ISPOD:



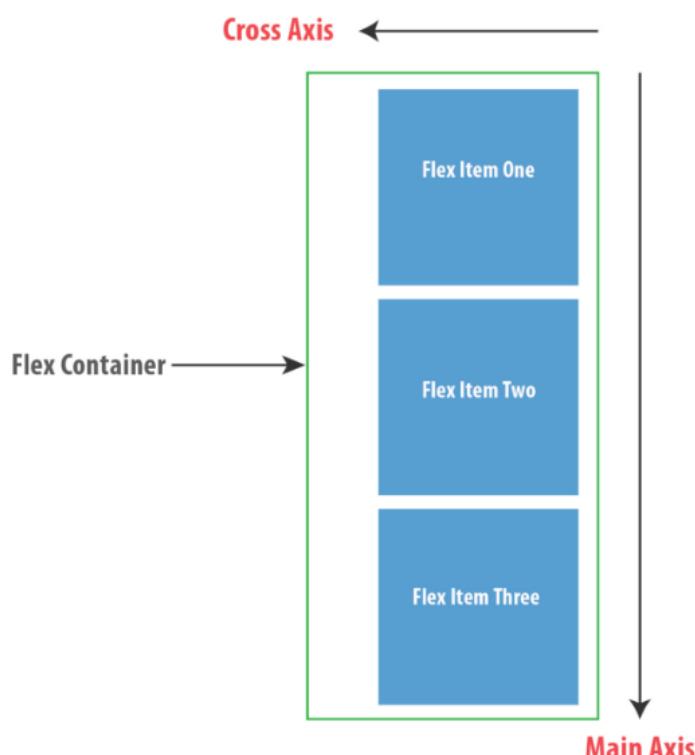
AKO SI, IKADA NAPISAO BILO KOJI TEKST NA ENGLESKOM, ONDA VEĆ ZNAS DA SE, TAJ JEZIK PISE, OD LEVE KA DESNOJ I OD VRHA KA DOLE; TO JE, PODJEDNAKO I PRAVAC ZA, PODRAZUMEVANU MAIN I CROSS OSU FLEXBOX-A

MEĐUTIM, PRI PROMENI flex-direction NA column, VIŠE NE SLEDI OBRAZAC "ENGLESKOG JEZIKA", VEĆ JAPANSKOG; O DA, JAPANSKOG

AKO SI NAPISAO, BILO KOJI TEKST NA JAPANSKOM JEZIKU, ONDA ĆE TI TO BITI POZNATO (JA NIKAD NISAM NAPISAO TEKST NA JAPANSKOM JEZIKU); JAPANSKI TEKST SE PISE OD VRHA KA DNU I OD DESNE KA LEVOJ STRANI

RAZMISLI O TOME NA TRENTAK; NIJE TAKO ČUDNO, A?

TO OBJAŠJAVA ZAŠTO OVO MOŽE BITI MALO ZBUNJUJUĆE ZA ONE, KOJI PISU NA ENGLESKOM



Top to Bottom, Right to Left)

SADA CU DEFINISATI FLEX CONTAINER, I U NJEMU TRI FLEX ITEM-A

```
ul {  
    border: red solid 1px;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
}  
  
li {  
    color: #fff;  
    background-color: #8cacea;  
    margin: 8px;  
    padding: 4px;  
  
    flex: 0 0 auto;  
}  
  
<ul>  
    <li>Item 1</li>  
    <li>Item 2</li>  
    <li>Item 3</li>  
</ul>
```

```
Item 1 Item 2 Item 3
```

SADA CU DA PODESIM flex-direction PROPERTI, NA column VREDNOST

```
ul {  
    border: red solid 1px;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
  
    flex-direction: column;  
}
```

Item 1

Item 2

Item 3

I ŠTA SE DESILO? "TEKST" JE SADA NAPISAN U JAPANSKOM STILU - OD VRHA PREMA DOLJE (MAIN AXIS JE SADA, OD VRHA PREMA DOLE)

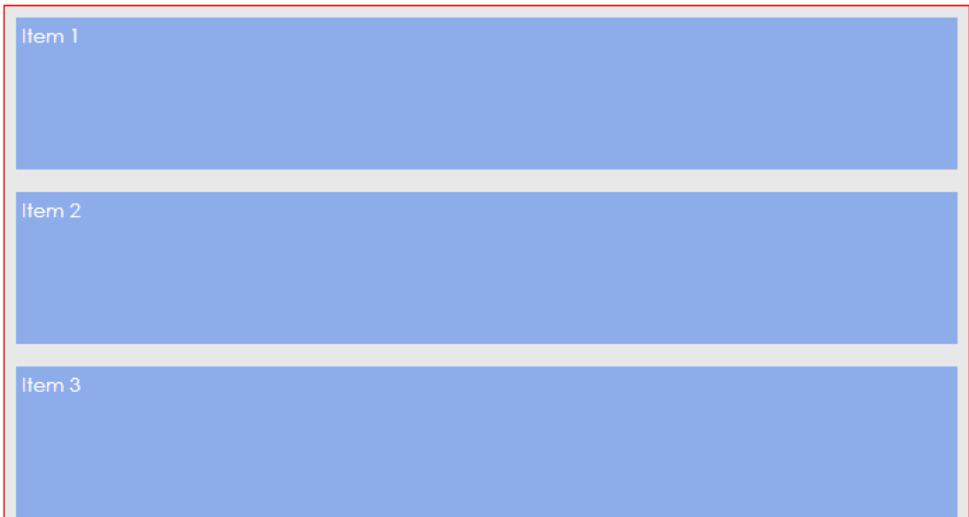
IMA NEŠTO ŠTO MOŽE DA TI BUDE SMEŠNO, A TO CU ISTACI

VIDIS DA ŠIRINA ITEM-A ISPUNJAVA PROSTOR, ZAR NE?

KADA SAM TO RANIJE ZELEO DA PROMENIM, KORISTIO BI flex-grow I/ILI flex-basis PROPERTIJE

DA VIDIM KAKO ONI UTIČU NA, NOVI IZGLED ELEMENATA; PRVO flex-basis PROPERTI

```
ul {  
  
    border: red solid 1px;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
  
    flex-direction: column;  
}  
  
li {  
  
    color: #fff;  
    background-color: #8cacea;  
    margin: 8px;  
    padding: 4px;  
  
    flex: 0 0 auto;  
  
    flex-basis: 100px;  
}
```



KAO STO VIDIM, POMENUTI PROPERTI JE UTICAO NA VISINU, ALI NE I NA SIRINU

RANIJE SAM REKAO DA SVOJSTVO flex-basis A DEFINIŠE POČETNU ŠIRINU SVAKOG FLEX ITEM-A; POGREŠIO SAM - ILI BOLJE RECENO, RAZMIŠLJAO SAM NA "ENGLESKOM"; HAJDE DA SE MALO PREBACIM NA JAPANSKI

DAKLE, TO NE MORA UVEK BITI "ŠIRINA"

NAKON flex-direction PROMENE, IMAJ NA UMU DA SVAKO SVOJSTVO KOJE JE UTICALO NA MAIN AXIS, SADA UTIČE NA NOVU MAIN AXIS

PROPERTI KAO ŠTO JE flex-basis, KOJA JE UTICALA NA ŠIRINU FLEX ITEM-A, DUŽ MAIN AXIS-A, SADA UTIČE NA VISINU, A NE ŠIRINU. PRAVAC JE PROMENJEN

DAKLE, ČAK I DA SAM KORISTIO flex-grow PROPERTI, TO BI UTICALO I NA VISINU!

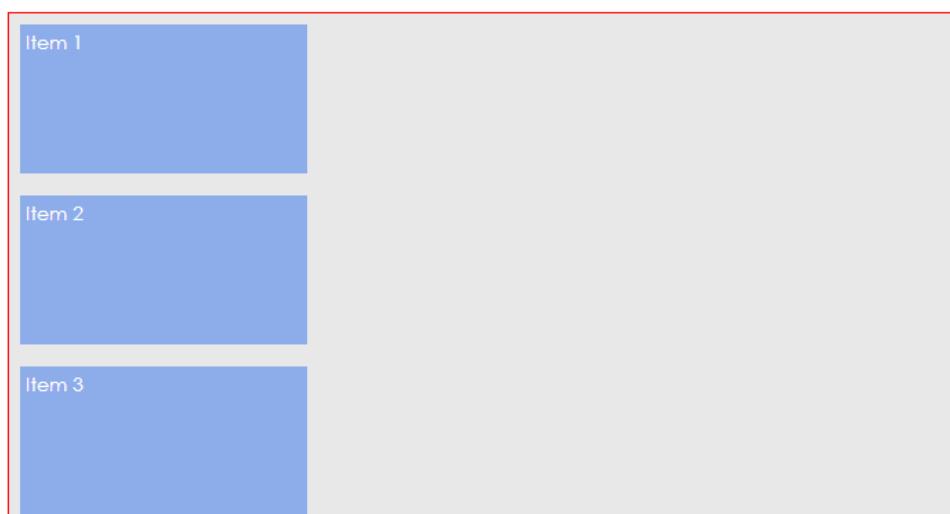
U SUŠTINI, SVAKI FLEX PROPERTI, KOJI JE RUKOVODIO (OPERATED), NA HORIZONTALNOJ OSI (TADA MAIN, ODNOSNO GLAVNA OSA) SADA RADI NA NOVOJ GLAVNOJ, ODNOSNO MAIN OSI; A TO JE VERTIKALNO

DAKLE, DOGODILA SE SAMO PROMENA U POGLEDU PRAVCA

DA VIDIM JOŠ JEDAN PRIMER; OBEĆAVAM DA ĆU POSLE TOG PRIMERA, SVE BOLJE RAZUMETI

SMANICU SIRINU FLEX ITEM-A I VIDEĆU DA ONI VISE NECE ZAUZIMATI, CEO PROSTOR

```
ul {  
    border: 1px solid red;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
    flex-direction: column;  
}  
  
li {  
  
    color: #fff;  
    background-color: #8cacea;  
    margin: 8px;  
    padding: 4px;  
  
    flex: 0 0 auto;  
    flex-basis: 100px;  
    width: 200px;  
}
```



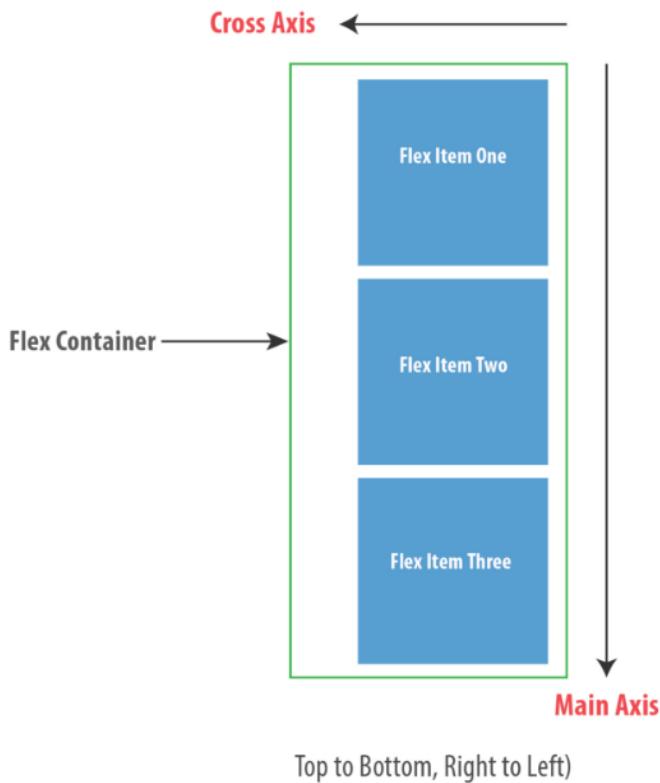
ŠTA AKO ŽELITE DA PREMESTITIM FLEX ITEME NA CENTAR EKRANA?

NA ENGLESKOM JEZIKU, NA KOJI NAČIN SAM DO SADA RADIO SA FLEX CONTAINER-IMA, TO BI ZNAČILO "POMJERANJE FLEX ITEM-A NA CENTAR MAIN-AXIS-A". DAKLE, KORISTIO BIH:

justify-content: center

AKO URADIM ISTO, I SADA, TO NECE FUNKCIONISATI

POŠTO SE SMER (ODNOSNO PRAVAC OSA) PROMENIO, CENTAR JE DUŽ CROSS AXIS-A; POGLEDAJTE PONOVO:



ZATO RAZMISLI NA JAPANSKOM; MAIN AXIS JE OD VRHA PREMA DOLJE, ZATO MI ONA NE TREBA CROSS-AXIS JE OD LEVA NA DESNO; ZVUČI KAO DA MI ONA TREBA

DAKLE, POTREBNO JE DA "PREMJESTIS FLEX ITEM-E, OD POČETKA CROSS AXIS-A, DO CENTRA CROSS AXIS-A"

DA LI SE SECAM IJEDNOG PROPERTIJA, SA KOJIM MOGU POSTICI POMENUTO; DA TO JE align-items

ZAPAMTI DA align-item PROPERTI RUKUJE SA ALIGNMENT-OM NA CROSS AXIS-U; TAKO DA BIH POMENUTE FLEX ITEME POMERIO NA CENTAR, MORAM URADITI SLEDECE

```
ul {  
    border: red solid 1px;  
    background-color: #e8e8e9;  
    list-style: none;  
    padding: 0;  
  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}
```



MOŽE SE ZBUNITI, ZNAM; ZATO OVU LEKCIJU OPET TREBAM PRECI
MNOGI TUTORIJALI I CSS KNJIGE, ZAOBISLE SU UPRAVO OVAJ DEO

**NEKI VAZNI LINKOVI NA KOJIMA SE NALAZI INFORMACIJE O TOME,
STA JE TO FLEXBOX, KONRETNO RESIO, A STO JE DO NJEGOVE POJAVE
BILO PROBLEMATICNO, ZATIM SVI POZNATI BUG-OVI U POGLEDU
FLEXBOXA SA SU POBROJANI I OBJASNJENI NA JEDNOJ STRANICI**

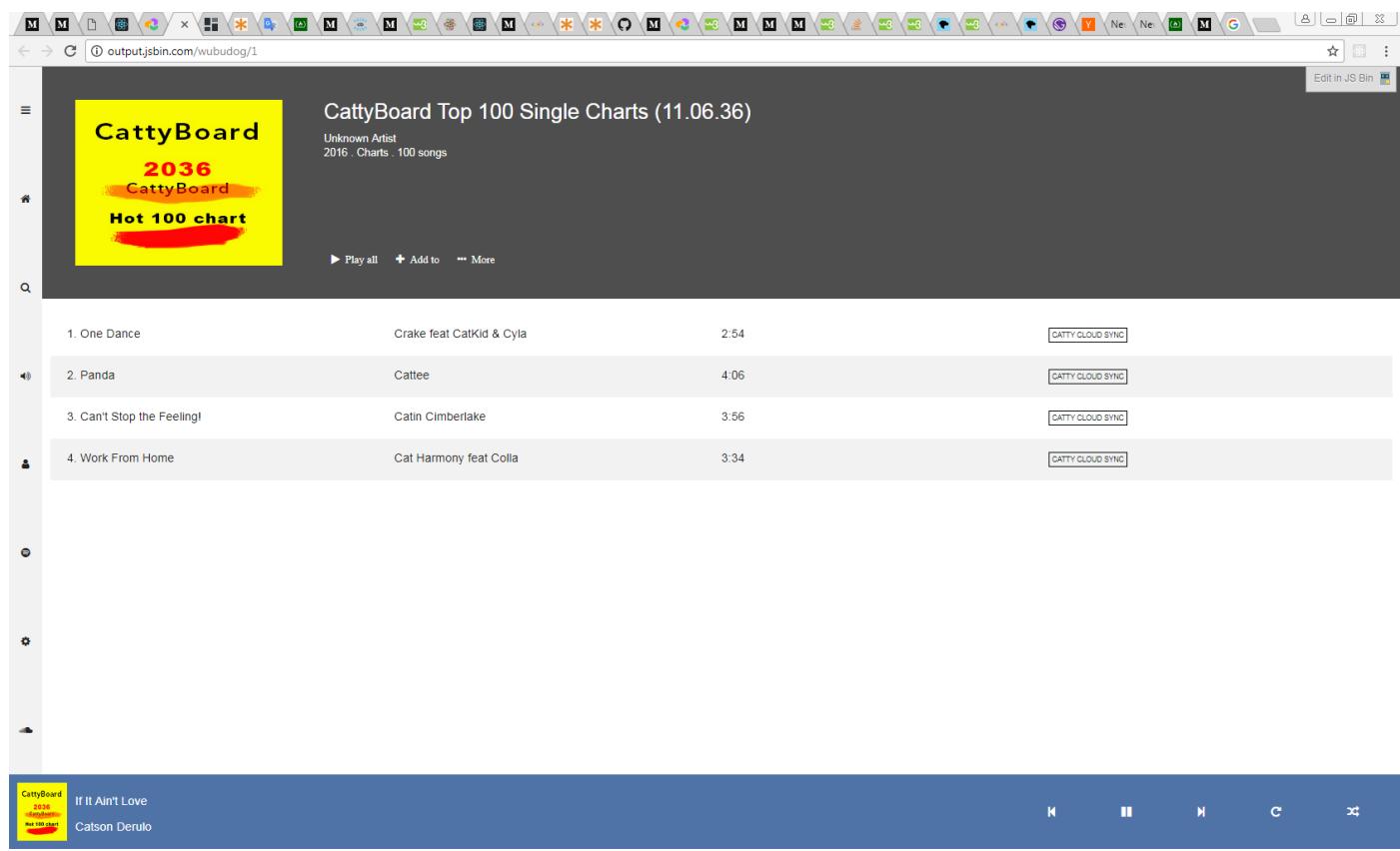
<https://github.com/philipwalton/solved-by-flexbox>

<https://github.com/philipwalton/flexbugs>

<https://philipwalton.com/>

GRADJENJE MUSIC APPLICATION LAYOUT-A SA FLEXBOX-OM

OVAKO CE IZGLEDATI, GOTOV LAYOUT, KADA GA BUDEM SAGRADIO



AKO OVO GLEDAM NA MOBILNOM UREĐAJU, IMAĆEU MALO DRUGAČIJI IZGLED; TO JE NEŠTO NA ČEMU ĆU RADITI U "RESPONSIVE" DIJELU, OVIH TUTORIJALA

MEDJUTIM, LOSA PRAKSA JE, POTPUNA IZGRADNJA LAYOUT-A, POMOCU FLEXBOX-A

IZ MNOGIH RAZLOGA OVO MOŽDA NIJE IDEALNO; ALI TO JE NAMJERNO U OVOJ LEKCIJI

MEDJUTIM, ONO STO CU VIDETI JESU SVE STVARI KOJE MOGU DA URADIM SA FLEXBOX-OM, OBRAĐENO U OKVIRU JEDNOG PROJEKTA

SLEDECİ CLANAK BI TREBAO DA PROCITAM O TOME, ZASTO JE POMENUTO LOSA PRAKASA

<https://medium.com/@ohansemmanuel/flexbox-is-awesome-but-its-not-welcome-here-a90601c292b6>

SVE U Catty-Music-U JE IZGRADJENO, POMOĆU FLEXBOX MODELA - I TO JE NAMJERNO, JER AUTOR TUTORIJALA ZELI DA POKAZE STA JE MOGUĆE

PRE IZRADE PROJEKTA, POTREBNO JE PLANIRANJE

PRVO CU PROCI KROZ PROCES PLANIRANJA, ODNOSEN KROZ APPROACH PLANIRANJA, GRADNJE Carry-Music LAYOUT-A

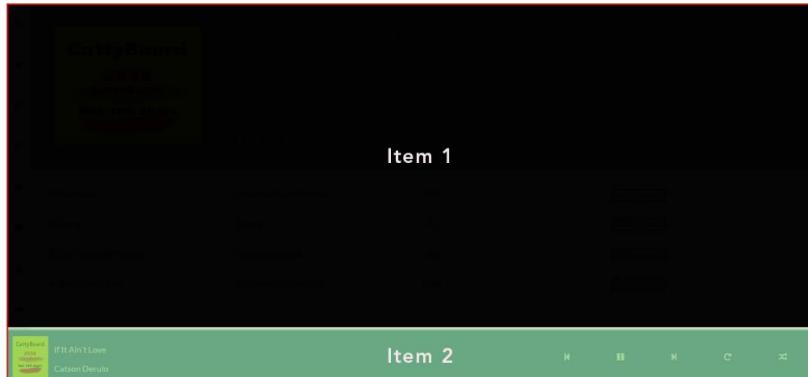
PA, GDE POČINJEŠ?

KADA GRADIM LAYOUT SA FLEKSBOXOM, TREBALO BI DA ZAPOČNEM, TAKO ŠTO ĆU POTRAŽITI KOJI SE DELOVI MOG LAYOUT-A MOGU IZDVOJITI KAO FLEX CONTAINER-I; POTOM MOŽETE POTAKNUTI (LEVERAGE) MOĆNE PROPERTIJE ALIGNMENT-A, KOJE OBEZBEDJUJE FLEXBOX

BREAKDOWN

PONOVO CU POGLEDATI ZAVRSENI LAYOUT

MOGU IMATI CELOKUPNO CONTAINING BODY KAO FLEX CONTAINER (PREDSTAVLJEN CRVENOM GRANICOM NA SLICI ISPOD) KOJE CE OBUVATATI DVA FLEX ITEM-A



OVO IMA POTPUN SMISAO, JER FLEX ITEM 1 SADRŽI SVAKI DEO LAYOUT-A, KOJI NIJE FOOTER "PODNOŽJE" - DEO KOJI SADRŽI TASTERE ZA KONTROLU MUZIKE

ONO STO JE MOGUĆE jeste da FLEX ITEM, TAKODJE BUDE FLEX CONTAINER

DA, TO JE MOGUĆE!

MOZE SE NEST-OVATI, KOLIKO ZELIM (IAKO JE TO NORMALNO DA TO CINIM NA RAZUMNOM NIVOU)

DAKLE, S TIM NOVIM OTKRIĆEM DOLAZI OVO ...

Item 1 (PRVI FLEX ITEM) TAKOĐE MOŽE BITI FLEX CONTAINER

BOĆNA TRAKA, ODNOSENIO SIDEBAR (Item 1b) I GLAVNI DIO (Item 1a) TADA BI BILI FLEX ITEM-I



JOŠ SI SA MNOM, ZAR NE?

RAZLAGANJE (ODNOSNO DECOMPOSING) LAYOUTA, NA POMENUTI NACIN MI DAJE DOBAR MENTALNI MODEL ZA RAD

KADA POČNEM DA GRADIM, JOŠ SLOŽENIJE IZGLEDE SA MODEЛОМ FLEKBOX-A, VIDEĆU KAKO JE TO VAŽNO

NE TREBA VAM FANTASTIČNA SLIKA POPUT ONIH IZNAD. JEDNOSTAVNA GRUBA SKICA NA PAPIRU TREBA DA BUDE U REDU TO KEEP YOU GOING

SEĆAŠ LI SE DA SAM REKAO DA JE MOGUĆ NESTING, DEEP AS I WANT? IZGLEDA DA I OVDE JOS MOGU NEST-OVATI

POGLEDAJ GLAVNI DEO GORE (Item 1a).

OD NJEGA SE MOZE NAPRAVITI I FLEX CONTAINER, KAKO BI SE POSTAVILI DIJELOVI KOJI SU OZNAČENI ISPOD (Item 1a-A I Item1a-B)



MOŽES ODLUČITI DA GLAVNU SEKCIJU (Item 1a) NE NAPRAVITE FLEX CONTAINER-OM, I U NJU STAVIM SAMO DVA DI -A ZA HOUSING, DVE OZNACENE (HIGHLIGHTED) SEKCIJE

DA, TO JE MOGUĆE, POŠTO SU "Item 1a-A" I "Item 1a-B" STACKED VERTIKALNO

PODRAZUMEVANO, "DIV"-OVI SE STACK-UJU VERTIKALNO; JER TAKO FUNKCIONIŠE box MODEL

AKO ODLUČIM DA GLAVNI DEO NAPRAVIM FLEX CONTAINER-OM, DOBIJAM MOĆNE KARAKTERISTIKE ALIGNMENT-A NA RASPOLAGANJU; SAMO AKO SU MI SLUCAJNO POTREBNE U BILO KOM TRENUTKU.

"flex" U FLEXBOX-U, ZNAČI FLEKSIBILAN

FLEX CONTAINER-I SU, PODRAZUMEVANO FLEKSIBILNI, STO JE VRST RESPONIVENESS-A

IVO JE MOŽDA JOŠ JEDAN RAZLOG ZA KORIŠTENJE FLEX KONTEJNERA UMESTO REDOVNIH "DIVS"-A; TO ZAVISI I OD SCENARIJA SLUČAJA

DODIRNUĆU NEKE DRUGE STVARI DOK, BUDEM PRAVIO Catty Music; JER SADA BI TREBALO DA PIŠEM, NEKI CODE

POCECU SA OSNOVNIM HTML SETUP-OM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Catty Music</title>
  </head>
  <body>

    <main></main>      <!--TREBA DA SADRZI GLAVNU SEKCIJU APLIKACIJE-->

    <footer></footer>  <!--TREBA DA SADRZI MUSIC CONTROL BUTTONS I SONG DETAILS-->

  </body>
</html>
```

```
html, body {
  height: 100%;      /*EKSPlicitno podeševanje ovoga je važno*/
}

body {
  display: flex;      /*flex superpowers activated*/
  flex-direction: column; /*STACK-UJ VERTIKALNO FLEX ITEM-E (main I footer ELEMENT), A NE HORIZONTALNO */
}
```

PRVI KORAK U KORIŠĆENJU FLEXBOX MODELA JESTE USPOSTAVLJANJE FLEX CONTAINER-A

UPRAVO TO I RADI CODE IZNAD. POSTAVLJA body-JEV display PROPERTI NA flex

SADA IMAM flex CONTAINER, A TO JE body ELEMENT

FLEX ITEM-I SU TAKOĐE DEFINISAN (Item 1 | Item 2) - KAO ŠTO JE U BREAKDOWN-U NAVEDENO I PRIKAZANO

IMAJ NA UMU DA BISTE TREBALI POGLEDATI SLIKE KOJE SU POKAZANE U INICIJALNOM BREAKDOWN-U, RANIJE; AKO OVAJ KONCEPT I DALJE IZGLEDA NEJASANIM

IMAJUCI SLIKE FINALNOG PROIZVODA U VIDOKRUGU, OMOGUCICE LAKSI BUILDING

UCINI DA footer **BUDE ZALEPLJEN ZA DNO (MAKE THE footer STICK TO THE BOTTOM)**

footer , KOJI HOUSE-UJE MUZIČKE KONTROLE TREBA DA BUDE ZALEPLJEN ZA DNO, STRANICE, DOK mian SEKCIJA, TREBA DA ISPUNJAVA OSTATAK STRANICE

KAKO TO DA URADIM?

```
main {
  flex: 1 0 auto;      /*POPUNJAVANJE EXTRA SPACE-A*/
}

footer {
  flex: 0 0 90px;     /*NEMOJ NI DA RASTES, NI DA SE SMANJUJES, OSTANI SAMO NA VISINI 90px*/
}
```

ZAHVALJUJUĆI FLEX-GROW PROPERETIJU; RELATIVNO JE LAKO UCINITI DA main SEKCIJA POPUNI CEO PROSTOR; DAKLE SAMO PODESIM VREDNOST flex-grow NA 1; A TAKOĐE TREBA POSTAVITI VREDNOST flex-shrink PROPERTIJA NA NULA (0)

RAZLOG MOŽDA OVDE NIJE OČIGLEDAN JER JE PROMENJENA VREDNOST flex-direction

U NEKIM PREGLEDAČIMA POSTOJI GREŠKA KOJA DOZVOLJAVA FLEX ITEM-IMA DA SE SMANJUJU ISPOD NJIHOVE VELIČINE SADRŽAJA; STO JE ČUDNO PONAŠANJE

REŠENJE ZA OVU GREŠKU JE DA SE VRIJEDNOST flex-shrink ODRŽAVA NA NULI, A NE NA DEFULT 1 VREDNOSTI, I TAKODJE TREBA POSTAVITI VREDNOST flex-basis NA auto

TO JE KAO DA KAZEM FLEX ITEM-U: "MOLIM TE, PREUZMI POČETNU ŠIRINU ZASNOVANU NA TVOJOJ VELIČINI SADRŽAJA, ALI NIKADA SE NE SMANJUJ"

SA OVOM STENOGRAFSKOM VREDNOŠĆU, I DALJE DOBIJAM PODRAZUMEVANO PONAŠANJE FLEX ITEM-A

FLEX ITEM BI SE SMANJIO, PRILIKOM PROMENE VELIČINE PRETRAŽIVAČA; TAJ RESIZING NIJE BAZIRAN NA flex-shrink; ZASNOVAN JE NA RECOMPUTING-U ŠIRINE FLEX ITEM-A, AUTOMATSKI:

flex-basis: auto

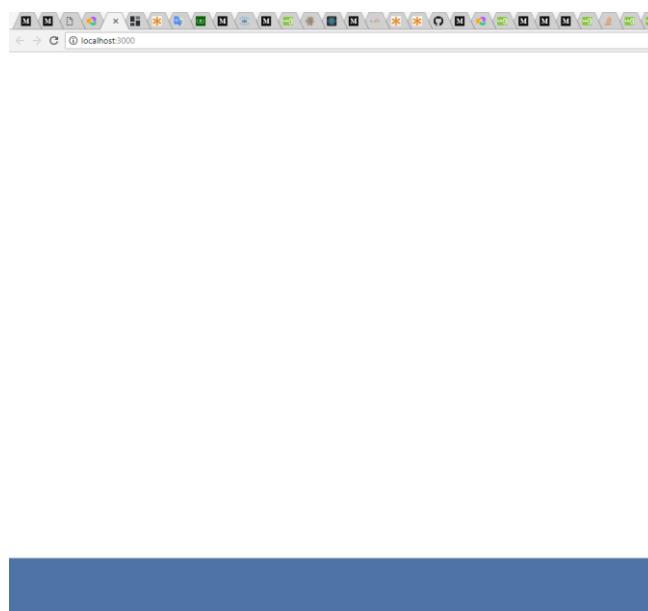
IVO ĆE UZROKOVATI DA JE FLEX ITEM, BAREM TOLIKO VELIK KAO NJEOVA ŠIRINA ILI VISINA (AKO JE DEKLARISANA) ILI NJEOVA PODRAZUMEVANA VELIČINA SADRŽAJA

MOLIM TE DA NE ZABORAVIS OKVIR (FRAMEWORK) ZA KOJI, IZ KOJEG POTICU SVE VREDNOSTI flex SHORTHAND-A, JER CE U NASTAVKU DOCI MNOGO SHORTHAND STVARI

SADA CU DODATI STILIZOVANJE, KAKO BI DEFINISAO SPACING, COLOR I JOS TOGA

```
body {  
background-color: #fff;  
margin: 0;  
font-family: Lato, sans-serif;  
color: #222;  
font-size: 0.9em;  
  
display: flex;  
flex-direction: column;  
}  
  
footer {  
padding: 10px;  
color: #fff;  
background-color: rgba(61, 100, 158, .9);  
  
flex: 0 0 90px;  
}
```

NISTA, JOS MAGICNO, OVAKO ZA SADA IZGLEDA STRANICA



GLEDAJUCI, KAKO STVARI POCINUJU DA SE OBLIKUJU, UCINICU JE JOS BOLJOM

SADA CU UPDATE-OVATI HTML DOKUMENT (KORISTICU IKONE, SA Font Awesome STRANICE)

<https://fontawesome.com/how-to-use/svg-with-js>

OVO IDE U HEAD SEKCIJU HTML DOKUMENTA

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<!--<Link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.10/css/all.css" integrity="sha384-d0P83n9kaQMcwj8F4RJB66tzIwOKmrdb46+porD/OvrJ+37WqIM7UoBtwHO6Nlg" crossorigin="anonymous">-->
```

MOGAO SAM I UMESTO UCITAVANJE SA STRANICE (UZ POMOC link TAGA), ZAMENITI svg TAGOM U, KOJI JE VEC DEFINISAN I KOJEG BI KOPIRAO SA STRANICE, ALI NECU TO SADA URADITI

```
<main>
  <aside>          <!-- OVO PREDSTAVLJA SIDE-BAR I OBUHVATA SVE IKONICE SA Font Awesome WEB STRANICE--&gt;
    &lt;i class="fa fa-bars"&gt;&lt;/i&gt;
    &lt;i class="fa fa-home"&gt;&lt;/i&gt;
    &lt;i class="fa fa-search"&gt;&lt;/i&gt;
    &lt;i class="fa fa-volume-up"&gt;&lt;/i&gt;
    &lt;i class="fa fa-user"&gt;&lt;/i&gt;
    &lt;i class="fa fa-spotify"&gt;&lt;/i&gt;
    &lt;i class="fa fa-cog"&gt;&lt;/i&gt;
    &lt;i class="fa fa-soundcloud"&gt;&lt;/i&gt;
  &lt;/aside&gt;
  &lt;section class="content"&gt;&lt;/section&gt;  <!-- OVA SEKCIJA CE HOUSE-OVATI, SVE OSTALO; OSIME SIDE-BAR-a--&gt;
&lt;/main&gt;
&lt;footer&gt;&lt;/footer&gt;</pre>
```

KAO ŠTO JE OBJAŠNJENO RANIJE, main ĆE, TAKOĐE BITI, UCINJEN FLEX CONTAINER-OM; TAKO DA CE BOČNA TRAKA, ODNOSENJE SIDEBAR (REPREZENOVANA aside TAGOM NA SLICI GORE), I SEKCIJA (section) BITI FLEX ITEM-I

```
main {
  flex: 1 0 auto;      /*DAKLE main JESTE I FLEX ITEM*/
  display: flex;        /*ALI JE I FLEX CONTAINER, KOJI OBUHVATA aside I section, KAO FLEX ITEM-E*/
}
```

U REDU, OVO POSTAJE ZANIMLJIVO, A?

SADA IMAM main, KAO FLEX CONTAINER

SADA SE MORAM POZABAVITI, JEDNIM OD NJEGOVIH FLEX ITEM-A, A TO JE SIDE BAR

OVDE SE NESTO COOL DOGADJA

SIDEBAR TREBA DA IMA IKONE SLOŽENE VERTIKALNO; TAKO DA CU SIDEBAR (aside) UCINITI FLEX CONTAINER-OM, I ZADACU MU ODREDJENI flex-direction, KOJI OMOGUCAVA DA SE IKONICE STACK-UJU VERTIKALNO

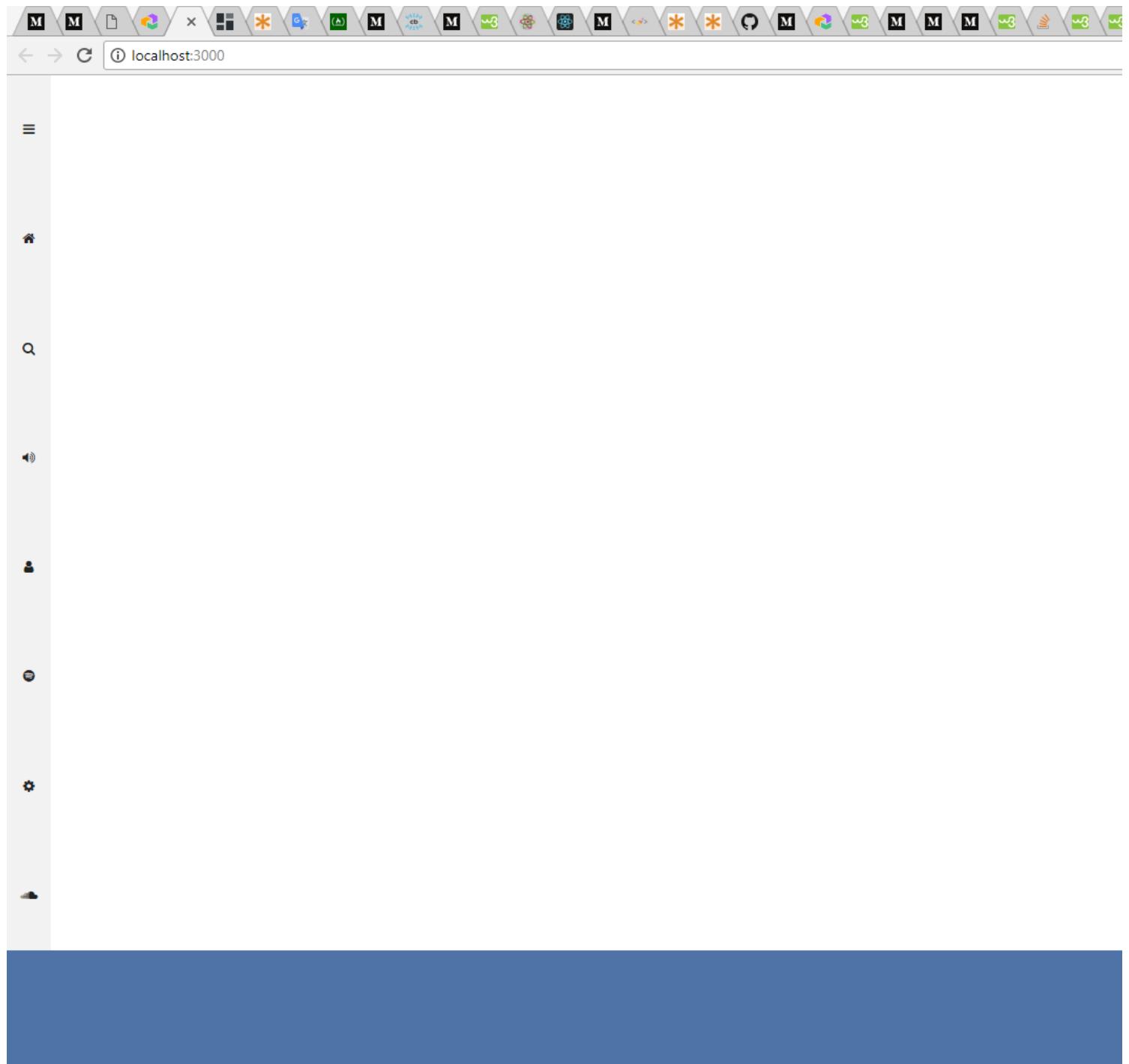
```
aside {
  flex: 0 0 42px;      /*OVAJ FLEX ITEM NE TREBA NI DA GROW, A NI DA SHRINK; A SA FIKSIRANOM VISINOM JE OD 42 PIKSELA*/
  display: flex;        /*ALI TAKODJE JE I FLEX CONTAINER, KOJI MOZE ODLUCITI, KAKO SU NJEGOVI FLEX ITEM-I, POLOZENI*/
  flex-direction: column; /*NJEGOVI FLEX ITEM-I, SU STACKED VERTIKALNO, DAKLE PROMENJEN JE DEFAULT*/
  justify-content: space-around; /*OVO JE INTERESANTNO, POSTO JE DIRECTION PROMENJEN, OVO SAD FUNKCIONISE ZA VERTIKALNI DIRECTION*/
  align-items: center;   /*POSTO JE DIRECTION PROMENJEN, OVO UTICE NA HORIZONTALNI DIRECTION*/
  background-color: #f2f2f2; /*DA UCINIM OVAJ FLEX CONTAINER LEPIM*/
}

aside i.fa {
  font-size: 0.9em;     /*VELICINA FONTA ZA IKONICE*/
}
```

GORNJI CODE JE, OPSESIVNO, PROKOMENTARISAN

DAKLE, SVE JE UREDNO, SAMO NEKOLIKO LINIJA CODE-A; MOGU DA VIDIM REASONABLE CODE BEZ MESSY HACK-OVA

PRIKAZACU, KAK OCE ZA SADA, SVE IZGLEDATI, NA WEB STRANICI



DODAVANJE SADRZINE, main -U (ADDING CONTENT TO THE main SECTION)

CONTENT SEKCIJA JE PRAZNA, ALI NE ZABORAVI DA JE ONA DRUGI LIST ITEM; A SIDEBAR JE PRVI STAVI NEKE STVARI TAMO

ŠTA MISLIŠ?

PONOVO MOŽES POGLEDATI, ZAVRŠENI PROJEKAT, TAKO DA NE IZGUBIS SIGHT, NA TO GDE SI KRENUO; A ŠTO JE JOŠ VAŽNIJE, POMOGLO BI TI DA RAZUMES SLEDEĆI CODE LISTING

```

<section class="content">          <!--OVA SEKCIJA JE BILA PRAZNA, POPULATING IT WITH CONTENT-->

    <div class="music-head">      <!--PRVI LIST ITEM, SADRZI MUZICKE DETALJE-->

              <!--ALBUM ART-->

        <section class="catty-music">    <!--DRUGI DETALJI ALBUMA-->

            <div>
                <p>CattyBoarf Top 100 Single Charts (11.06.36)</p>
                <p>Unknown Artist</p>
                <p>2016 . Charts . 100 songs</p>
            </div>

            <div>  <!--MUSIC CONTROLS-->
                <i class="fa fa-play">&ampnbspPlay all</i>
                <i class="fa fa-plus">&ampnbspAdd to</i>
                <i class="fa fa-ellipsis-h">&ampnbsp&ampnbspMore</i>
            </div>

        </section>

    </div>      <!--KRAJ music-head-A-->

    <ul class="music-list">          <!--DRUGI LIST ITEM; SADRZI LISTU SVIH DISPLAYED PESAMA-->

        <li>
            <p>1. One Dance</p>
            <p>Crake feat CatKid &amp; Cyla</p>
            <p>2:54</p>
            <p><span class="catty-cloud">CATTY CLOUD SYNC</span></p>
        </li>
        <li>
            <p>2. Panda</p>
            <p>Cattee</p>
            <p>4:06</p>
            <p><span class="catty-cloud">CATTY CLOUD SYNC</span></p>
        </li>
        <li>
            <p>3. Can't Stop The Feeling!</p>
            <p>Catin Cimberlake</p>
            <p>3:56</p>
            <p><span class="catty-cloud">CATTY CLOUD SYNC</span></p>
        </li>
        <li>
            <p>4. Work From Home</p>
            <p>Cat Harmony feat Colla</p>
            <p>3:34</p>
            <p><span class="catty-cloud">CATTY CLOUD SYNC</span></p>
        </li>
    </ul>

</section>

```

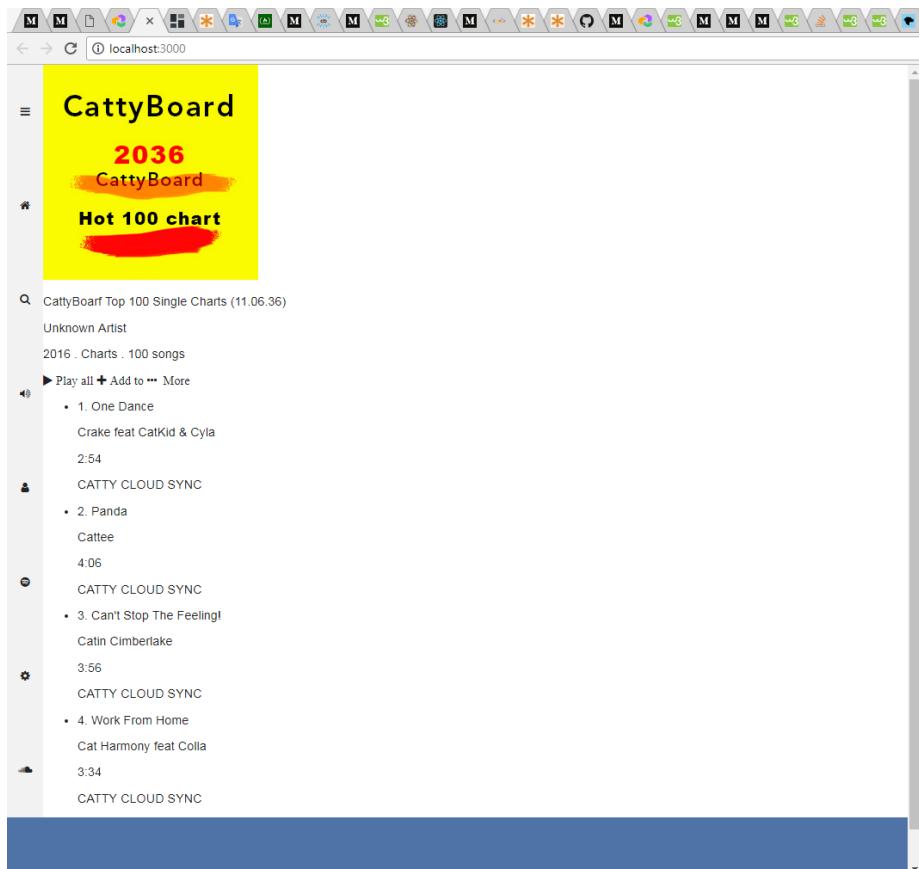
DODAO SAM MALO VIŠE NEGO POSLEDNJI PUT, ALI PRILIČNO JE JEDNOSTAVNO

NASELIO (POPULATED) SAM, PRAZNI DEO SADRŽINE SA DIV-OM KOJI HOLDS ALBUM ART (image) I NEKE DETALJE O ALBUMU

ŠTA JE SA UNORDERED LISTOM?

ul HOLDS LISTU PESAMA IZ ALBUMA; NASLOV PESME, UMJETNIK, VRIJEME I "CATTY CLOUD SYNC" SADRŽANI SU U POJEDINAČNIM PARAGRAFIMA U LISTI

POGLEDAJ SVE I PROBAJ DA UVIDIŠ STA CES SA STILOM



PRVO, TREBALO BI DA GLAVNI CONTENT section, NAPRAVIM FLEX CONTAINER-OM

```
.content {  
    display: flex;  
    flex-direction: column;  
    flex: 1 0 auto; /*OVIM SE POSTIZE DA SECTION RASTE (PO SIRINI) KAKO BI ISPUNIO CEO DOSTUPNI PROSTOR*/  
}
```

SADA CU DEFINISATI STILOVE, I ZA NJEGOVE FLEX ITEM-E

```
.music-head {  
    display: flex;  
    flex: 0 0 280px; /*NEMOJ DA RASTES (PO VISINI OVOG PUTA)*, A NI DA SE SMANJUJES, A VISINA TREBA DA BUDE STALNO 280px */  
    padding: 42px;  
    background-color: #4e4e4e;  
}  
  
.music-list {  
    flex: 1 0 auto;  
    list-style-type: none;  
    padding: 5px 10px 0px;  
}
```

.music-head HOLDS ALBUM ART (img) I DRUGE SRODNE DETALJE O ALBUMU

NE RASTE ILI SE SMANJUJE, ALI ZADRŽAVA VISINU OD 280px

VISINA? NIJE ŠIRINA? DA!

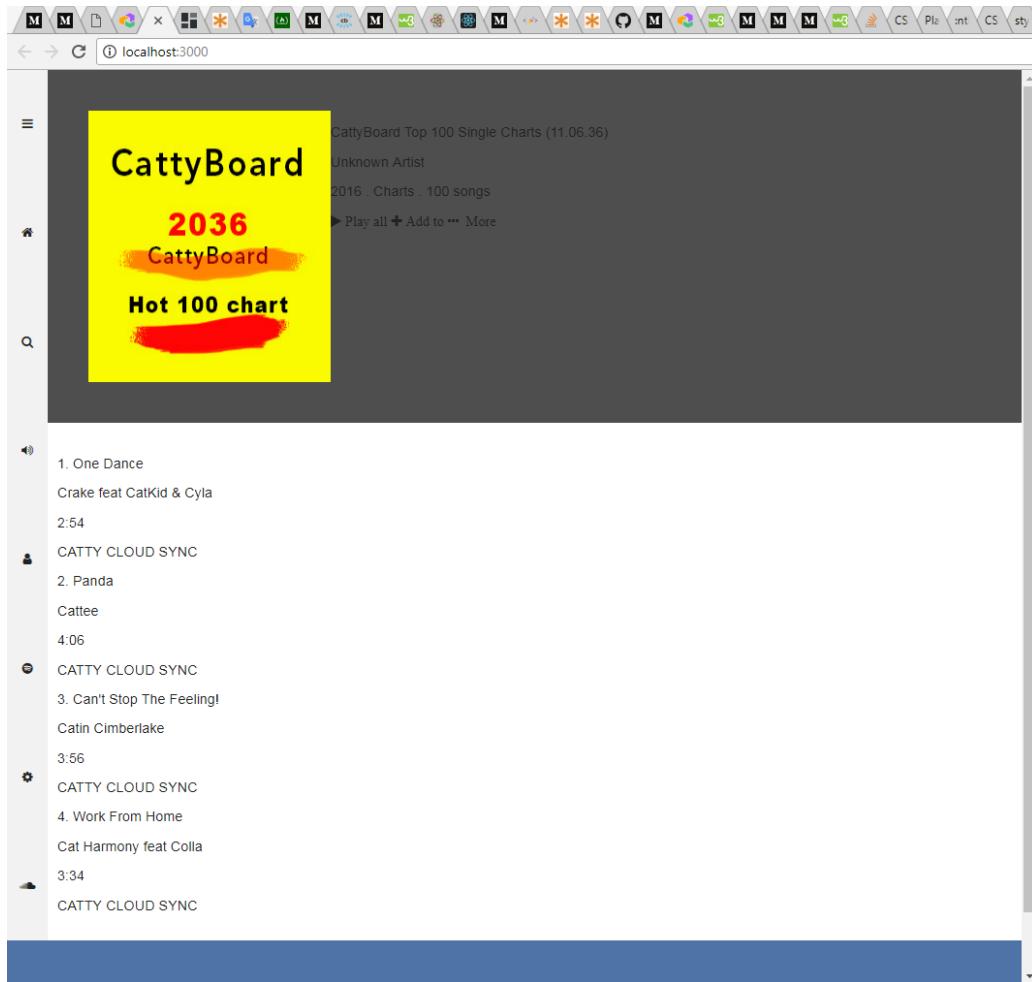
RODITELJSKI ELEMENT VEĆ JE PROMENIO flex-direction

.music-head TAKODJE MORA DA BUDE I FLEX CONTAINER, ZA KASNIJE, ZATO SAM MU DODAO:

display: flex

.music-list SADRŽI LISTU PESAMA I POPUNJAVA PREOSTALI RASPOLOŽIVI PROSTOR KOJI SE DELI SA .music-head OM, IZNAD NJEGA

MEDJUTIM ONO STO MI SE NE SVIDJA JESTE, DA JE ZBOG DEFINISANJA PADDING-A, FOOTER-VISE NIJE ZALEPLJEN ZA DNO BROWSER WIDNOW-A (STO SE MOZE VIDETI NA SLEDECOJ SLICI, TAKO STO MOGU DA PRIMETIM DA POSTOJI "SCROLL-ER") (****OVO CU DODATNO PROVERITI, KADA BUDEM OPET GRADIO SLICAN LAYOUT****)

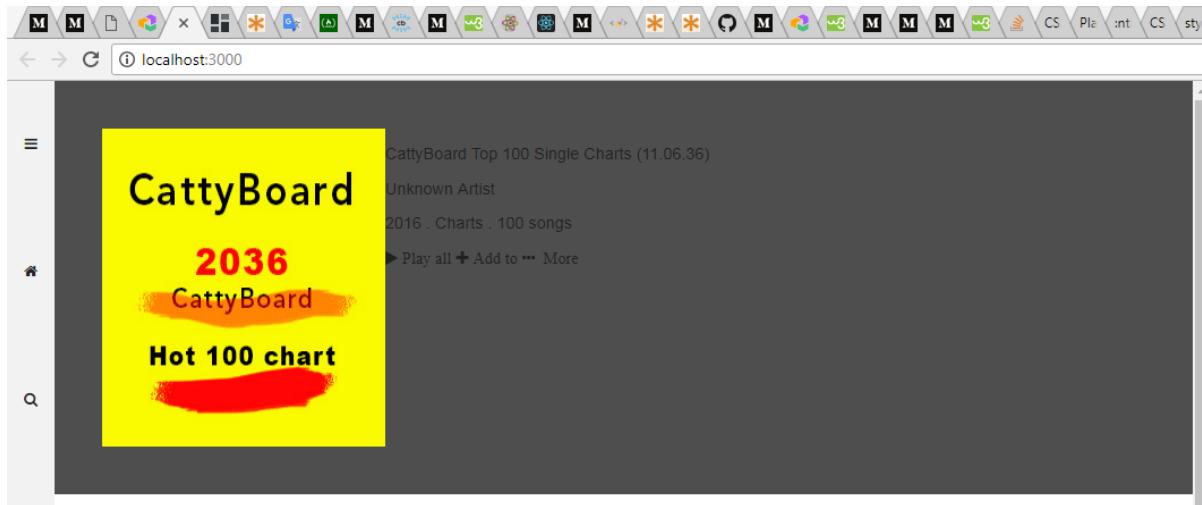


MEDJUTIM, PRE NEGO STO SE POZABAVIM POMENUTIM, POSTOJI JOS NEKOLIKO, PROBLEMATIČNIS STVARI

1) LISTA PESAMA IZGLEDA UZASNO



2) section KOJI SADRŽI MUSIC ART, IMA TEKST, KOJI VEOMA RUZNO IZGLEDA



BAVLJENJE SA LISTOM PESAMA (DEALING WITH THE LIST OF SONGS)

SVAKA LISTA PESAMA SADRŽI 4 PARAGRAFA:

NASLOV PESME, UMETNIK, TRAJANJE I "SINHRONIZACIJA U OBLAKU"

MORA POSTOJATI NAČIN DA SVE OVO STAVIM U JEDNU LINIJU; A DA SVAKI PARAGRAF, ZAUZIMA ISTI PROSTOR DUŽ POMENUTE LINIJE

OPET CU KORISTITI FLEXBOX

KONCEPT, OVDE PRISUTAN, JE EMPLOYED, U MNOGIM GRID SISTEMIM-A

PREVESCU TO U CODE

```
li {  
    display: flex; /*TARGET-UJE SVAKI LIST ITEM, A TAJ LIST ITEM OBUVATA I PARAGRAFE*/  
    padding: 0 20px;  
    min-height: 50px;  
}  
  
li p {  
    flex: 0 0 25%;  
}
```

VIDIS LI STA SE DESAVA SA PARAGRAFIMA

SLEDECE: flex: 0 0 25% UPRAVO, ZNACI:

"NE RASTI, A NI NE SMANJUJ SE, ALI SVAKI PARAGRAF BI TREBAO ZAUZIMA 25% RASPOLOŽIVOG PROSTORA" (MOGU ZAPASTI U ZABLUDU, PA RECI SEBI DA JE REC O PREOSTALOM PROSTORU, KOJI NIJE ZAUZET OD PARAGRAF ELEMENTATA; MEDJUTIM, OVDE SE MISLI NA UKUPAN PROSTOR (UKUPNA SIRINA), I SVAKI PARAGRAF CE ZAUZETI 25% OD OG PROSTORA)

PROSTOR SE PODJEDNAKO DELI MEĐU PARAGRAFIMA

ZA SADA LAYOUT, IZGLEDA OVAKO

CattyBoard Top 100 Single Charts (11.06.36)

Unknown Artist

2016 . Charts . 100 songs

▶ Play all + Add to *** More

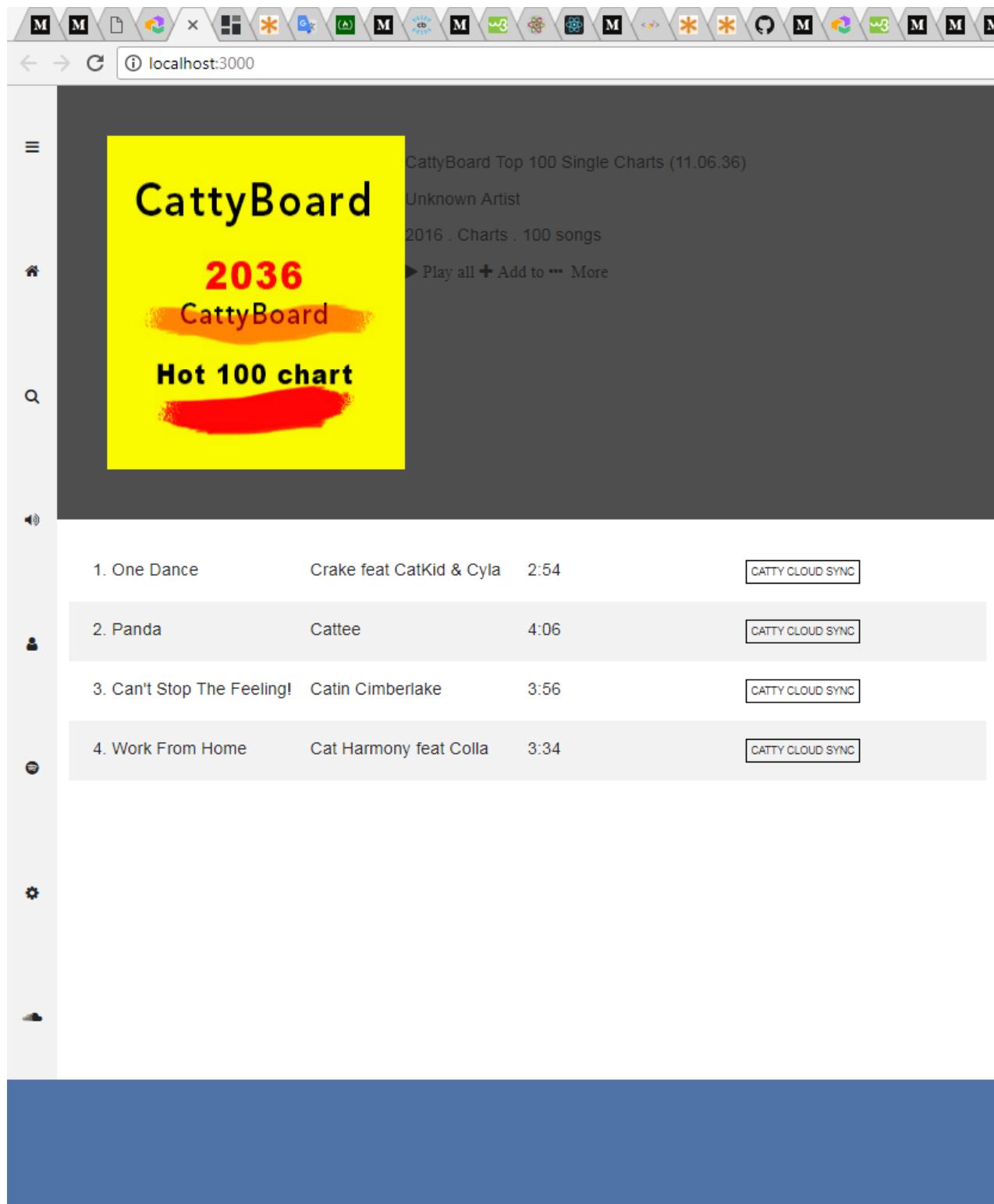
CattyBoard
Hot 100 chart

	Song	Artist	Length	Source
1.	One Dance	Drake feat CatKid & Cyla	2:54	CATTY CLOUD SYNC
2.	Panda	Cattee	4:06	CATTY CLOUD SYNC
3.	Can't Stop The Feeling!	Catin Cimberlake	3:56	CATTY CLOUD SYNC
4.	Work From Home	Cat Harmony feat Colla	3:34	CATTY CLOUD SYNC

STILIZOVACU LISTU JOS MALO, TAKO STO CU SVAKOM DRUGOM LIST ITEMU DATI RAZLICIT BACKGROUND COLOR; A POZABAVICU SE I SA "CATTY CLOUD SYNC"-OM

```
li:nth-child(2n) {  
    background-color: #f2f2f2;  
}  
  
li span.catty-cloud {  
    border: black solid 1px;  
    font-size: 0.6em;  
    padding: 4px;  
}
```

POKAZACU JOJ JEDNOM, KAKO IZGLEDA LAYOUT



KORISCENJE OVE TEHNIKE (MILSI SE NA flex: 0 0 25%)

OVA TEHNIKA JE NEPROCENJIVA; MOŽES JE KORISTITI DA BI KREIRAO NEJEDNAKE OBLASTI SADRŽAJA; NA PRIMER, PRIKAZ 2 KOLONE

JEDAN DEO MOŽE DA ZAUZIMA 60%, RASPOLOŽIVOG PROSTORA, A DRUGI MOZE 40%

.first-section {flex: 0 0 60%;}

.second-section {flex: 0 0 40%;}

OVU TEHNIKU MOŽETE KORISTITI ZA IZRADU GRID SISTEMA

A SADA CU SE POZABAVITI I SA DRUGIM PROBLEMMOM

CINJENJE DA ALBUM DETAILS TEKST, POSTANE LEPSI

```
.catty-music {  
    padding-left: 50px;  
    font-weight: 300;  
    color: #fff;  
  
    display: flex;  
    flex-direction: column;  
  
    flex: 1 0 auto;  
}  
  
.catty-music div:nth-child(1) {  
    margin-bottom: auto;  
}  
  
.catty-music div:nth-child(2) {  
    margin-top: 0;  
}  
  
.catty-music div:nth-child(2) i.fa {  
    font-size: 0.9em;  
    padding: 0 0.8em;  
    font-weight: 300;  
}  
  
.catty-music div:nth-child(1) p:first-child {  
    font-size: 1.8em;  
    margin: 0 0 10px;  
}  
  
.catty-music div:nth-child(1) p:not(:first-child) {  
    font-size: 0.9em;  
    margin: 2px 0;  
}
```

The screenshot shows a web-based music player interface. At the top, there's a navigation bar with various icons. Below it, the title "CattyBoard Top 100 Single Charts (11.06.36)" is displayed, along with the artist "Unknown Artist" and the date "2016 . Charts . 100 songs". To the left of the main content area, there's a sidebar with icons for search, refresh, and other functions. The main content area features a yellow album cover with the text "CattyBoard 2036 CattyBoard Hot 100 chart". Below the cover, there are buttons for "Play all", "Add to", and "More". The main list displays four songs:

Rank	Song Title	Artist	Length	Action
1.	One Dance	Crake feat CatKid & Cyla	2:54	CATTY CLOUD SYNC
2.	Panda	Cattee	4:06	CATTY CLOUD SYNC
3.	Can't Stop The Feeling!	Catin Cimberlake	3:56	CATTY CLOUD SYNC
4.	Work From Home	Cat Harmony feat Colla	3:34	CATTY CLOUD SYNC

SADA CU SE POZABAVITI I FOOTER-OM

```
<footer>
  
  <div>
    <p>If it Ain't Love</p>
    <p>Catson Derulo</p>
  </div>
  <section>
    <i class="fa fa-play"></i>
    <i class="fa fa-pause"></i>
    <i class="fa fa-step-forward"></i>
    <i class="fa fa-reply"></i>
    <i class="fa fa-volume-up"></i>
  </section>
</footer>
```

```
<link rel="stylesheet" type="text/css" href="%PUBLIC_URL%/fontawesome.css">
```

```
footer {
  display: flex;
}

footer div {
  margin-left: 8px;
  margin-right: auto;
}

footer img {
  height: 5rem;
}

footer section {
  display: flex;
  flex: 0 0 25%;
  justify-content: space-between;
  align-items: center;
}
```

CattyBoard Top 100 Single Charts (11.06.36)

Unknown Artist
2016 Charts 100 songs

CattyBoard 2036 Hot 100 chart

Play all Add to More

1. One Dance	Crake feat CatKid & Cyla	2:54	CATTY CLOUD SYNC
2. Panda	Cattee	4:06	CATTY CLOUD SYNC
3. Can't Stop The Feeling!	Catin Cimberlake	3:56	CATTY CLOUD SYNC
4. Work From Home	Cat Harmony feat Colla	3:34	CATTY CLOUD SYNC

If it Ain't Love
Catson Derulo

RESPONSIVE DESIGN SA FLEXBOX-OM

KNJIGE SU NAPISANE O RESPONSIVE DIZAJNU, DOBRE KNJIGE

POŠTO SE OVAJ TUTORIJAL FOKUSIRA NA FLEXBOX MODEL; U OVOM TRENTUKU, NECU SE DETALJNIJE BAVITI RESPONSIVE DESIGN-OM

KORISCENJEM FLEXBOX MODELA, POSTISE SE ODREDJENI RESPONSIVENESS

FLEKBOX ZNACI "FLEKSIBILNA KUTIJA"; MEĐUTIM, MOGUĆE JE CILJATI RAZLIČITE VELIČINE EKRANA PUTEM MEDIJSKIH UPITA (MEDIA QUERIES), A ZATIM PROMENITI FLEXPONASANJE

EVO PRIMERA; KREIRACU JEDNU UNORDERED LIST-U

```
ul {  
    list-style-type: none;  
    border: #4e4e4e solid 1px;  
  
    display: flex;  
}  
  
ul li {  
  
    background-color: #8cacea;  
    padding: 10px;  
    margin: 10px;  
    color: #fff;  
    font-size: 1em;  
  
    flex: 0 0 auto;  
}
```

```
<ul>  
  <li>Home</li>  
  <li>About</li>  
  <li>Contacts</li>  
  <li>Register</li>  
  <li>Login</li>  
</ul>
```

NAVIGATION BAR IZGLEDA OVAKO



TREBALO BI SADA DA RAZUMEM CODE, KOJI SAM NAPISAO GORE

IAKO JE OVO COOL ZA DESKTOPE I TABLETE, NA ODREĐENIM VELIČINAMA, OVO NECE IZGLEDATI DOBRO

NA MOBILNOM TELEFONU, ŽELECU DA, VERTIKALNO STACK-UJEM NAV ITEMU

ZATO SE MORAM POZABAVITI MEDIA QUERIES-IMA

```
@media screen and (max-width: 769px) {  
  
    /*code koji je ovde applies samo na screen device-ima koji imaju sirinu manju od 769px*/  
  
    ul {  
        flex-direction: column;  
    }  
}
```

DAKLE, NA MANJIM UREDJAJIMA CE SE PROMENITI flex-direction, FLEX XONTAINER-A

KADA SMANJIM SIRINU MOG BROWSERA, ISPOD POMENUTE VELICINE, OVAKO CE IZGLEDATI FLEX ITEM-I

POSTO SE JOS NISAM UPOZNAO SA MEDIA QUERIES-IMA, SADA CU SE NJIMA, MALO POZBAVITI (ALI SAMO NAKRATKO), A KASNIJE ILI U NEKO MDRUGOM TUTORIJALU, POTREBNO JE DA SE NJIMA DETALJNO POZABAVIM

NAIME, NAJBOLJE, PO ONOME STO SAM SHVATIO, KORISTITI MEDIA QUERIES, ZAJEDNO SA NA PRIMER FLEXBOX-OM; I NEKIM DRUGIM STVARIMA, KOJE PLANIRAM DA NAUCIM (CSS GRID)

MEDIA QUERIES

MEDIJSKI UPITI (MEDIA QUERIES) SU U SRCU RESPONSIVE DIZAJNA. OMOGUĆAVAJU VAM DA CILJAJU ODREĐENE VELIČINE EKRANA I NAVODE ŠIFRE KOJE SE MOGU POKRENUTI SAMO NA UREĐAJIMA

NAJPOPULARNIJI OBLIK U KOJEM SE KORISTE MEDIJSKI UPITI JE NEŠTO ŠTO SE ZOVE @media PRAVILA

IZGLEDA OVAKO:

```
@media screen and (max-width: 300px) {  
    /*U OVOM BLOKU SE PISE CSS CODE*/  
}
```

GLEDAJUĆI U CODE SA SLIKE GORE, MOGU SKORO DA POGODIM, STA TAJ CODE, USTVARI RADI "ZA EKRAN SA MAKSIMALNOM ŠIRINOM OD 300px ... DO THIS AND THAT"

SVI STILOVI UNUTAR, POKAZANOG BLOKA CODE-A ĆE SE PRIMJENJIVATI SAMO NA UREĐAJE KOJI ODGOVARAJU EKSPRESIJI: screen and (max-width: 300px)

PRETPOSTAVLJAM DA JE OVO POMOGLO U OTKLANJANJU, NEKIH NEDOUUMICA

BRZA VEZBA

POKUSACU DA NA PRIMERU IZ PROSLE LEKCIJE PRIMENIM MEDIA QUERIES

ALI TO CU URADITI KASNIJE

NEKI RESURSI, KOJI SU DOBRI ZA VEZBANJE I USAVRSAVANJE FLEXBOX UMECA

<http://flexboxfroggy.com/>

OVDE MOGU VIDETI, KOJI BROWSERI PODRZAVAJU FLEXBOX

<https://caniuse.com/>

CSS GRID

CSS GRIDIMA PREDNOST, U ODNOSU NA FRAMEWORS, KAO STO JE BOOTSTRAP (NECU ULAZITI U TO ZASTO, JER SE NISAM BAVIO BOOTSTRAP-OM)

GRID MODEL, JESTE SISTEM KOJI SE SASTOJI OD HTML ELEMENATA, GRUPISANIH U REDOVE I KOLONE

KAKO BI NEKI ELEMENT UCINIM GRID CONTAINER-OM, JA MU MORAM DEFINISATI `display` PROPERTI, SA VREDNOSCЮ `grid`

KREIRACU JEDAN PRIMER

PRVO CU DEFINISATI HTML

```
<div class="container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

ZATIM CU DO ODREDJENOOG NIVOA STILIZOVATI, PRIKAZANE ELEMENTE, ALI I `html` I `body` ELEMENTE

```
.container > div:nth-child(1n) {
  background-color: #96ceb4;
}

.container > div:nth-child(3n) {
  background-color: #88d8b0;
}

.container > div:nth-child(2n) {
  background-color: #ff6f69;
}

.container > div:nth-child(4n) {
  background-color: #ffcc5c;
}
```

```
html, body {
  margin: 10px;
  background-color: #ffeedad;
}
```



UCINICU SADA DA SVAKI, NESTED DIV ELEMENT (BUDE I FLEX CONTAINER (UZ TO DODATNO STILIZOVATI, NESTED DIV-OVE, KOJI SU)

```
.container > div {  
    font-size: 2em;  
    color: #ffeed;  
  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}
```



ONO STO CU SADA URADITI, JESTE DA CU DIV ELEMENT, U KOJI SU NESTED OSTALI DIV-OVI (FLEX CONTAINER-I), ODNOSNO ELEMENT, KOJI IMA class ATRIBUT SA VREDNOSCЮ container, UCINITI GRID CONTAINER-OM; TAKO STO CU ZA NJEGA DEFINISATI display PROPERTI, SA VREDNOSCЮ grid

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

OVO ZNACI DA JE DIV ELEMENT STILIZOVAN .container CSS KLASOM, USTVARI POSTAO GRID CONTAINER; A DA SU SVA NJEGOVA DECA (CHILDREN), USTVARI GRID ITEM-I

KORISCENJEM PROPERTIJA grid-template-columns, MOGU DEFINISATI DA JE, MOJ GRID PODELJEN U ZELJENI BROJ KOLONA, SA ZELENIM VREDNOSTIMA SIRINA, ZA SVAKU KOLONU

```
.container {  
    display: grid;  
  
    grid-template-columns: 100px auto 100px;  
}
```

NAIME POSMATRAJUCI VREDNOST grid-templates-column PROPERTIJA, RECI CU SLEDECE:

GRID CE SE SASTOJATI IZ TRI KOLONE

PRVA I TRECA KOLONA CE IMATI SIRINU OD 100px, DOK CE DRUGA KOLONA IMATI AUTOMATSKI DODELJENU SIRINU, KOJA CE OBUHVATATI, SAV, PREOSTALI PROSTOR; ZATO STO SAM TOJ KOLONI KAO VREDNOST SIRINE, DEFINISAO VREDNOST auto

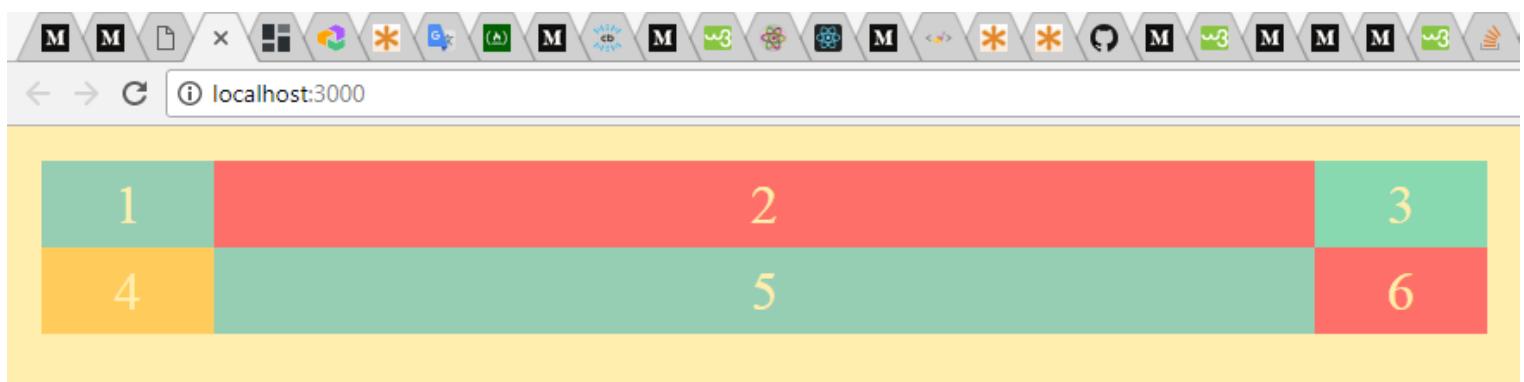


ONO STO MOGU DA PRIMETIM SA SLIKE GORE, JESTE DA SE GRID SASTOJI IZ 3 KOLONE I DVA REDA

POSTOJI, JOS JEDAN PROPERTI, KOJI SE ZOVE grid-template-rows, POMOCU KOJEG MOGU DEFINISATI I ZELJENI BROJ REDOVA; ALI I VISINU REDA; I TO NA ISTI NACIN KAO STO SAM DEFINISAO VREDNOST ZA grid-template-columns PROPERTI

DEFINISACU DA VISINA, OBA REDA SA SLIKE GORE BUDE PO 50px

```
.container {  
    display: grid;  
    grid-template-columns: 100px auto 100px;  
    grid-template-rows: 50px 50px;  
}
```



OPET CU SE VRATITI NA BAVLJENJE PROPERTIJIMA grid-template-rows i grid-template-columns; KONKRETNO, KAKO BI SE MALO POIGRAO POMENUTIM PROPERTIJIMA, I UKLONIO NEKE NEDOUMICE VEZANE ZA POMENUTE PROPERTIJE; ALI PRE TOGA CU SE POZBAVITI, JOS JEDNIM PROPERTIJEM:

grid-gap (**SHORTHAND PROPERTI ZA** grid-column-gap | grid-row-gap)

POMOCU, POMENUTOG PROPERTIJA, MOGU DEFINISATI RAZMAK IZMEDJU GRID ITEM-A

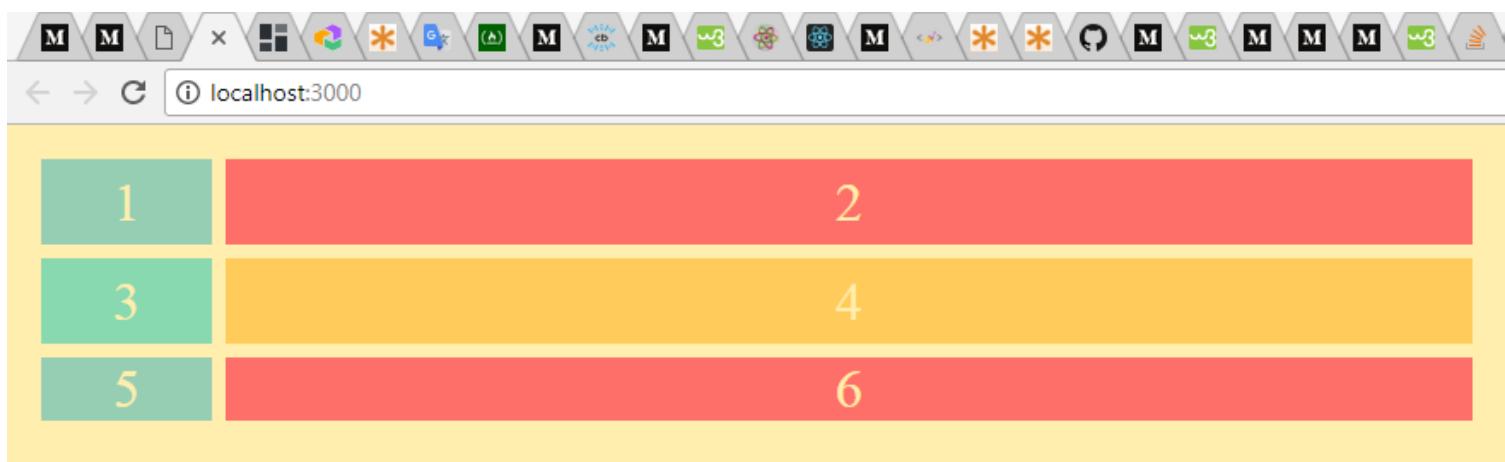
```
.container {  
    display: grid;  
    grid-template-columns: 100px auto 100px;  
    grid-template-rows: 50px 50px;  
    grid-gap: 8px;  
}
```



SADA CU SE POIGRATI PROPERTIJIMA grid-template-rows i grid-template-columns

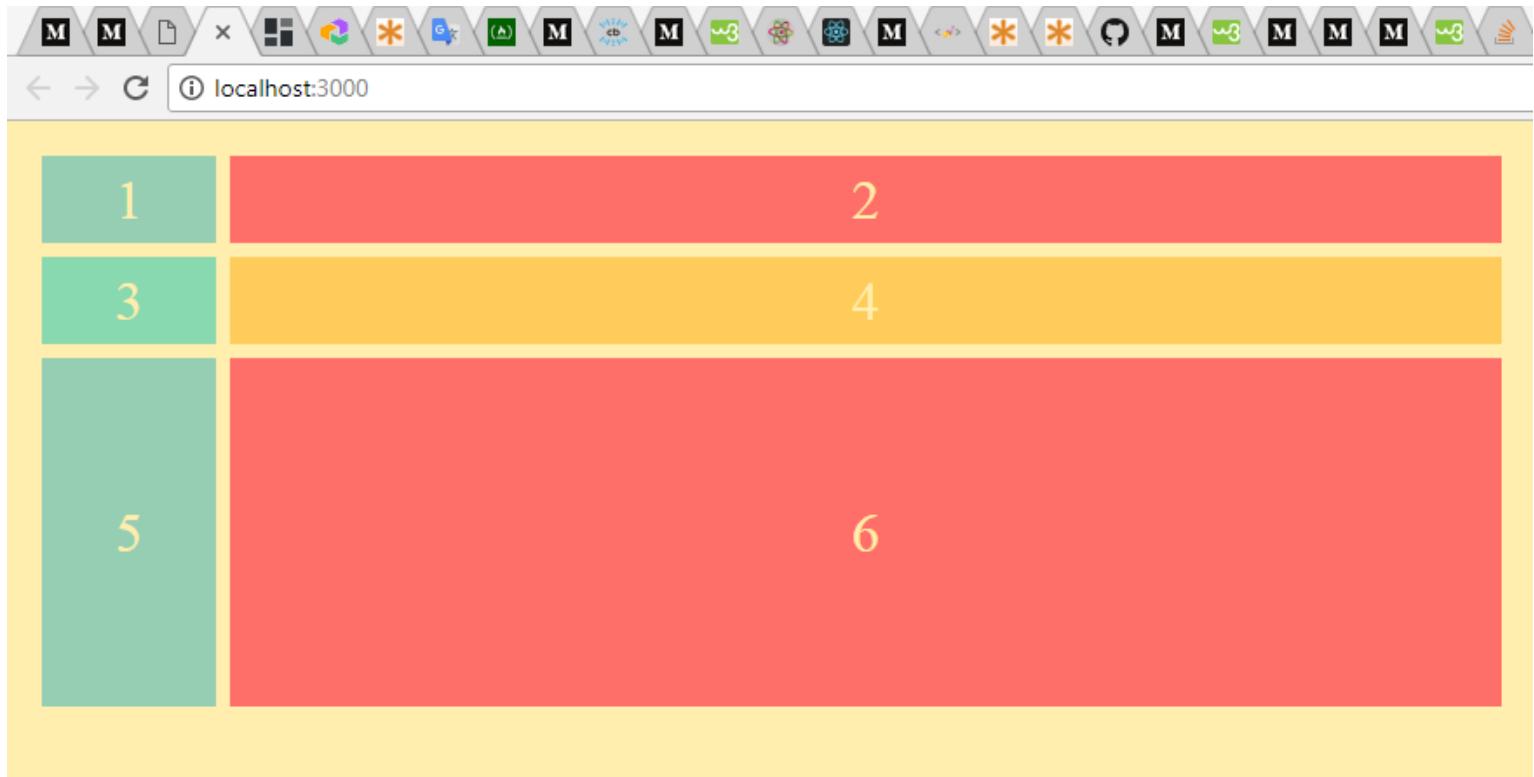
DEFINISACU DA SE GRID SASTOJI OD DVE KOLONE

```
.container {  
    display: grid;  
    grid-template-columns: 100px auto;  
    grid-template-rows: 50px 50px;  
    grid-gap: 8px;  
}
```



KAO STO VIDIM, PRVA DVA REDA IMAJU ISTU VISINU, JER JE TAKO ZAISTA DEFINISANO grid-template-rows PROPERTIJEM; A POSTO ISTIM PROPERTIJEM VISINA TRECEG REDA NIJE DEFINISANA, ON JE DOBIO AUTOMATSKE VISINU; SADA CU DEFINISATI VISINU I TRECEG REDA

```
.container {  
    display: grid;  
    grid-template-columns: 100px auto;  
    grid-template-rows: 50px 50px 200px;  
    grid-gap: 8px;  
}
```



ONO STO SAM ZAKLJUCIO U POGLEDU POMENUTA DVA PROPERTIJA, ALI I U POGLEDU OSOBINA GRIDA JESTE SLEDECE:

“KOLONE SU JACE OD REDOVA”

NAIME, **BROJ DEFINISANIH KOLONA MOZE MODIFIKOVATI, KOLIKI BROJ REDOVA MOZE POSTOJATI, A NIKAKO OBRATNO**

SADA CU I TESTIRATI, POMENUTU TVRDNJU; TAKO STO CU DEFINISATI, DA POMENUTI GRID SISTEM IMA I JOS JEDAN RED (DAKLE CETIRI REDA)

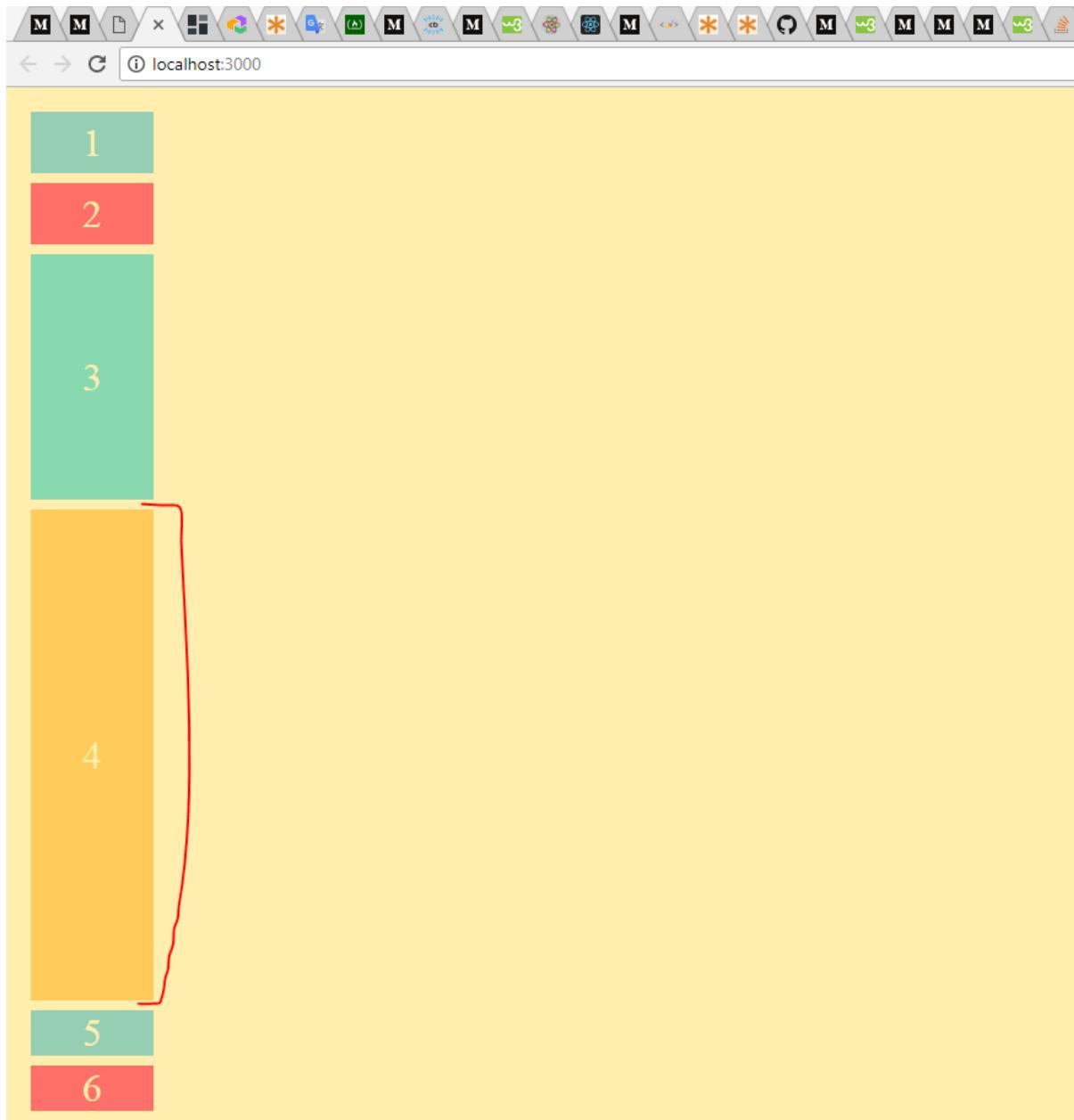
```
.container {  
    display: grid;  
    grid-template-columns: 100px auto;  
    grid-template-rows: 50px 50px 200px 400px;  
    grid-gap: 8px;  
}
```

DODAVANJEM VREDNOST, KOJU SAM OZNACIO NA SLICI GORE; NECE SE NISTA PROMENITI U GRIDU

ALI DA SAM SMANJIO BROJ KOLONA

```
.container {  
  display: grid;  
  grid-template-columns: 100px auto;  
  grid-template-rows: 50px 50px 200px 400px;  
  grid-gap: 8px;  
}
```

```
.container {  
  display: grid;  
  grid-template-columns: 100px;  
  grid-template-rows: 50px 50px 200px 400px;  
  grid-gap: 8px;  
}
```



I CETVRTI RED JE ZAISTA VISOK 400 PIKSELA

OSTALA DVA REDA, ODNOSNO PETI I SESTI, IMAJU AUTOMATSKE VIINE, BAZIRANE NA NJIHOVOJ SADRZINI

ZAPAZANJE NE VEZANO, KONKRETNO ZA GRID

SAMO JEDNO ZAPAZANJE, KADA SE DEFINISE STIL, PUTEM style TAGA (NESTED U head TAGU), ILI PUTEM style ATRIBUTA U HTML, DAKLE INLINE STYLE; STRANICA SE MORA RELOAD-OVATI, DA BI SE STIL APPLICIRAO (APPLY), NA DEFINISANIM ELEMENTIMA

ALI TO NIJE SLUCAJ, KADA SE .css FAJL UCITVA U HTML STRANICU, TADA SE DOGADJA AUTOMATSKA PROMENA STILA NA STRANICI, BEZ IKAKVOG RELOAD-OVANJA STRANICE

FRACTION UNIT AND REPEAT (fr JEDINICA I repeat FUNKCIJA)

ONO STO CU SADA, NA POCETKU URADITI, JESTE REORGANIZACIJA CSS CODE-A, IZ PROSLE LEKCIJE NAIME, JA ZELIM DA SE U index.js FAJLU, SAMO NALAZI CSS CODE, KOJIM SAM JEDAN ELEMENT NACINIO GRID CONTAINER-OM (TO JE BILA DEFINICIJA STILOVA ZA, SAMO .container KLASU)

A DEFINISACU I JEDAN FAJL KOJI SE ZOVE basic.css U KOJEM CE SE NALAZITI SVI OSTALI STILOVI

```
html, body {  
    margin: 10px;  
    background-color: #fffeead;  
}  
  
.container > div {  
    font-size: 2em;  
    color: #fffeead;  
  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.container > div:nth-child(1n) {  
    background-color: #96ceb4;  
}  
  
.container > div:nth-child(3n) {  
    background-color: #88d8b0;  
}  
  
.container > div:nth-child(2n) {  
    background-color: #ff6f69;  
}  
  
.container > div:nth-child(4n) {  
    background-color: #ffcc5c;  
}
```

```
.container {  
    display: grid;  
    grid-template-columns: 100px auto 100px;  
    grid-template-rows: 50px 50px;  
    grid-gap: 8px;  
}
```

```
<div class="container">  
    <div>1</div>  
    <div>2</div>  
    <div>3</div>  
    <div>4</div>  
    <div>5</div>  
    <div>6</div>  
</div>
```

basic.css

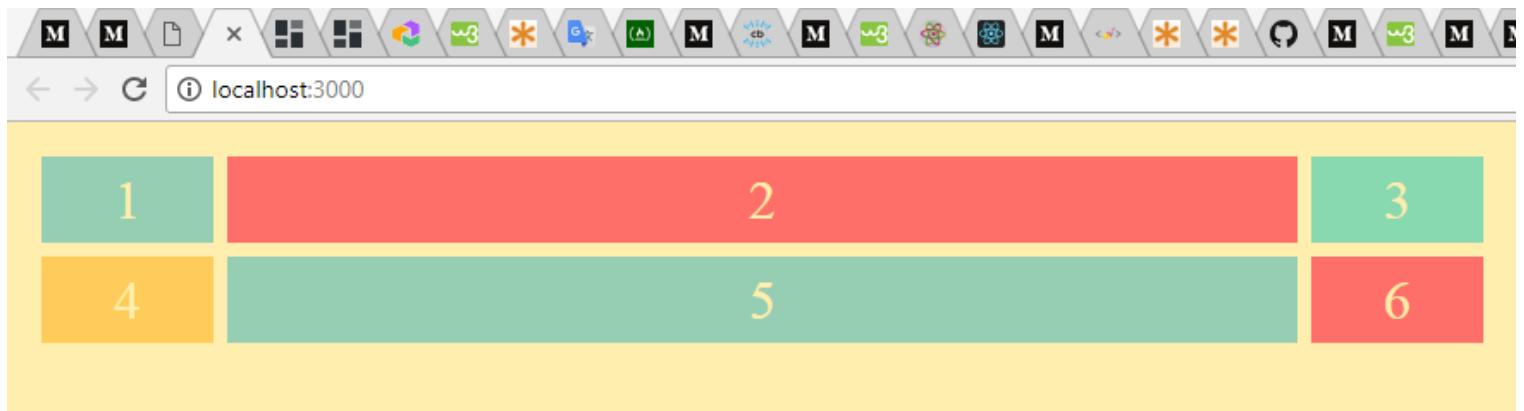
index.css

index.html

ALI NARAVNO, POSTO SE JA UPOZNAJEM SA GRIDOM, U OKVIRU REACT APLIKACIJE, JA U JAVASCRIPT FAJL MORAM DA IPORTUJEM I basic.css FAJL (index.js JE VEC IMPORTOVAN) (NECU GOVORITI ZASTO SE TO MORA URADITI, JER SAM VEC U DOKUMENTU, U KOJEM SAM SE BAVIO REACT-OM, TO VEC NAPOMENUO)

```
import React from 'react';  
import ReactDOM from 'react-dom';  
import './index.css';  
  
import './basic.css';
```

index.js



NAIME, OVAJ GRID JESTE RESPONSIVE, JER PROMENOM VELICINE BROWSCERA, DRUGA KOLONA SE POVECAVA ILI SMANJUJE PO SIRINI (TO JE UPRAVO JER DRUGA KOLONA, ZA SVOJU SIRINU, KAO VREDNOST IMA auto)

ONO STO SADA ZELIM DA URADIM, JESTE DA DODAM "DODATNI" RESPONSIVENESS, POMENUTOM GRID-U

ODNOSNO, JA ZELIM DA SE SIRINA SVE TRI KOLONE JEDNAKO SMANJUJE I POVECAVA, PROMENOM VELICINE PROZORA BROWSCERA

NAIME, JA MOGU SVAKOJ KOLONI ZA SIRINU DODELITI auto KAO VREDNOST

```
.container {  
    display: grid;  
    grid-template-columns: auto auto auto;  
    grid-template-rows: 50px 50px;  
    grid-gap: 8px;  
}
```

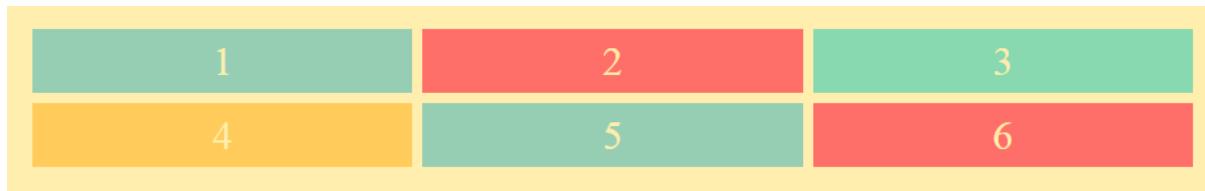
OVIM BIH POSTIGAO ONO STO SAM NAUMIO, ALI POSTOJI BOLJE RESENJE; A ONO SE OGLEDU U SLEDECSEM

NAIME, KORISTIO BIH POSEBAN TIP UNITA (JEDINICE MERE), KOJI SE ZOVE **FRACTAL** ILI FRAKTAL JEDINICA; A KORISTI SE TAKO STO SE NAPISE BROJ I UZ NJEGA fr

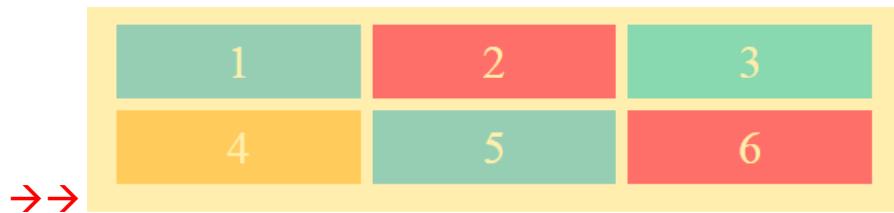
```
.container {  
    display: grid;  
    grid-template-columns: 1fr 1fr 1fr;  
    grid-template-rows: 50px 50px;  
    grid-gap: 8px;  
}
```

IVO STO SAM OZNACIO NA GORNJOJ SLICI, MOGU DA INTERPRETIRAM NA SLEDECI NACI:

GRID JE PODELJEN U TRI KOLONE; A UKUPNA SIRINA, SVIH KOLONA, ZAJEDNO SE MOZE POSMATRATI KAO TRI TRECINE (3/3; CELA SIRINA); DAKLE, SVAKOJ KOLONI PRIPADA, JEDNAKA JEDNA TRECINA OD UKUPNE SIRINE, SVIH KOLONA



→→SMANJUJEM
WINDOW→→



I NA BILO KOJOJ SIRINI PROZORA, SVE KOLONE GRIDA SU RESPONSIVE, I SVE NJEGOVE KOLONE SU UVEK ISTE SIRINE

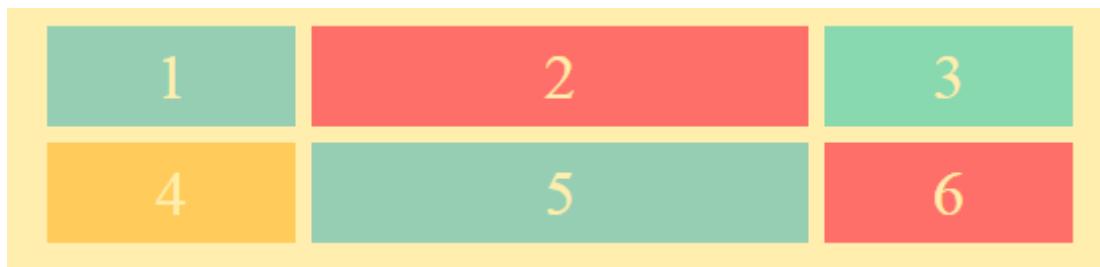
MOGAO SAM DEFINISATI I NEKU DRUGU "FRAKTALNU" VREDNOST, SIRINE KOLONA

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 50px 50px;
    grid-gap: 8px;
}
```

IVO STO SAM DEFINISAO I OZNACIO NA GORNJOJ SLICI, MOGU INTERPRETIRATI NA SLEDECİ NACI:

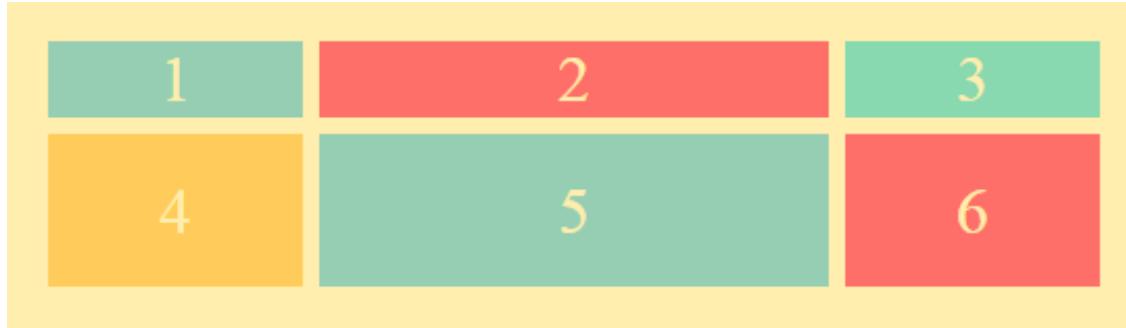
GRID JE PODELJEN U TRI KOLONE; A UKUPNA SIRINA, SVIH KOLONA, ZAJEDNO SE MOZE POSMATRATI KAO CETIRI CETVRTINE (4/4; CELA SIRINA); S TIM STO PRVOJ I TRECOJ KOLONI PRIPADAJU, PO JEDNA CETVRTINA (1/4) DOSTUPNE SIRINE; A DRUGOJ KOLONI PRIPADAJU DVE CETVRTINE (2/4); ODNOŠNO JEDNA POLOVINA (1/2) OD UKUPNE DOSTUPNE SIRINE

TO SE I JASNO VIDI NA SLEDECOJ SLICI



NA ISTI NACIN SAM MOGAO DEFINISATI I VISINE KOLONA

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 1fr 2fr;
    grid-gap: 8px;
}
```

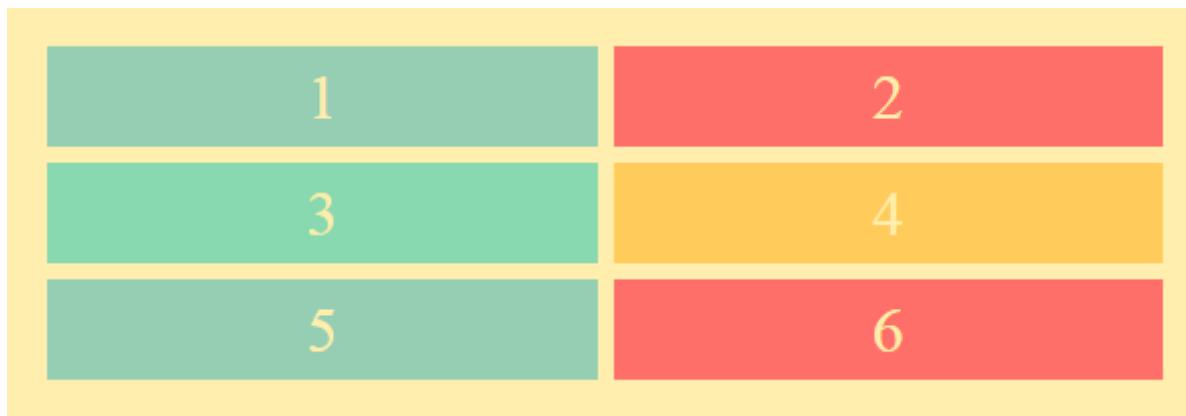


MEDJUTIM, MOGUĆE JE JOS JEDNOSTAVNIJE, PISANJE, POMENUTIH VREDNOSTI, POMENUTIH PROPERTIJA; A TO JE NAIME

repeat FUNKCIJA

CIJE POZIVANJE SE DODAJE KAO VREDNOST, POMENUTIM PROPERTIJIMA; A NJEN PRVI ARGUMENT JESTE BROJ KOLONA ILI REDEOVA (STO SE JOS NAZIVAI MOUNT-OM KOLONA I REDOVA), A DRUGI ARGUMENT JESTE VREDNOST SIRINE, BILA ONA U FRKTALIMA, ILI BILO KOM DRUGOM UNIT-U

```
.container {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    grid-template-rows: repeat(3, 50px);
    grid-gap: 8px;
}
```



MEDJUTIM, POSTOJI JOS JEDNOSTAVNIJI NACIN DEFINISANJA REDOVA I KOLONA; A TO JE SHORTHAND PROPERTI, KOJI ZAMENJUJE, UPOTREBU grid-template-rows i grid-template-columns PROPERTIJA; I TAJ PROPERTI SE ZOVE:

grid-template

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 50px);
    grid-gap: 8px;
}
```

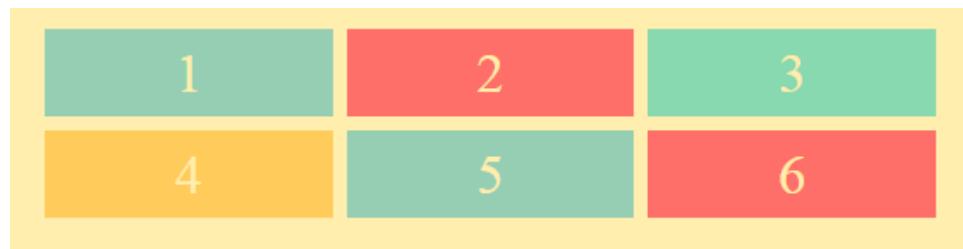
DAKLE, OVO OVDE (KOJE CU I UKLONITI)

JE ISTO KAO I

OVO OVDE

```
.container {  
    display: grid;  
    grid-template: repeat(2, 50px) / repeat(3, 1fr);  
    grid-gap: 8px;  
}
```

NE ZABORAVI DA SE U SLUCAJU VREDNOSTI, PROPERTIJA (OZNACENOGR CRVENOM NA GORNJOJ SLICI) grid-template; PRVO PISU VREDNOSTI ZA REDOVE, PA ZNAK "KROZ", PA VREDNOSTI ZA KOLONE



POZICIONIRANJE GRID ITEM-A

ONO CIME CU SE SADA POZABAVITI, JESTE UCENJE, KAKO MOGU IZVRISTI ADJUSTEMENT-E, U POGLEDU POZICIONIRANJA I VELICINE GRID-ITEM-A

STO JE KLJUCNO (CRUCIAL) STVAR, KOJU TREBA POZNAVATI, KAKO BIH MOGAO KREIRATI REAL-WOLD WEBSITES SA CSS GRID-OM

KREIRACU, ZATO MARKUP ZA WEBSITE

```
<div class="container">
  <div class="header">HEADER</div>
  <div class="menu">MENU</div>
  <div class="content">CONTENT</div>
  <div class="footer">FOOTER</div>
</div>
```

OVO SA SLIKE GORE, JESTE NARAVNO NESTED U body TAGU; A U head TAGU MOGU DA BUDU NESTED link TAGOVI, KOJIMA SE UCITAVAJU FAJLOVI index.css I basic.css

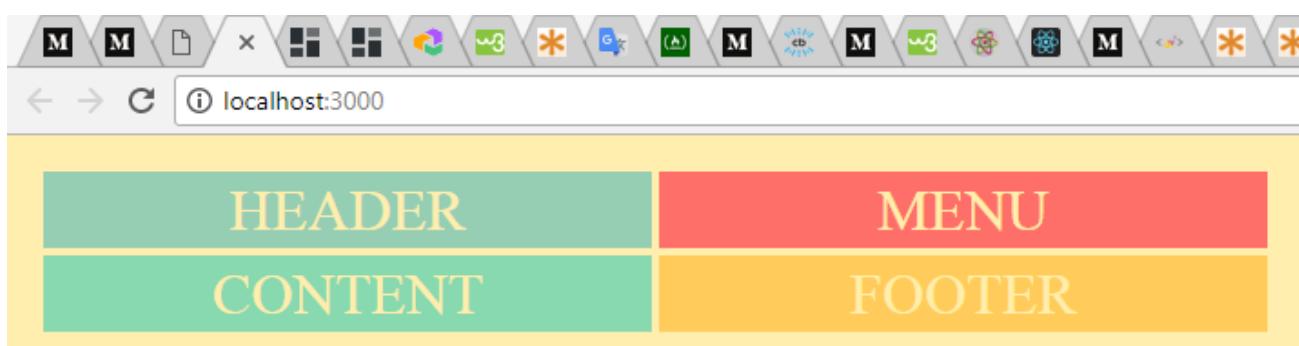
MEDJUTIM, POSTO SVE OVO PISEM U FAJLOVIMA REACT APLIKACIJE, JA CU IMPORTOVATI CSS MODULE U JAVASCRIPT FAJL (NISTA OD TOGA NECU POKAZIVATI, JER VEC JESAM U PROSLOJ LEKCIJI), UMESTO DA IH DIREKTNO UCITAM U HTML

U basic.css FAJLU JE CODE, KAKAV JE BIO I U PROSLOJ LEKCIJI

A SADA CU DEFINISATI CODE index.css FAJLA, U KOJEM CU DEFINISATI STILOVE ZA .container KLASU

```
.container {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-rows: 42px 42px;
  grid-gap: 4px;
}
```

ZA POCETAK CE GRID IZGLEDATI OVAKO



GRID JE TAKODJE RESPONSIVE

ONO STO ZELIM, U POGLEDU WEB STRANICE, JESTE DA IMAM TRI REDA

I DA, SAMO HEADER ZAUZIMA PRVI RED

U DRUGOM REDU BI SE NALAZILI MENU I CONTENT

A U TRECEM, SAMO FOOTER

SADA CU REDEFINISATI grid-template-rows VREDNOST, JER ZELIM DA POKUSAM DA DEFINISEM DA GRID IMA TRI REDA, OD KOJIH CE DRUGI RED IMATI VISINU OD 200px

```
.container {  
    display: grid;  
    grid-template-columns: repeat(2, 1fr);  
    grid-template-rows: 42px 200px 42px;  
    grid-gap: 4px;  
}
```

MEDJUTIM, OVIM NECU POSTICI DA GRID IMA TRI REDA, SAMO CU POSTICI DA VISINA DRUGOG REDA BUDE 200px; NAIME MOJI GRID ITEMI, ZAUZIMAJU SAMO 4 CELIJE GRIDA, I UPRAVO ZATO, NECE POSTOJATI TRECJI RED



NAIME, SADA CU TARGETOVATI, SAMO HEADER; DAKLE SAMO PRVI GRID ITEM, A ONO STO ZELIM DA POSTIGNEM DA ON OBUHVATA ILI DA SE PROSTIRE OD LEFT-HANDSIDE-A (LEVE STRANE), SKROZ DO RIGHT-HAND-SIDE-A (DESNE STRANE), STO BI ZNACILO DA POMENUTI GRID ITEM, MORA DA SE PROSTIRE, KROZ CELI PRVI RED, ODNOSNO DA OBUHVATA CELI PRVI RED, GDE BI NORMALNO BILA DVA ELEMENTA, ODNOSNO "DVA POCKETA, DVE KOLONE"

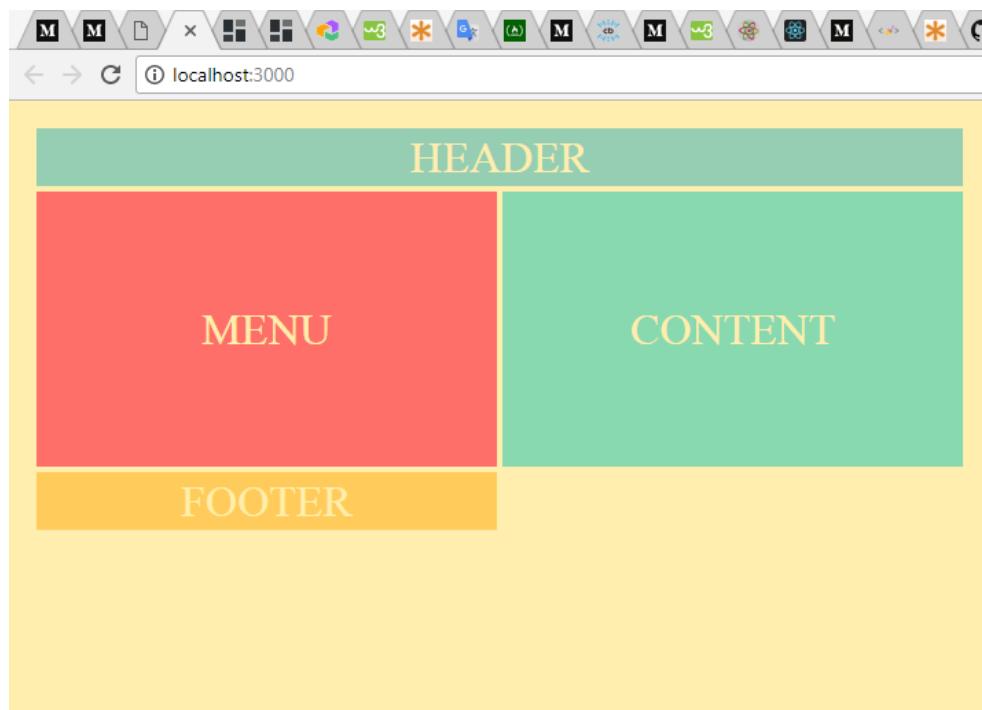
DAKLE, SADA DEFINISEM, NOVE PROPERTIJE, SAMO ZA HEADER ELEMENT, KOJI JE GRID ITEM

TO CE, NAIME BITI PROPERTIJI:

grid-column-start grid-column-end

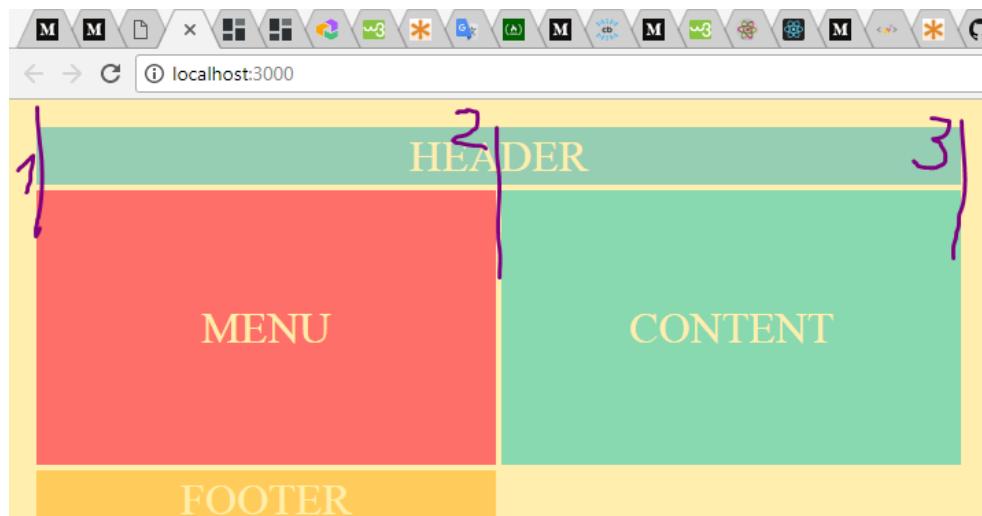
NAIME, POMENUTI PROPERTIJI DEFINISU OD KOJE LINIJE KOLONE, TREBA DA POCINJE, A DO KOJE LINIJE KOLONE SE ZAVRSAVA, GRID ITEM, KOJI NA OVAJ NACIN MOZE DA SE PROSTIRE PREKO VISE KOLONA; I DA ZAUZIMA VISE MESTA U JEDNOM REDU

```
.header {  
    grid-column-start: 1;  
    grid-column-end: 3;  
}
```



ONO STO SAM UNOSIO, KAO VREDNOSTI, POMENUTIH PROPERTIJA JESTE BIO BROJ LINIJE KOLONE; ODNOSNO COLUMN LINE NUMBER

UMESTO DA OBJASNJAVAAM, GDE SE NALAZE, TE LINIJE KOLONA ILI COLUM LINES, OZNACICU IH



A KAO STO VIDIM, HEADER ELEMENT SE, UPRAVO SADA PROSTIRE OD PRVE LINIJE KOLONE, DO TRECE LINIJE KOLONE, I NA TAJ NACIN JE "ISTISNUO" MENU ELEMENT, KOJI SADA PRIPADA DRUGOM REDU, A ZBOG TOGA JE FOOTER "ISTISNUT" U TRECI RED

MEDJUTIM JA SAM MOGAO I U OVOM SLUCAJU DA KORISTIM SHORTHAND PROPERTY, KOJI SE ZOVE:

grid-column

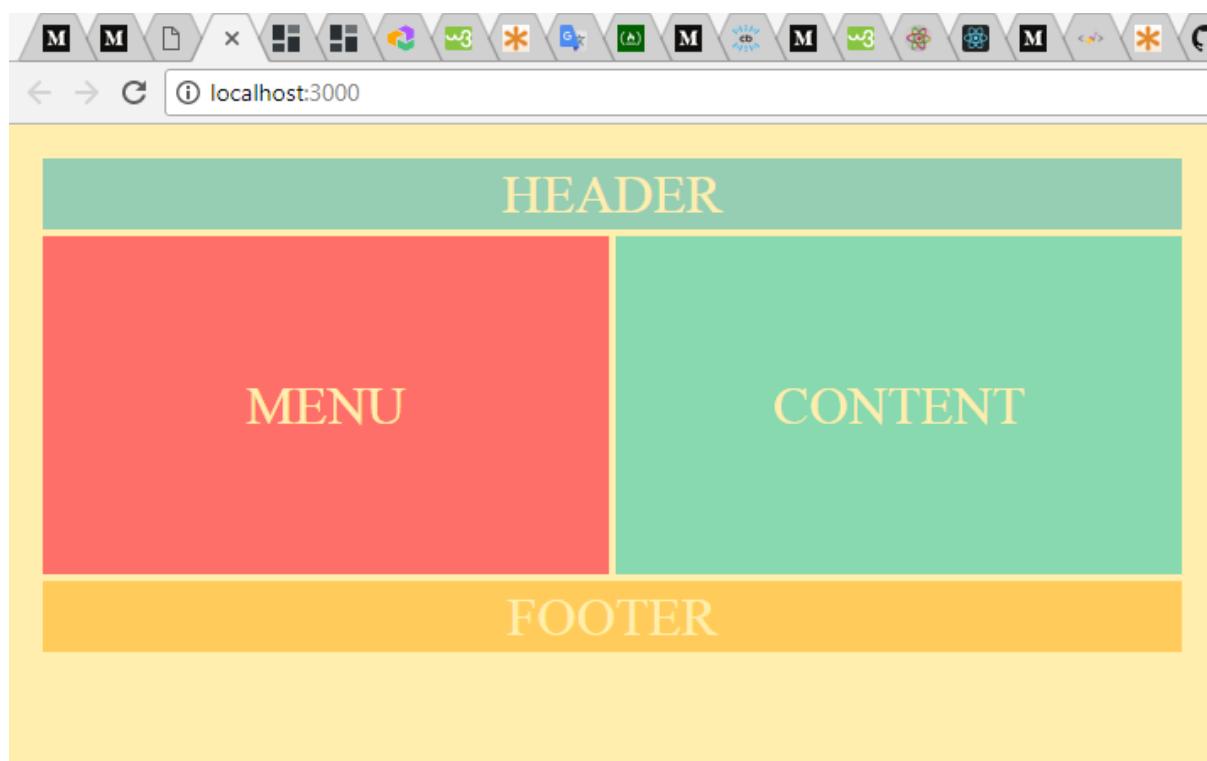
NA SLEDECINI NACIN:

```
.header {  
    grid-column-start: 1;  
    grid-column-end: 3;  
}  
  
→→→→ .header {  
    grid-column: 1 / 3;
```

KAO STO VIDIM I OVDE SE KORISTI "KROZ" ZNAK, KAO I KOD ODREDJENOG POMENUTOG U PREDHODNOJ LEKCIJI

SADA CU DA DEFINISEM, POMENUTI PROPERI I ZA FOOTER ELEMENT, ODNOSNO GRID ITEM

```
.footer {  
    grid-column: 1 / 3;  
}
```

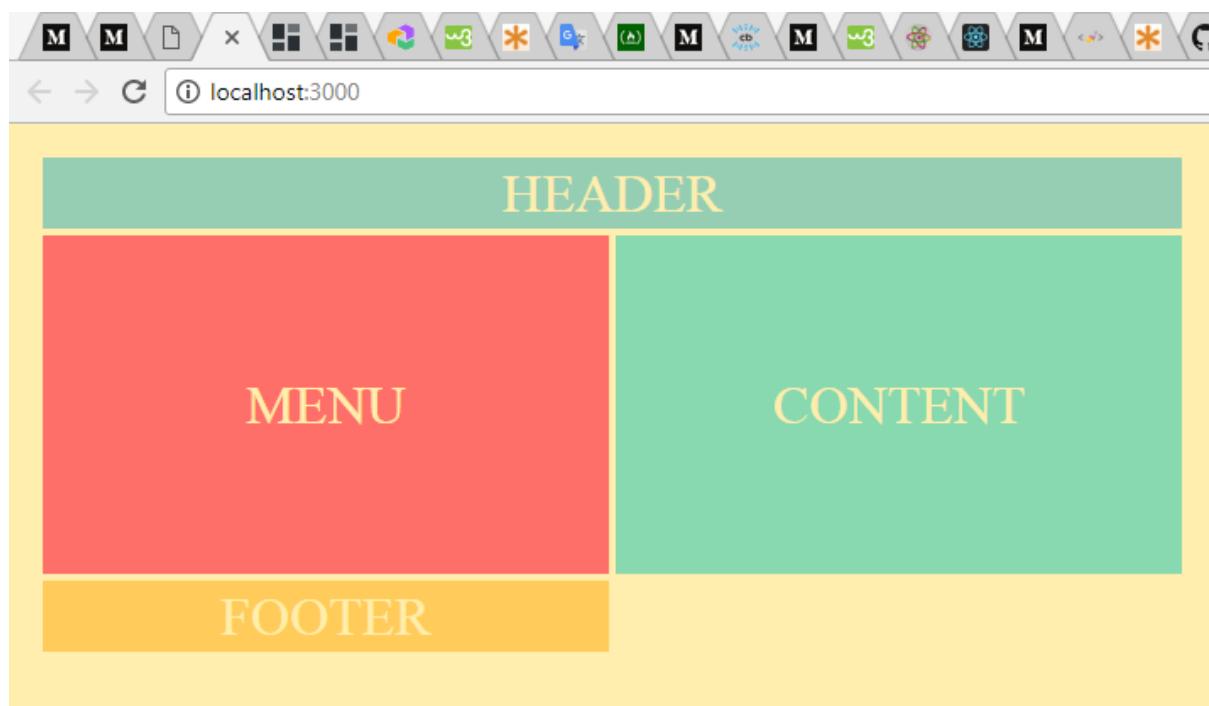


MEDJUTIM, POSTOJI JOS JEDAN NACIN DEFINISANJA VREDNOSTI, POMENUTOG grid-column PROPERTIJA

TAJ NACIN CU POKAZATI, ISTO ZA FOOTER ELEMENT

```
.footer {  
    grid-column: 1 / span 1;  
}
```

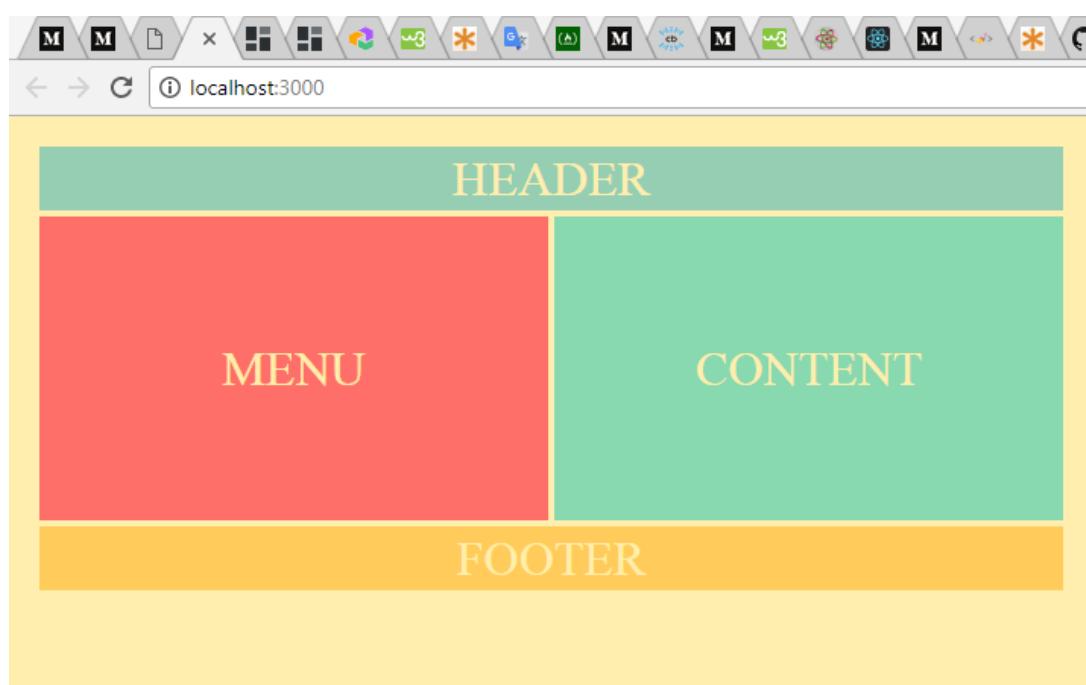
PRIKAZANO span SA SLIKE GORE I BROJ 1 PORED NJEGA,ZNACILI BI PROSTIRI (BUDI U RASPONU) SE KROZ JEDNU KOLONU; A 1 LEVO OD ZNAKA "KROZ", I DALJE JESTE BROJ LINIJE KOLONE



KAO STO VIDIM FOOTER SADA ZAISTA POCINJE OD PRVE LINIJE KOLONE I PROSTIRE SE KROZ JEDNU KOLONU

DEFINISACU, DA SE, FOOTER, OPET PROSTIRE, KROZ DVE KOLONE

```
.footer {  
    grid-column: 1 / span 2;  
}
```

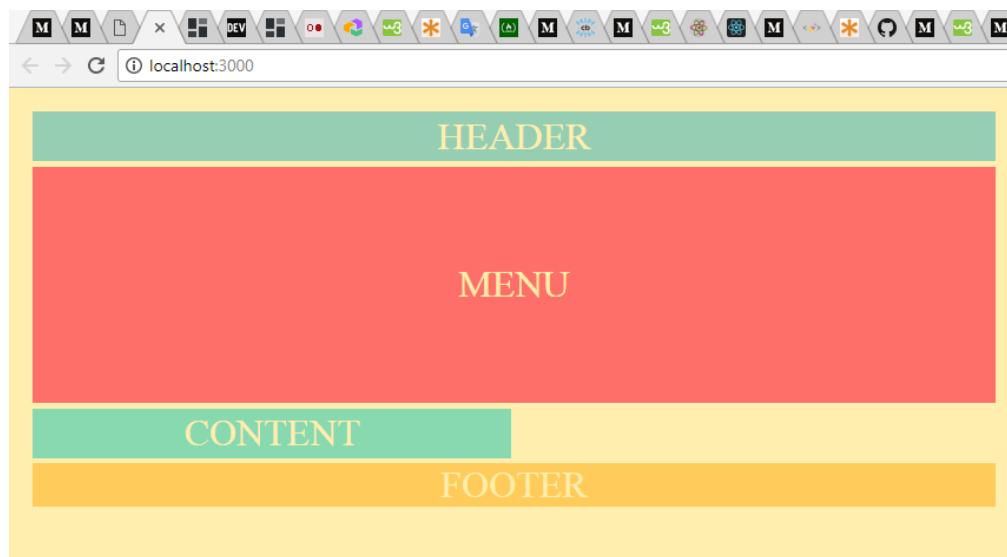


ALI, POSTOJI I JOS JEDAN NACIN

A TO JE DA SE ZA grid-column-end DODELI VREDNOST -1

TIME BI SE GRID ITEM PROSTIRAO, KROZ ONOLIKO KOLONA, KOLIKO IH JE DOSTUPNO, ODNOSNO ONOLIKO KOLIKO IH POSTOJI OD POCETKA, TOG GRID ITEMA, TJ PROSTIRAO BI SE DO KRAJA DESNE STRANE GRID-A (JER VREDNOST -1 SE USTVARI "ODNOSI" NA, POSLEDJI MOGUCI COLUMN LINE)

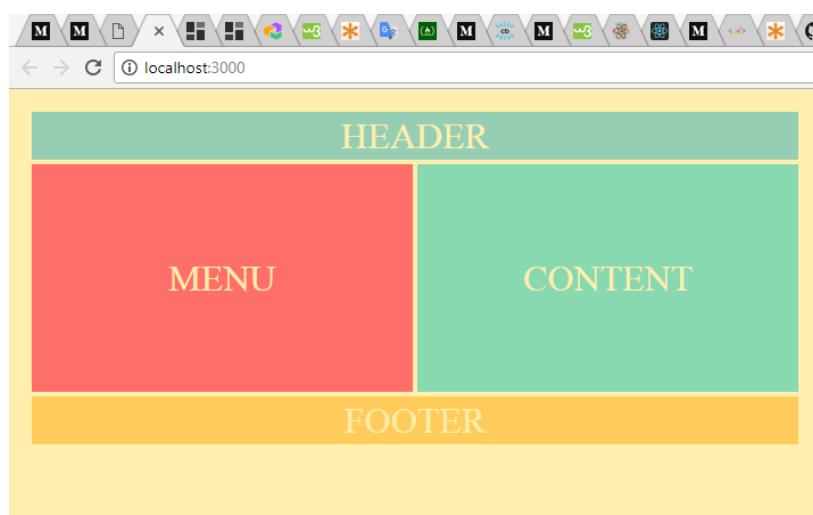
```
.menu {  
    grid-column: 1 / -1;  
}
```



UKLONICU ONO STO SAM DEFINISAO ZA MENU, JER NE ZELIM DA SE MENU PROSTIRE, KAKO JE GORE PRIKAZANO; A ONAKO KAKO JE BIO DEFINISAN MENU, APPLICIRACU ZA HEADER

TAKO DA ZA SADA IMAM OVAKVU SITUACIJU

```
.header {  
    grid-column: 1 / -1;  
}  
  
.footer {  
    grid-column: 1 / span 2;  
}
```

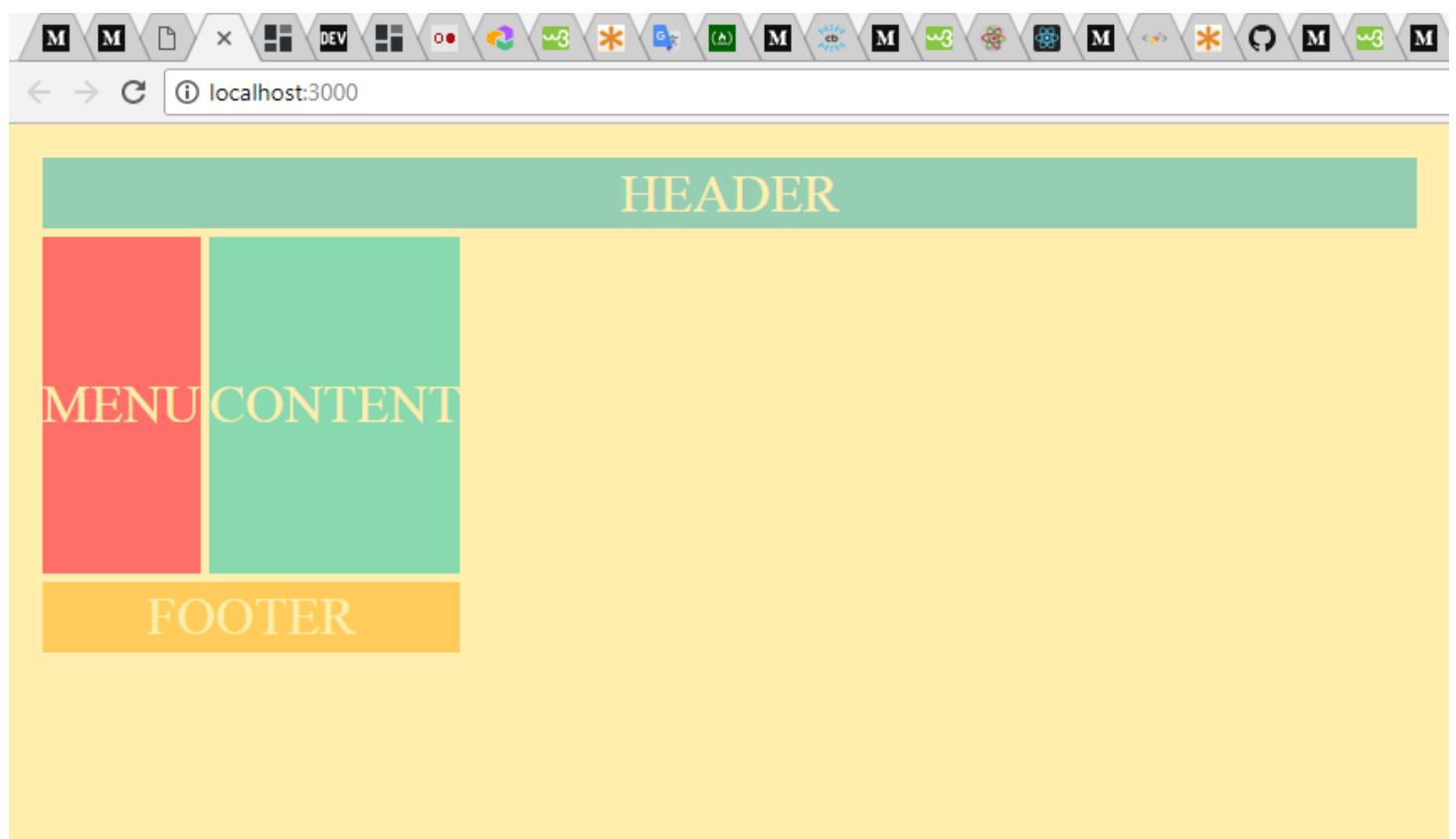


POMENUTU -1 VREDNOST JE DOBRO KORISTITI, KADA NE ZNAM OD KOLIKO KOLONA SE, USTVARI SASTOJI, MOJ GRID; NA TAJ NACIN DODELUJUJUCI -1, column-grid-column PROPERTIJU, ZAGARANTOVANO JE DA CE SE GRID ITEM, SPAN-OVATI, SVE DO "IVICE" GRID-A

KAKO BI TO POKAZAO, REDEFINISACU VREDNOST, BROJA KOLONA, KOJE IMA, MOJ GRID

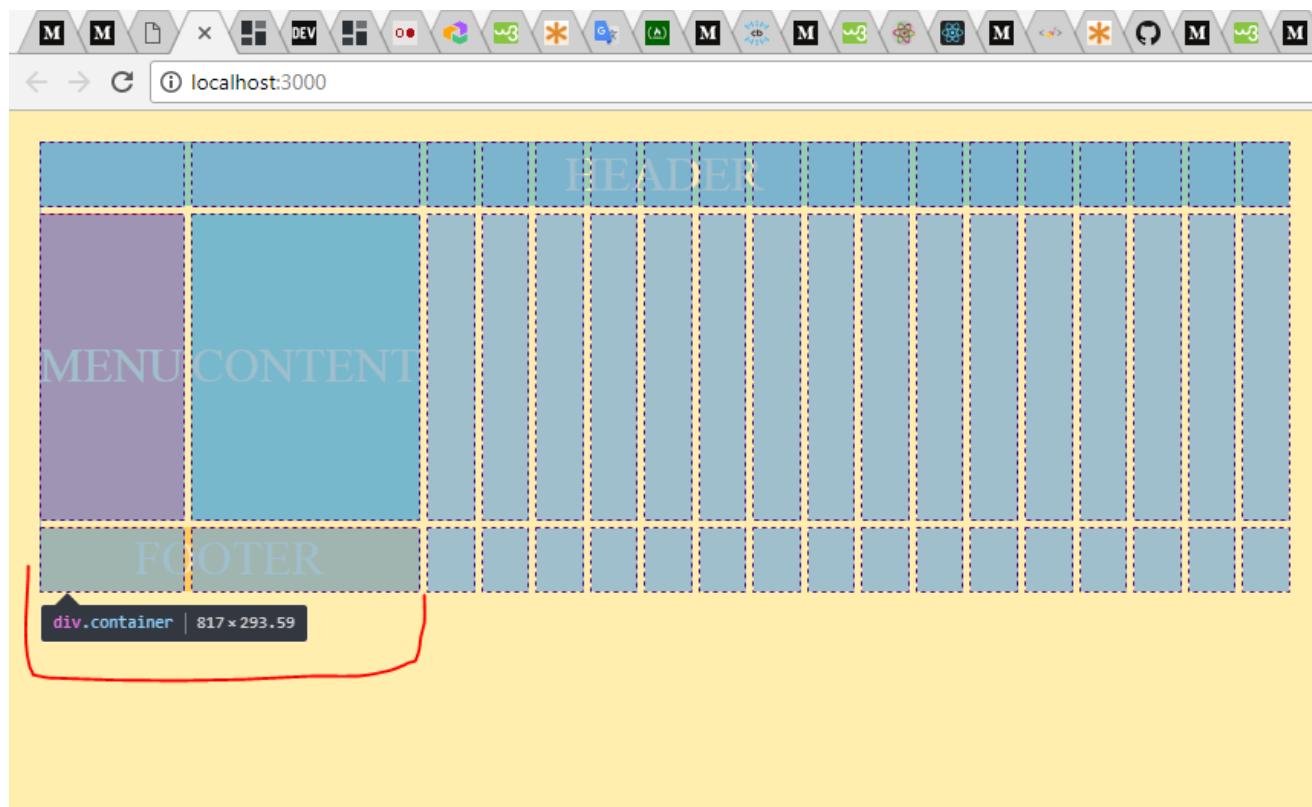
```
.container {  
    display: grid;  
    grid-template-columns: repeat(7, 1fr);  
    grid-template-rows: 42px 200px 42px;  
    grid-gap: 4.8px;  
}  
  
.header {  
    grid-column: 1 / -1;  
}  
  
.footer {  
    grid-column: 1 / span 2;  
}
```

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
    grid-template-rows: 42px 200px 42px;  
    grid-gap: 4.8px;  
}  
  
.header {  
    grid-column: 1 / -1;  
}  
  
.footer {  
    grid-column: 1 / span 2;  
}
```



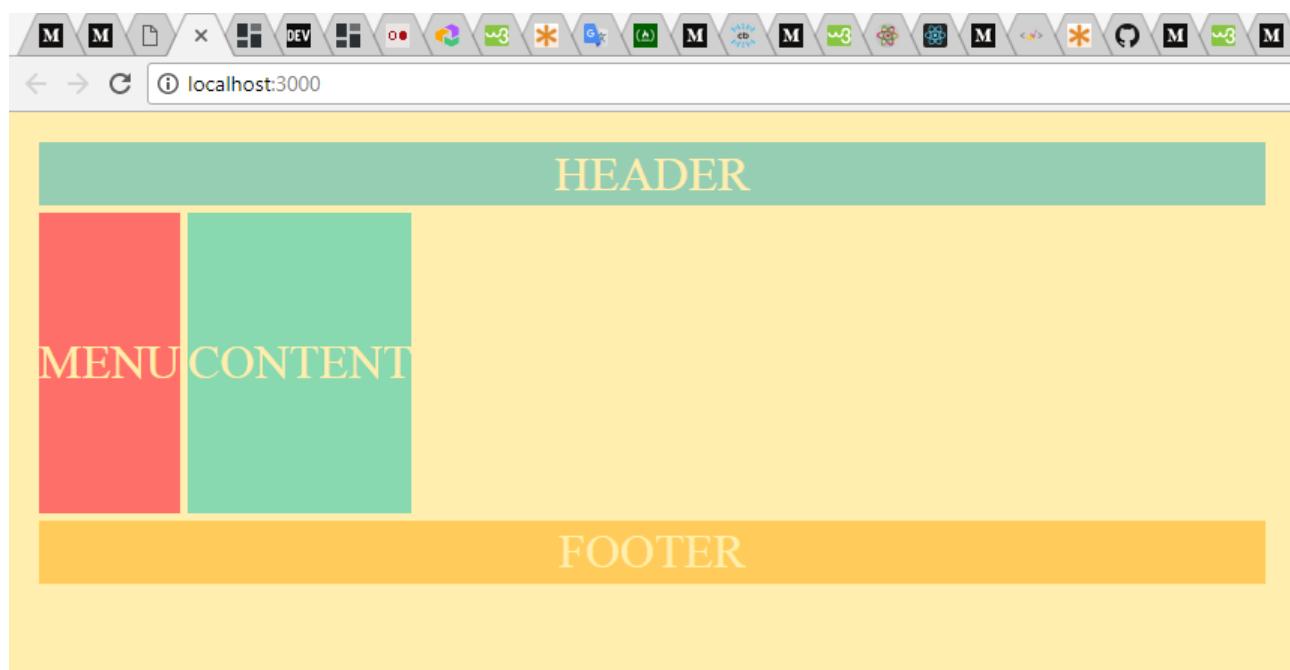
KAO STO VIDIM SA SLIKE GORE, HEADER SE SADA PROSTIRE, PREKO SVIH 18 KOLONA GRID-A (VREDNOST ZA grid-column-end JE -1); DOK SE FOOTER PROSTIRE, SAMO PREKO DVE KOLONE, JER SAM ZA NJEGOVU VREDNOST grid-column-end PROPERTIJA DEFINISAO span 2 VREDNOST

DA SE POMENUTI GRID ITEM PROSTIRE, SAMO PREKO DVE KOLONE, MOGU VIDETI I KADA U CHROME KONZOLI SELEKTUJEM GRID CONTAINER, CIME CE, CEO GRID POSTATI VIDLJIV



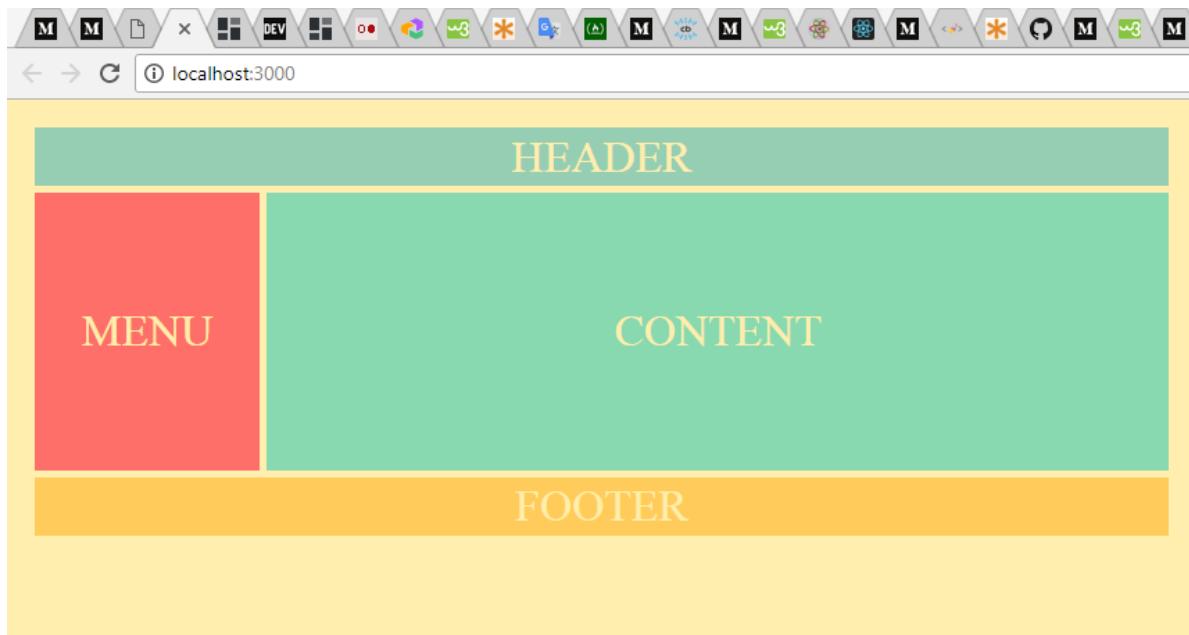
SADA CU DEFINISATI, I DA SE FOOTER IPAK, SPAN-UJE, PREKO SVIH MOGUCIH KOLONA

```
.footer {  
    grid-column: 1 / -1;  
}
```



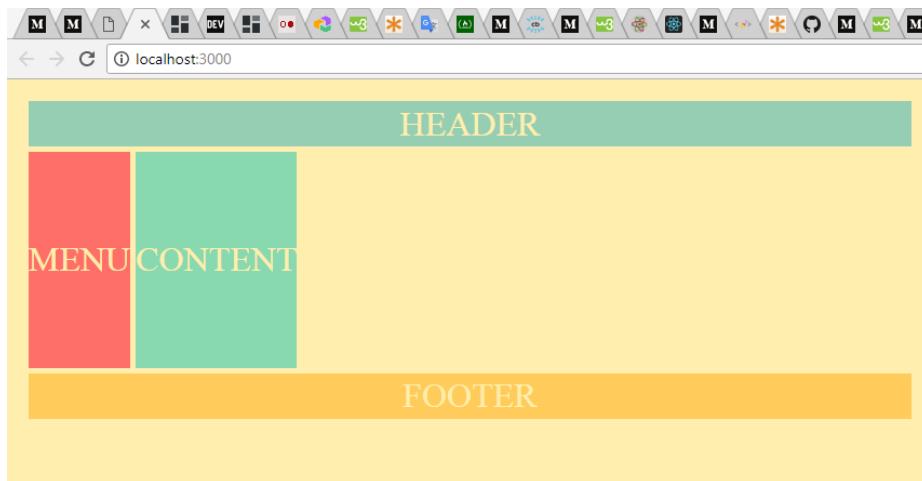
JA SAM U OVOM SLUCAJU, MOGAO SMANJITI BROJ KOLONA NA DVE; KAKO BI SE MENU I CONTENT, PROSTIRALI CELOM SIRINOM GRID-A (JER SVAKI OD NJIH, ZA RAZLIKU OD HEADER-A I FOOTER-A, JESU GRID ITEM-I KOJI SE PROSTIRU OD JEDNOG SUSEDNOG, DO DRUGOG SUSEDNOG COLUMN LINE-A; STO SE MOZE VIDETI I NA PRVOJ SLICI TRENTUTNE STRANICE OVOG WORD DOKUMENTA)

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 4fr;  
  grid-template-rows: 42px 200px 42px;  
  grid-gap: 4.8px;  
}
```



ALI, IPAK CU OSTAVITI, VECI BROJ KOLONA, JER TIME SE POSTIZE VEGA FLEKSIBILNOST, ODNOSNO VISE MOGUCNOSTI, PRILIKOM, POTENCIJALNOG SLEDECEG SHUFFLING-A GRID ITEM-A

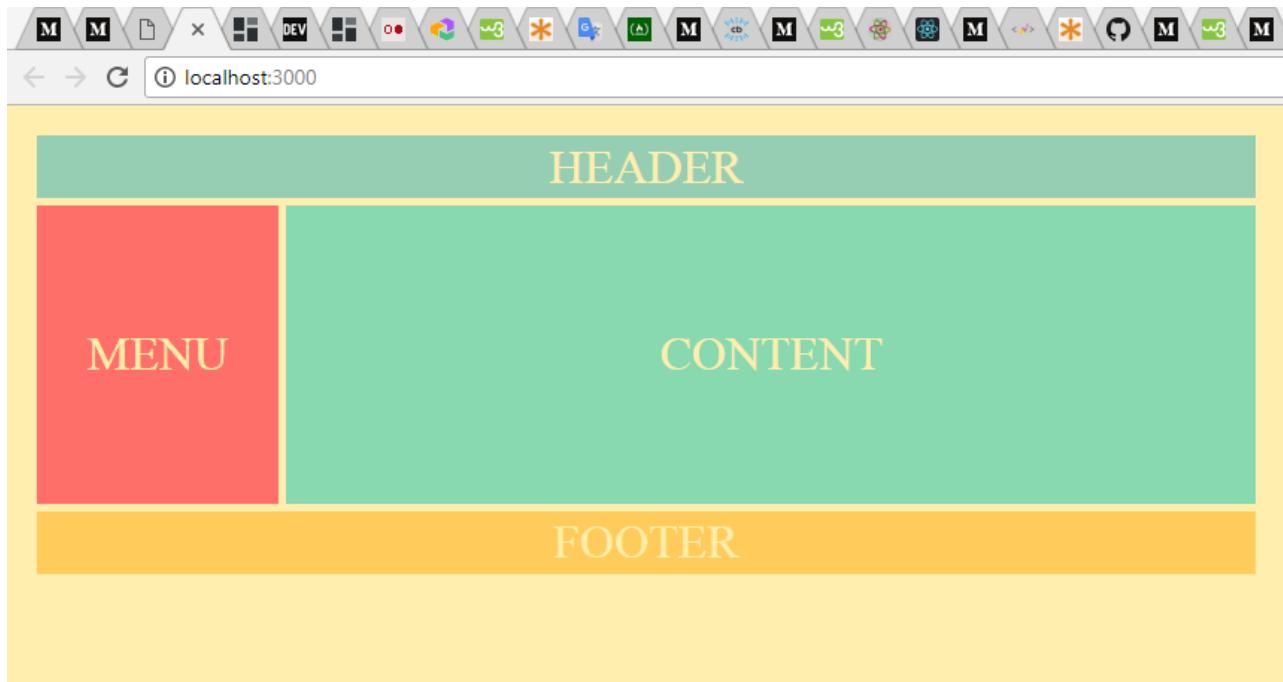
```
.container {  
  display: grid;  
  grid-template-columns: repeat(18, 1fr);  
  grid-template-rows: 42px 200px 42px;  
  grid-gap: 4.8px;  
}
```



A UCINICU SADA DA SE CONTENT, SPAN-UJE, SVE DO KRAJA DESNE IVICE GRIDA (POSLEDJNEG COLUMN LINE-A)

```
.content {  
    grid-column: 2 / -1;  
}
```

(****OPET CU INTERPRETIRATI DA SAM OVIM DEFINISAO DA SE GRID ITEM PROSTIRE OD DRUGOG COLUMN-LINE-A, PA DO KRAJNJE DESNOG COLUMN LINE-A)



ONO CIME CU SE SADA POZABAVITI, JESU PROPERTIJI `grid-row-start`, `ZATIM` `grid-row-end` I
NJIHOVIM SHORTHAND-OM `grid-row`

NAIME, POTPUNO JE SUGESTIVNO I JASNO STA SE POSTIZE, POMENUTIM PROPERTIJIMA, SAMO
TERMINOLOSKIE ODREDNICE KOJE TREBA PROMENITI JESU:

RED, ROW LINE (LINIJA REDA)

BOLJE DA POMENUTI PROPERTI, ODNOSNO PROPERTIJE, POKAZEM MOJIM PRIMEROM

NAIME, JA ZELIM DA DEFINISEM, DA SE MENU, IPAK PROSTIRE, OD PRVOG ROW LINE-A (DAKLE, U
TOM SLUCAJU, ZA NJEGA, VREDNOST PROPERTIJA `grid-row-start` TREBA DA BUDE 1, A VREDNOST
`grid-row-end` PROPERTIJA TREBA DA BUDE 2)

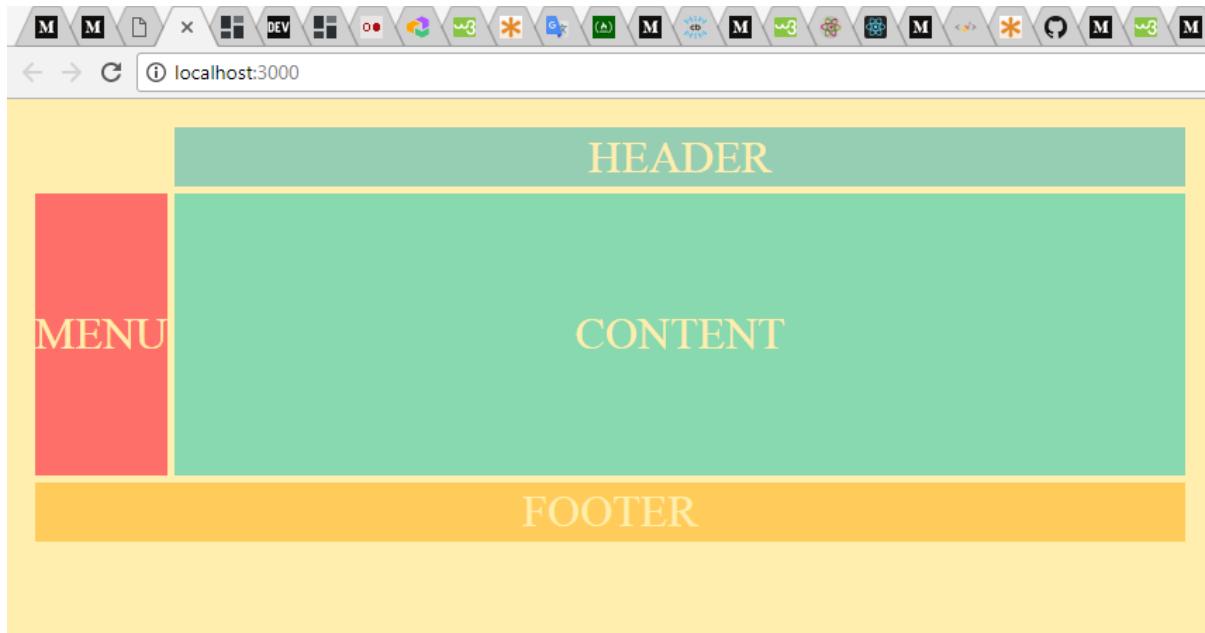
MEDJUTIM, PRE TOGA MORAM REDEFINISATI U POGLEDU KOLONA, ODAKLE POCINJE HEADER
GRID ITEM (NAIME, ON BI IPAK TREBALO DA POCINJE OD DRUGOG COLUMN LINE-A)

```
.header {  
    grid-column: 2 / -1;  
}
```

→→→→

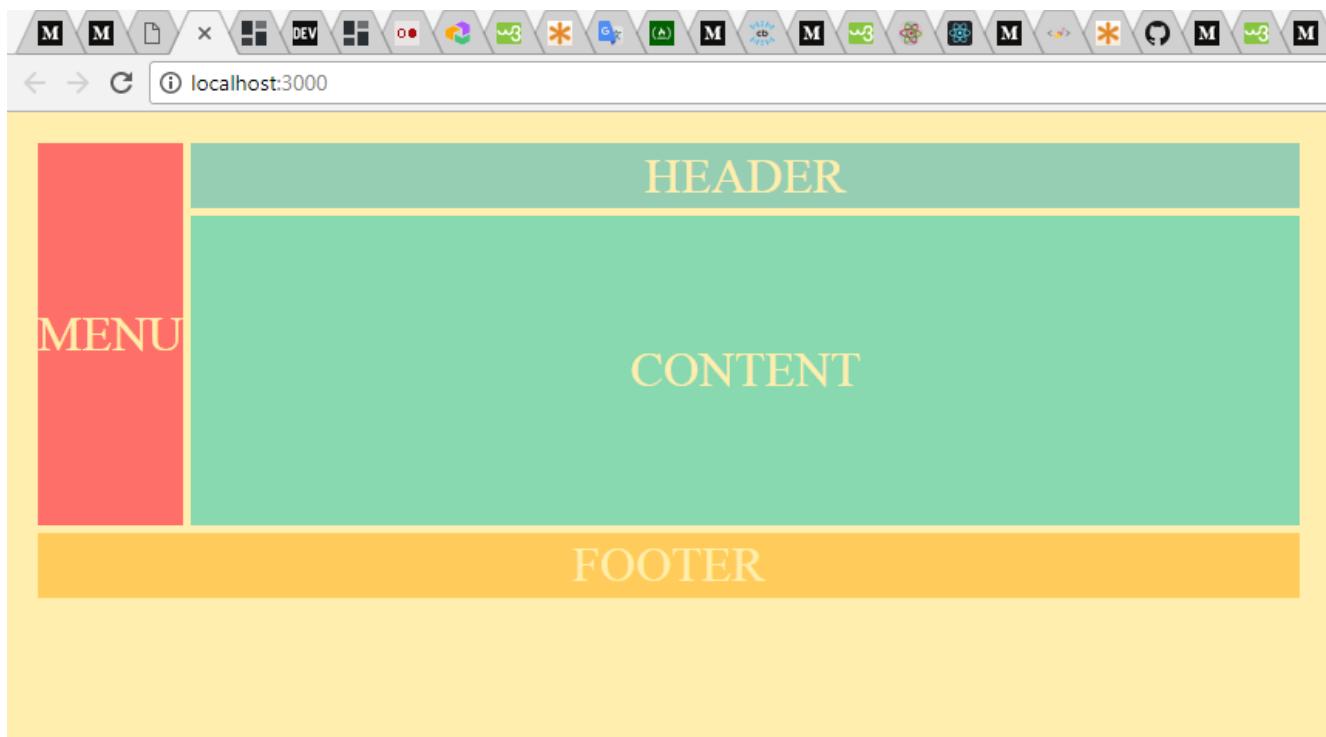
```
.header {  
    grid-column: 2 / -1;  
}
```

→→→→



SADA CU DEFINISATI POMENUTE VREDNOSTI grid-row-star I grid-row-end PROPERTIJA, ZA MENU GRID ITEM

```
.menu {  
    grid-row: 1 / 3;  
}
```



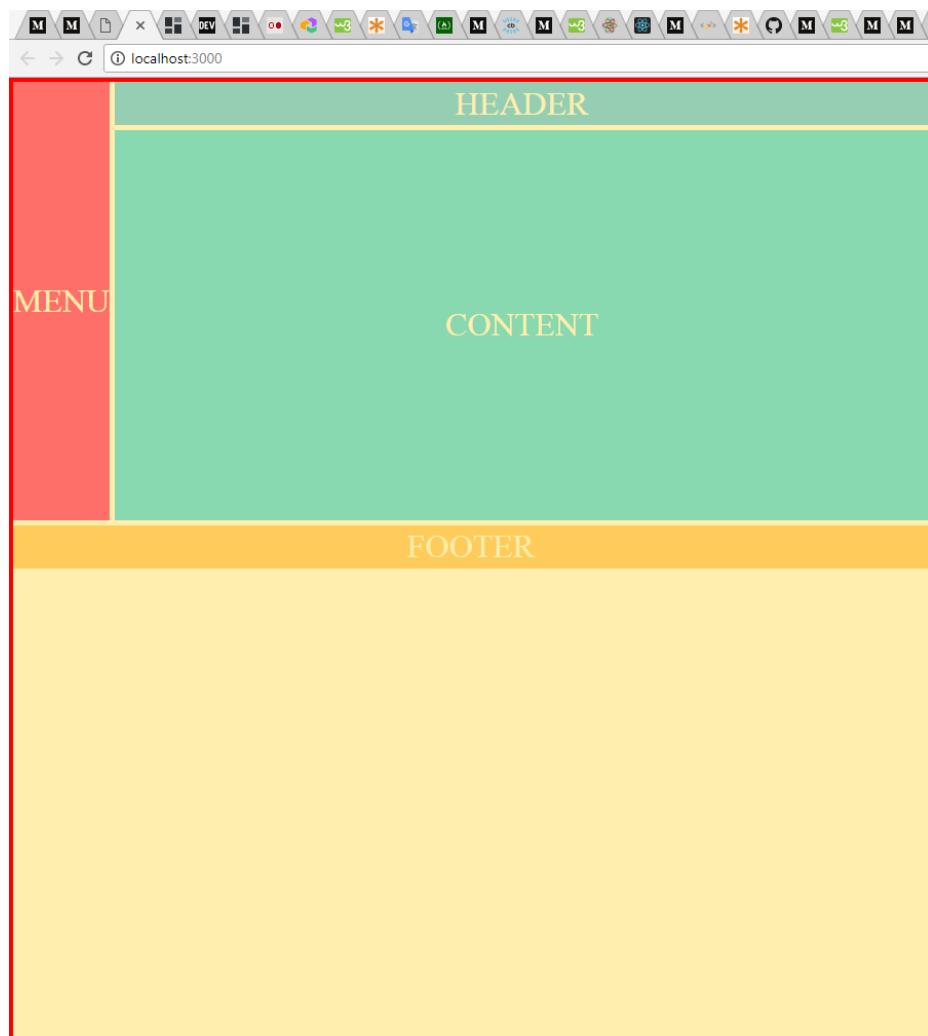
TEMPLATE PROSTORI (TEMPLATE AREAS)

PRE NEGO STO SE POZABAVIM TEMPLATE PROSTORIMA, UCINICU DA GRID IZ PREDHODNE LEKCIJE, POSTANE RESPONSIVE, I "PO VISINI"; ODNOSENJE ONO STO ZELIM DA POSTIGNEM, JESTE DA FOOTER, BEZ OBZIRA OD UREDJAJA SA KOJEG SE PREGLEDA SAJT, BUDE ZALEPLJEN ZA DNO BROWSER WINDOW-A

PRE TOGA CU DEFINISATI BORDER, ZA `html` I `body` TAGOVE, CISTO DA VIDIM, STA CE SE DESAVATI SA ELEMENTIMA, NAKON STO MODIFIKUJEM, NJIHOVE STILOVE; A DEFINISACU DA POMENUTI ELEMENTI IMAJU NULA, KAO VREDNOST ZA MARGIN-U, ZATIM DA IM VREDNOST `box-sizing` PROPERTIJA BUDE `border-box`; I ONO STO JE NAJVAZNIJE DEFINISACU DA VISINA, POMENUTIH TAGOVA BUDE 100%

`basic.js`

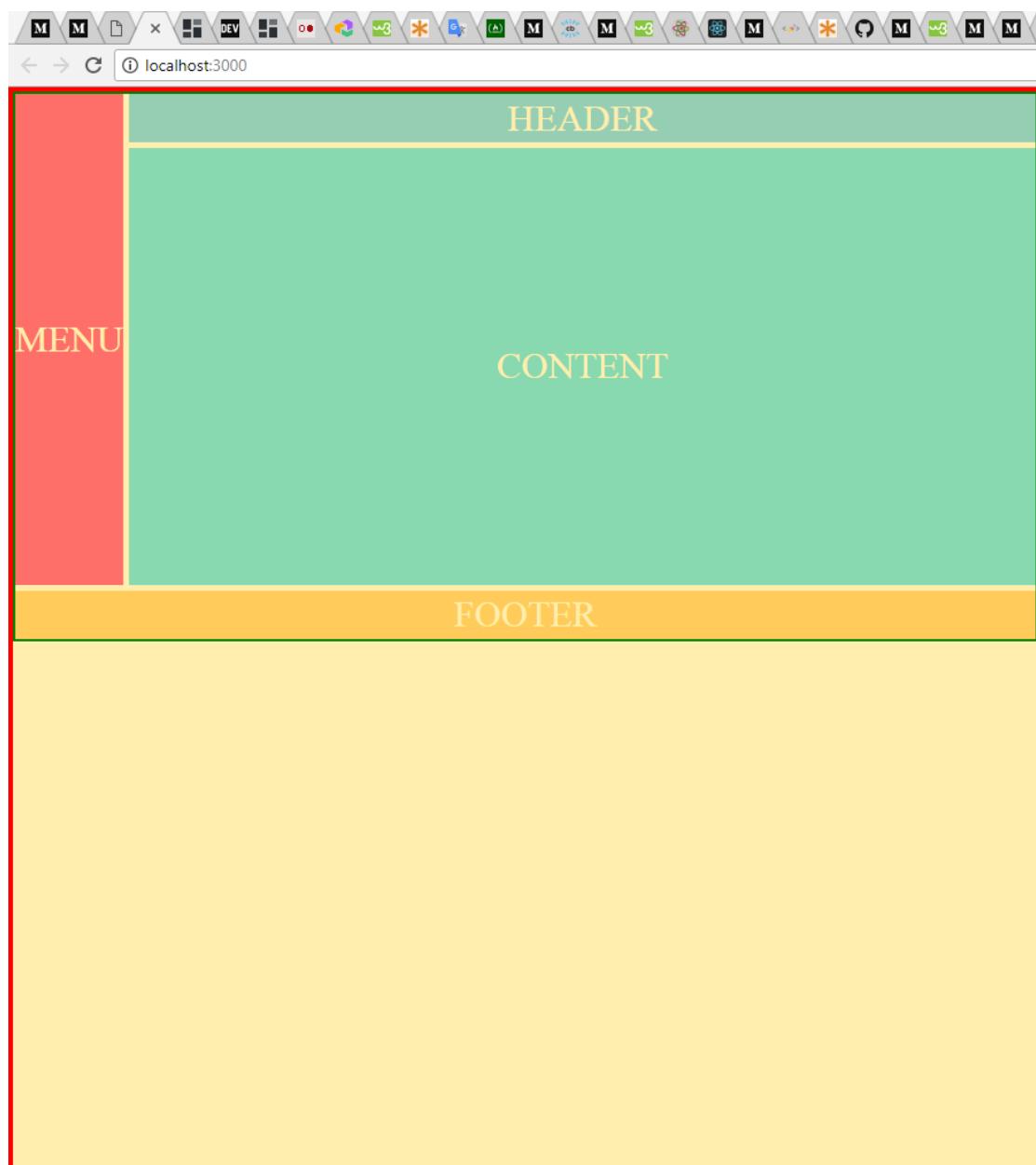
```
html, body {  
    margin: 10px;  
    background-color: #fffead;  
  
    border: red solid 2px;  
  
    margin: 0;  
    box-sizing: border-box;  
    height: 100%;  
}
```



DODACU I BORDER GRID CONTAINER-U, KAKO BIH JASNO VIDEO GDE SU, NJEGOVE GRANICE

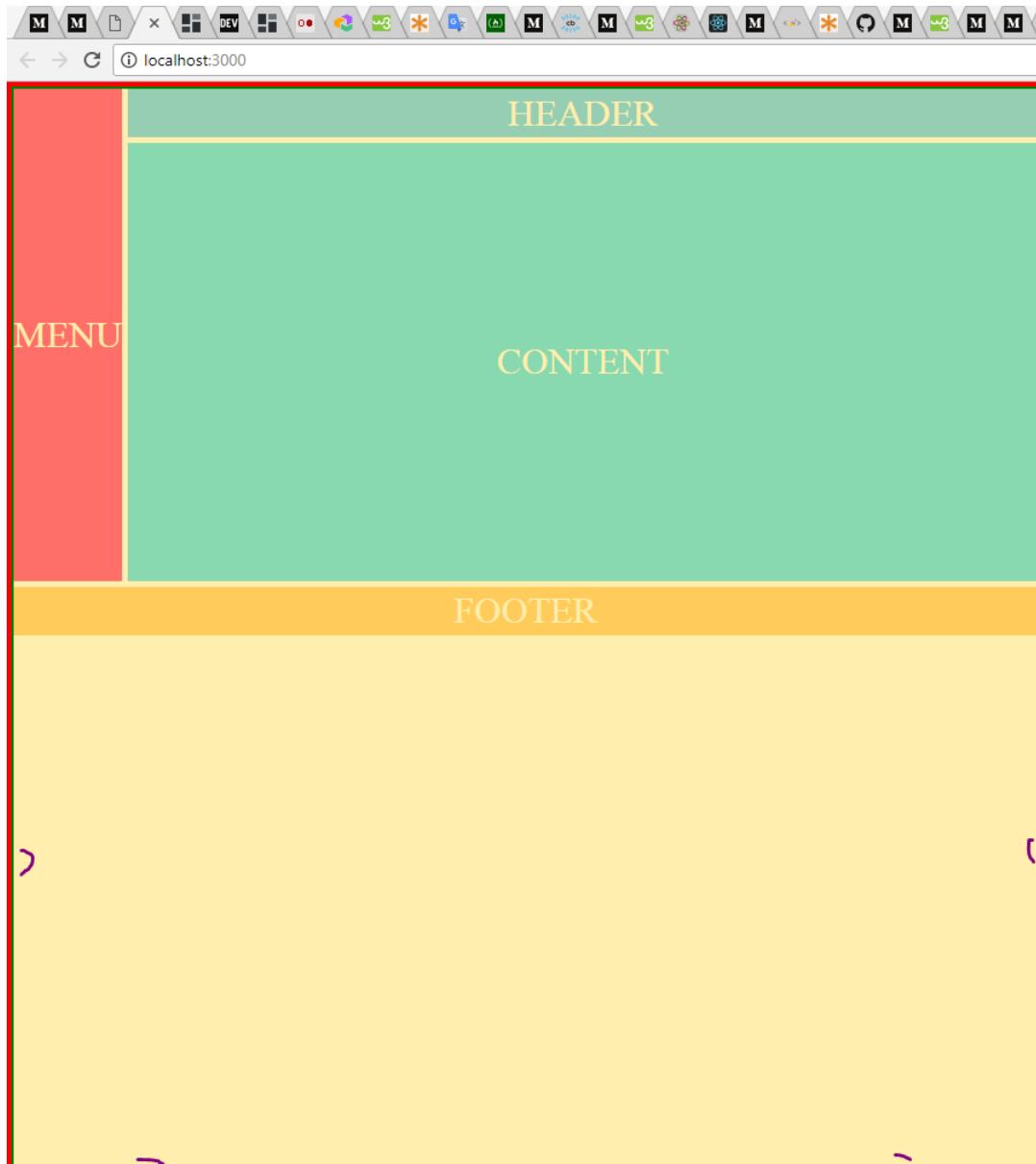
index.js

```
.container {  
  display: grid;  
  grid-template-columns: repeat(18, 1fr);  
  grid-template-rows: 42px 380px 42px;  
  grid-gap: 4.8px;  
  
  border: green solid 2px;  
}
```



ZATIM CU UCINITI DA I GRID CONTAINER, IMA VISINU 100%

```
.container {  
  display: grid;  
  grid-template-columns: repeat(18, 1fr);  
  grid-template-rows: 42px 380px 42px;  
  grid-gap: 4.8px;  
  
  border: green solid 2px;  
  
  height: 100%;  
}
```



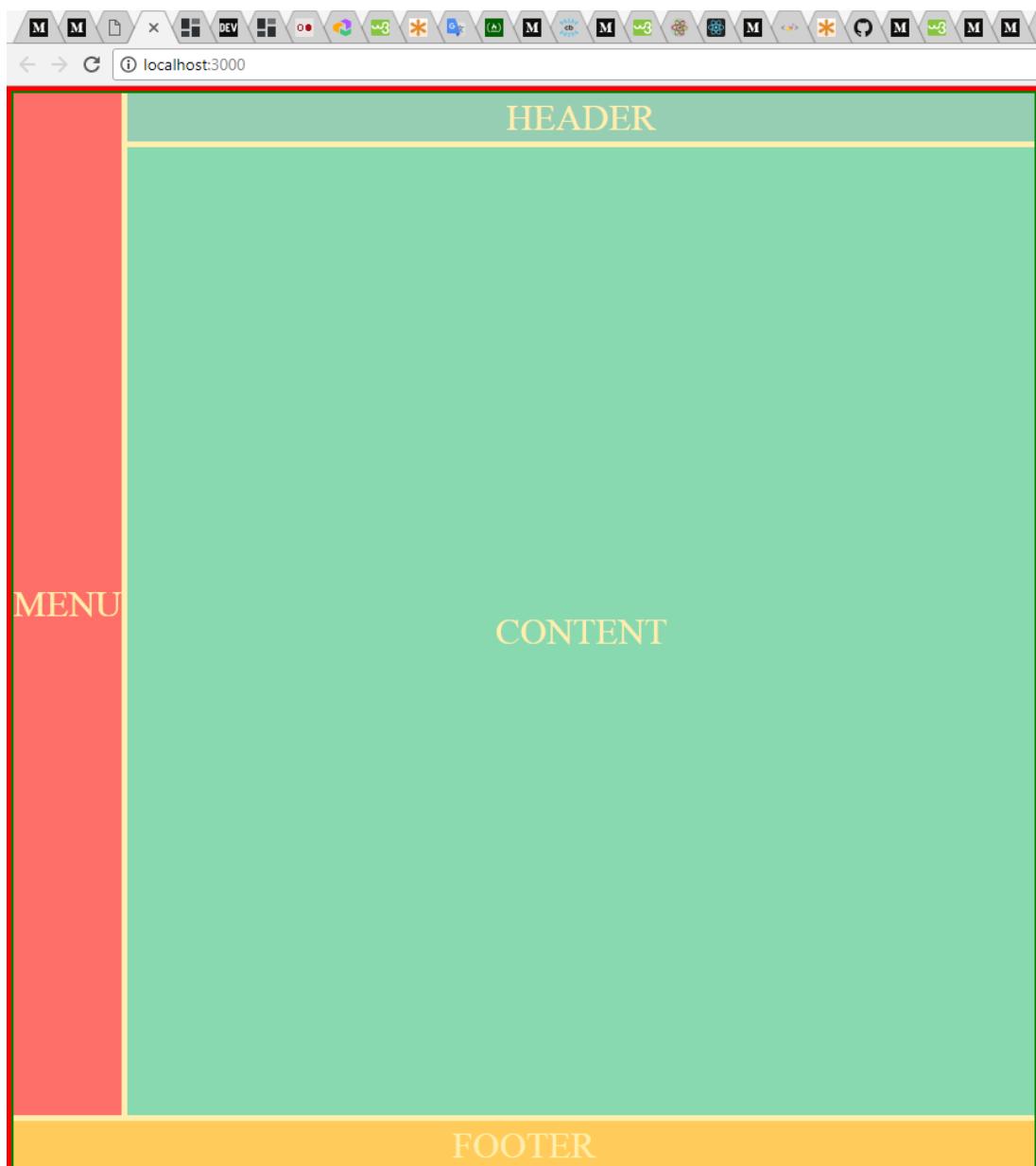
JASNO MI JE DA SE GRID KONTAINER, PROTEZE CELOM MOGUCOM VISINOM, ODNOSNO, ZAUZIMA 100% VISINE PROZORA, MA KOLIKA ONA BILA; ALI IZ SLEDECEG RALOGA:

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
    grid-template-rows: 42px 380px 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 2px;  
  
    height: 100%;  
}
```

ODNOSNO, JER SU VISINE REDOVA, UPRAVO SETTED UP NA ODREĐJENE VREDNOSTI U PIKSELIMA (KONKRETNE VREDNOSTI), DOSTUPNI REDOVI GRID-A, NECE SE PROSTIRATI, CELOM MOGUCOM, VISINOM; TO MOGU ISPRAVITI, TAKO STO CU DEFINISATI DA NEKI OD REDOVA IMAJU VISINU auto, CIME CE ONDA, TI REDOVI ZAUZETI, ONOLIKU VISINU, KOLIKA IM JE DOSTUPNA

POSTO ZELIM I DALJE DA PRVI I POSLEDJI RED, MOG GRID-A, IMAJU VISINE OD PO 42px (JER SE RADI O HEADER-U I FOOTER-U), JA CU SAMO DEFINISATI DA SREDNJI RED, MOG GRID-A IMA VISINU auto, JER SE U TOM REDU PROSTIRE CONTENT ELEMENT, ZA KOJI MI I ODGOVARA DA OBUVATI, STO VISE VISINE

```
.container {  
  display: grid;  
  grid-template-columns: repeat(18, 1fr);  
  grid-template-rows: 42px auto 42px;  
  grid-gap: 4.8px;  
  
  border: green solid 2px;  
  
  height: 100%;  
}
```



KAO STO VIDIM SA SLIKE GORE, SADA CE MOJ LAYOUT BITI POTPUNO RESPONSIVE, NA BILO KOJEM UREDJAJU, I PROMENOM SIRINE, ALI I VISINE BROWSER-OVOG PROZORA, MENJACE SE SIRINA SVIH GRID ITEM-A NA STRANICI, A ZA CONTENT I DEO HEADER-A (OSIM PRVOG ROW-A) CE SE MENJATI I VISINA

ONO STO SADA ZELIM DA URADIM JESTE DA UKLONIM SVE BORDERE (A MOGU I DA DEFINISEM DA NJIHOVA DEBLJINA BUDE NULA PIKSELA, STO JE BOLJI NACIN)

ZATIM ZELIM DA DODAM PADDING, ODNOSNO, NE ZELIM DA IMAM TRENUTNU SITUACIJU, U KOJOJ SU SVE IVICE GRID CONTAINER-A ZALEPLJENE ZA IVICE BROWSER-OVOG PROZORA

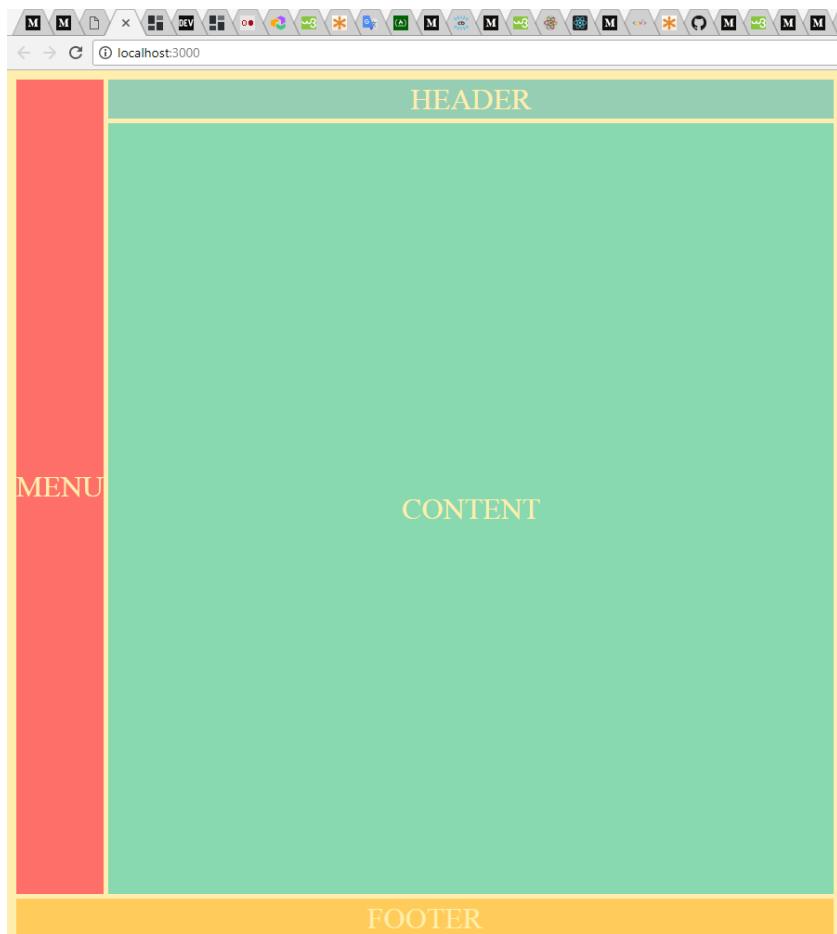
U TOM SLUCAJU MORA MOBRATITI PAZNJU NA box-sizing

NEMOJ NIKAD DA PREDPOSTAVIS DA u SLUCAJU DEFINISANJA box-sizingA JEDNOG ELEMENTA, NA border-box VREDNOST, I NJEGOV CHILD ELEMENT (ILI ELEMENTI) DOBIJAJU ISTO ISTU VREDNOST box-sizing- A

NAIME, SVAKI HTML ELEMENT IMA content-box VREDNOST, box-sizing PROPERTIJA, SVE DOK JA TO NE PROMENIM

POSTO html I body TAGOVI, VEC IMAJU DEFINISANU VREDNOST border-box ZA box-sizing PROPERTI, JA SAM SAMO MOGAO DEFINISATI padding NA TIM ELEMENTIMA; ALI JA TO NECU URADITI, JER JA LICNO ZELIM DA MOJ GRID CONTAINER IMA DEFINISAN KONKRETAN ZELJENI PADDING, ZATO CU NAPISATI SLEDECE

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
  
    padding: 10px;  
  
    box-sizing: inherit; /*NASLEDICE OD body TAGA  
                           border-box VREDNOST*/  
}
```



SADA GRID CONTAINER IMA I PADDING OD 10px

POSTO SAM UCINIO, MOJ LAYOUT, POTPUNO RESPONSIVE; POZABAVICU SE NECIM STO SE ZOVE:

TEMPLATE AREAS

UZ POMOC POMENUTOGA, OMOGUCENO VRSENJE PROMENA I EKSPERIMENTISANJE SA LAYOUTIMA, NA MNOGO LAKSI NACIN, NEGO SA TEHNIKAMA, KAKAV SU, ONE PREDSTAVLJEN-E, U PREDHODNIM LEKCIJAMA

OVA TEHNIKA JE IDEALNA, ZA BRZO KREIRANJE OBRAZACA, ODNOSENJE TEMPALTE-A LAYOUT-A KAKO BI POKAZAO, TU NOVU TEHNIKU, KORISTICU ISTI PRIMER, U KOJEM SAM KODIR-A I DO SADA U PROSLOM PRIMERU, JA SAM, ZA GRID ITEM-E KORISTIO PROPERTIJE, KAO STO SU:

grid-column-start grid-column-end grid-row-start grid-row-end

ILI SHORTHAND-E:

grid-column grid-row

MEDJUTIM, UMESTO SVIH POMENUTIH PROPERTIJA GRID ITEM-A, JA CU ZA ISTE GRID ITEME KORISTITI SAMO JEDAN PROPERTI, KOJI SE ZOVE:

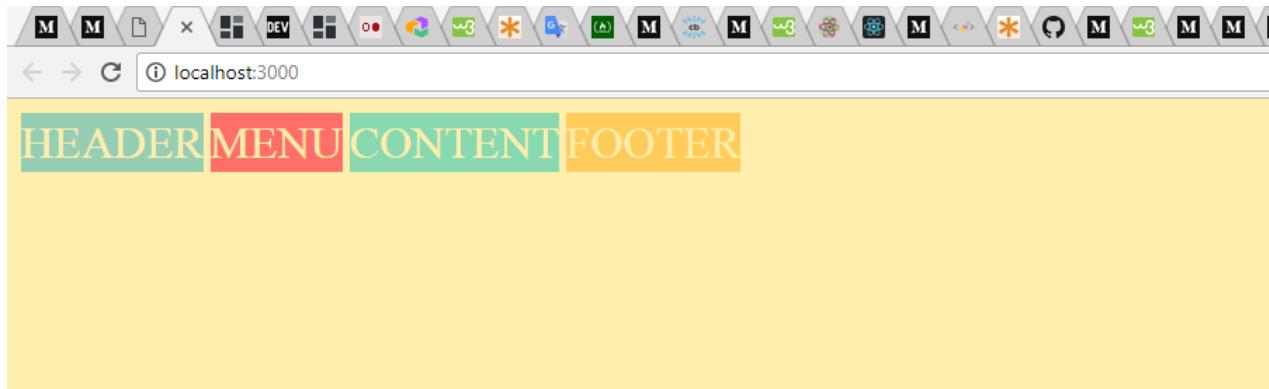
grid-area

MEDJUTIM DA BIH UOPSTE MOGAO KORISTITI, TAJ PROPERTI, MORAM ZA GRID CONTAINER DEFINISTI SLEDECI PROPERTI:

grid-template-areas

PRE NEGO STO OBJASNIM, KAKVA SE VREDNOST DODELUJE, POMENUTOM PROPERTIJU I STA ONA PREDSTAVLJA, POCECU DEFINISANJE, SA SLEDECOM "CSS SITUACIJOM"

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    /*NA GRID CONTAINER-U CU KORISTITI grid-template-areas PROPERTI*/  
}  
  
.header {  
    /*NA GRID ITEM-IMA CU KORISTITI grid-area PROPERTIJE*/  
}  
  
.footer {  
}  
  
.content {  
}  
  
.menu {  
}
```



POSTO SAM TO URADIO, MOGU RECI STA TREBA DA BUDE VREDNOST `grid-template-areas` PROPERTIJA

SINTAKSA, POMENUTE VREDNOSTI JE, POMALO CUDNA NA PRVI POGLED, A ONA USTVARI PREDSTAVLJA VIZUELNU REPREZENTACIJU GRID-A

SVAKI ROW, GRIDA JE PREDSTAVLJEN STRINGOM; SVAKA CELIJA GRID-A, JE PREDSTAVLJENA JEDNIM KARAKTEROM U TOM STRINGU, A SVAKO PRAZNO MESTO (SPACE) U TO MSTRINGU PREDSTAVLJA, ODREĐENI COLUMN LINE

SADA CU DEFINISATI VREDNOST, POMENUTOG PROPERTIJA, TAKO STO CU:

KREIRATI STRING ZA SVAKI RED GRIDA

CELIJU ZA SVAKU OD KOLONA GRIDA

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    grid-template-areas:  
  
        "h h h h h h h h h h h h h h h h"  
        "m m m m m m m m c c c c c c c c"  
        "f f f f f f f f f f f f f f f f"  
    ;  
}
```

DAKLE U PRVOM STRINGU SAM DODAO KARAKTER "h", 18 PUTA

U DRUGOM STRINGU SAM 8 PUTA DODAO KARAKTER "m", DOK SAM KARAKTER "c" DODAO 10 PUTA

U TRECEM STRINGU SAM DODAO "f" KARAKTER, 18 PUTA

DAKLE, SVAKI STRING MORA IMATI PO 18 KARAKTERA, JER TAJ BROJ MORA DA ODGOVARA, ONOLIKOM BROJU KOLONA, KOJE SAM RANIJE DEFINISAO UZ POMOC grid-template-columns PROPERTIJA; ODNOSNO MORA DA ODGOVARA MOUNT-U, KOLONA

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);      /*18*/  
  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    grid-template-areas:  
        "h h h h h h h h h h h h h h h h"  /*18*/  
        "m m m m m m m c c c c c c c"   /*18*/  
        "f f f f f f f f f f f f f f f"  /*18*/  
};  
}
```

DAKLE, POMENUTOM STRING SITAKSOM, JA USTVARI NE DEFINISEM, BROJ KOLONA, JER SAM TO VEC RANIJE URADIO, VEC SAMO VRSIM, IMENOVANJA TIH KOLONA, KAKO BI TA IMENA MOGAO KASNIJE REFERENCIRATI

A REFERENCIRACU IH KAO VREDNOSTI, grid-area PROPERTIJA, GRID ITEM-A

SA SLIKE GORE, MOGU VIDETI VIZUELNU REPREZENTACIJU ONOGA, KAKO ZELIM DA MOJ GRID IZGLEDA

MOZDA SAM DO SADA POGODIO, ALI TO CU I RECI DA SE

KARAKTER f ODNOSI NA HEADER ELEMENT

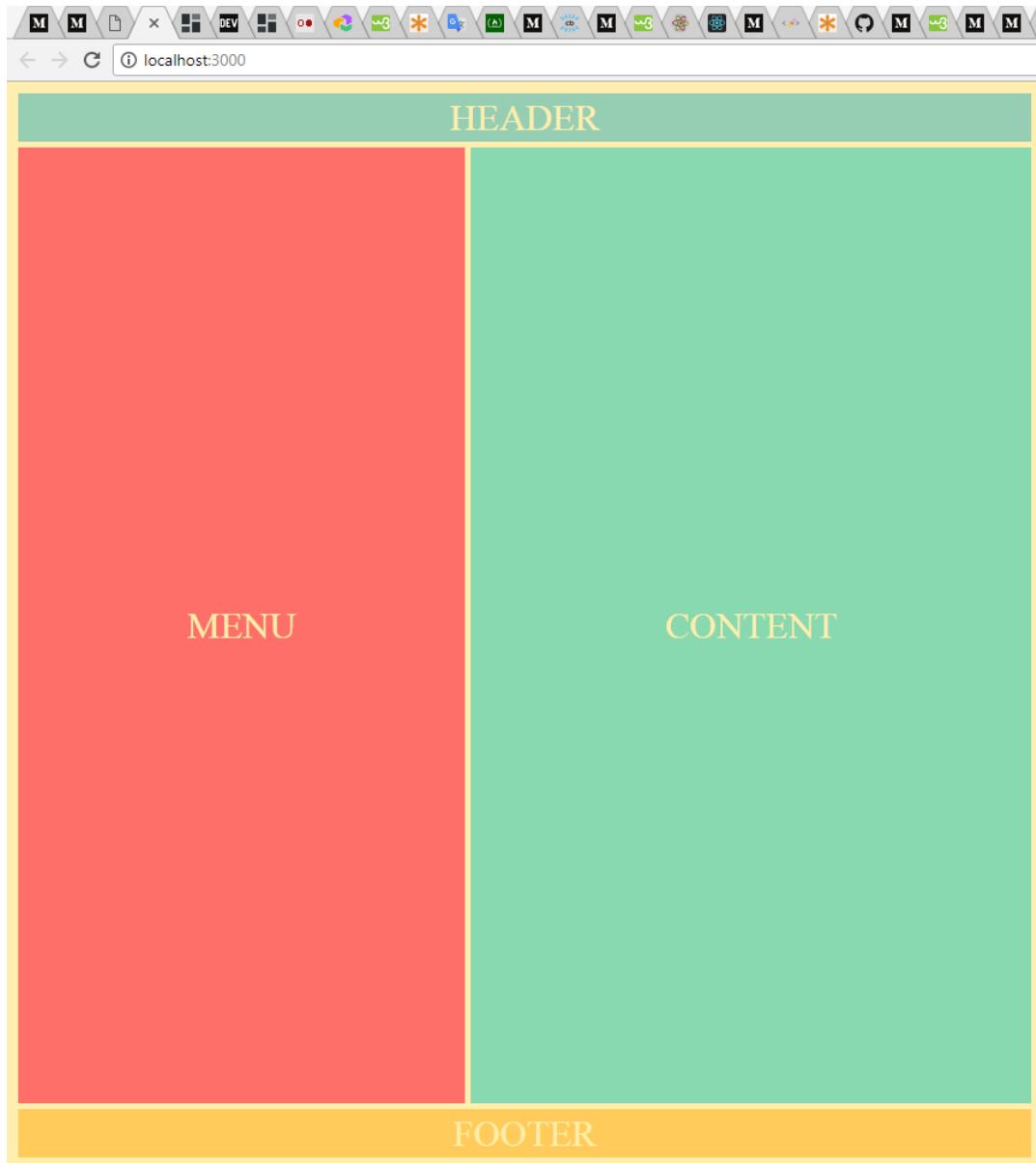
KARAKTER m ODNOSI NA MENU ELEMENT

KARAKTER c ODNOSI NA CONTENT ELEMENT

KARAKTER f ODNOSI NA FOOTER ELEMENT

SADA JE SAMO POTREBNO DA, ODGOVARAJUCI KARAKTER REFERENCIRAM, KAO VREDNOST PROPERTIJA grid-area, ODGOVARAJUCEG ELEMENTA

```
.header {  
    grid-area: h;  
}  
  
.footer {  
    grid-area: f;  
}  
  
.content {  
    grid-area: c;  
}  
  
.menu {  
    grid-area: m;  
}
```

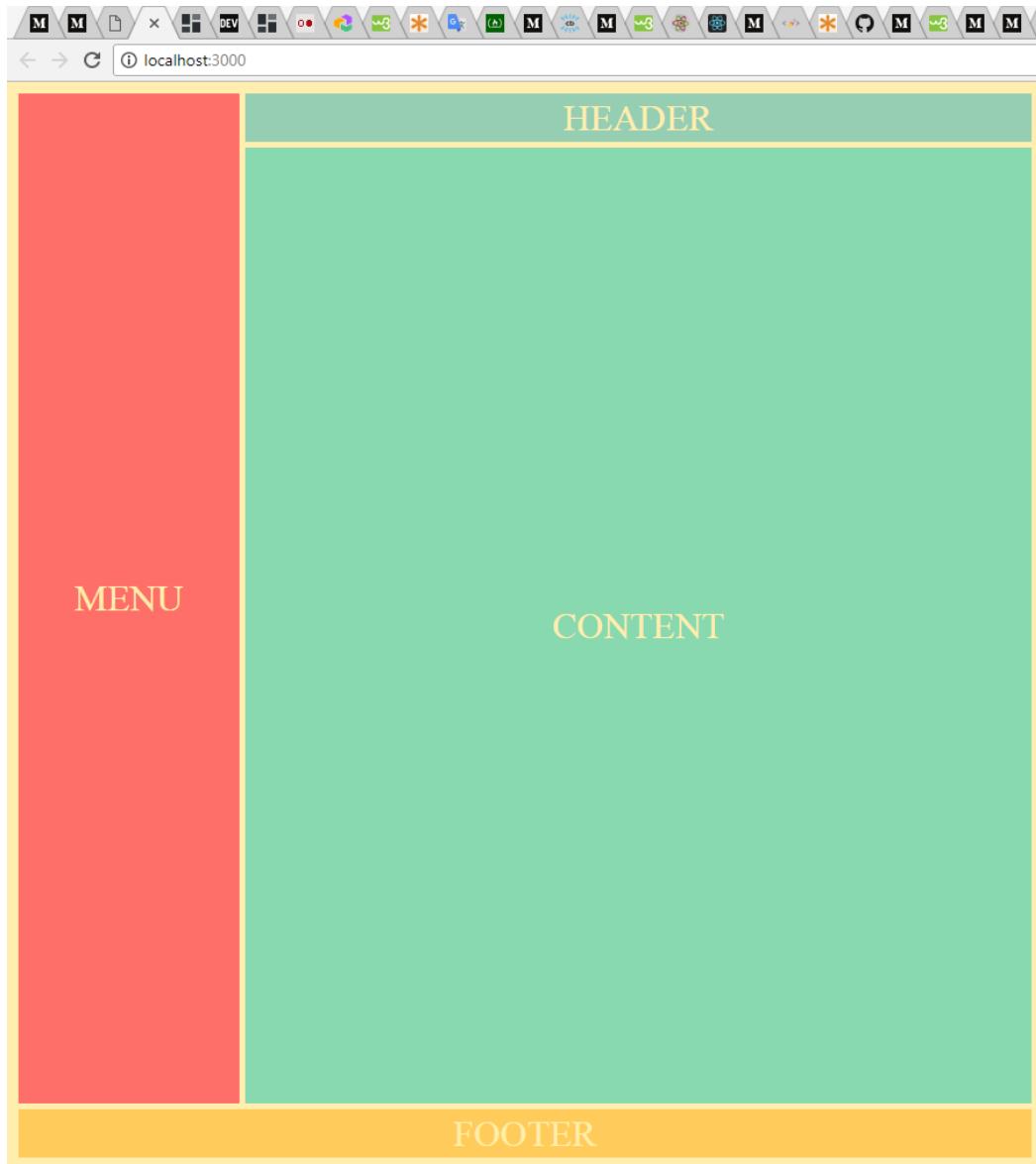


KAO STO VIDIM SA SLIKE GORE, ELEMENTI, ODNOSNO GRID ITEMI SU RASPOREDJENI BAS ONAKO KAKO SAM TO DEFINISAO U VIZUELNOJ REPREZENTACIJI, ODNOSNO VREDNOSTI grid-template-areas PROPERTIJA; A LAYOUT JE I DALJE RESPONSIVE

ZASTO JE OVAKVA TEHNIKA BRILIJANTNA?

PA ZATO STO, NA PRIMER AKO ZELIM DA SE MENU SPAN-UJE, SKROZ DO VRHA, MOGU DEFINISATI SLEDECE (TAKODJE CU UCINITI DA SE MENU SPANUJE I PREKO MANJEG BROJA KOLONA)

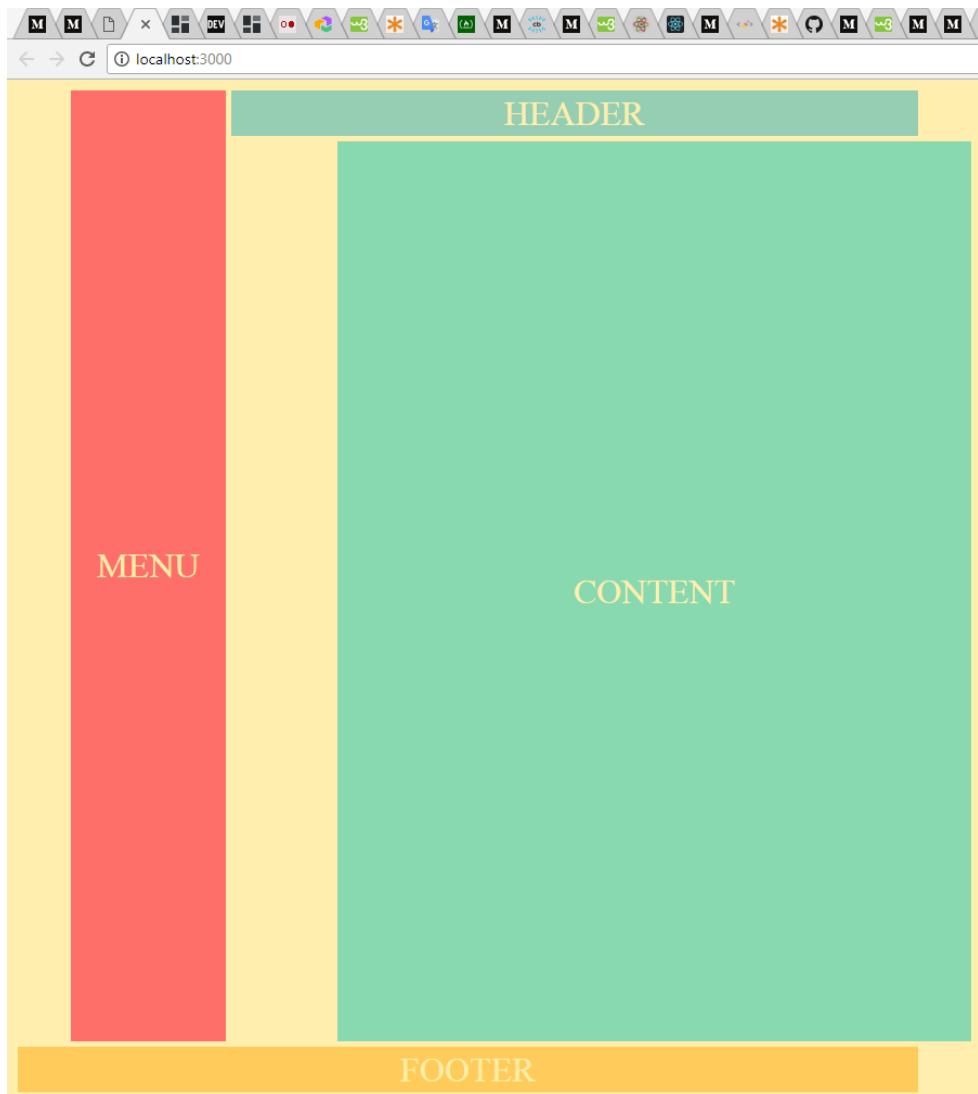
```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    grid-template-areas:  
        "m m m m h h h h h h h h h h h"  
        "m m m m c c c c c c c c c c c"  
        "f f f f f f f f f f f f f f f f"  
    ;  
}
```



KAO STO VIDIM SA SLIKE GORE, TEK TAKO SAM PROMENIO LEYAUT, BEZ TOGA DA SAM MORAO FIDDLE AROUND SA VREDNOSTIMA U ITEM KLASAMA (MISLI SE NA VREDNOSTI grid-column ili grid-row PROPERTIJA)

TAKODJE SAM MOGAO KORISTITI TACKE, U STRINGOVIMA, STO REPREZENTUJE PRAZNE CELIJE GRID-A

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    grid-template-areas:  
        ". m m m h h h h h h h h h h h ."  
        ". m m m . . c c c c c c c c c c c"  
        "f f f f f f f f f f f f f f f f f ."  
};  
}
```



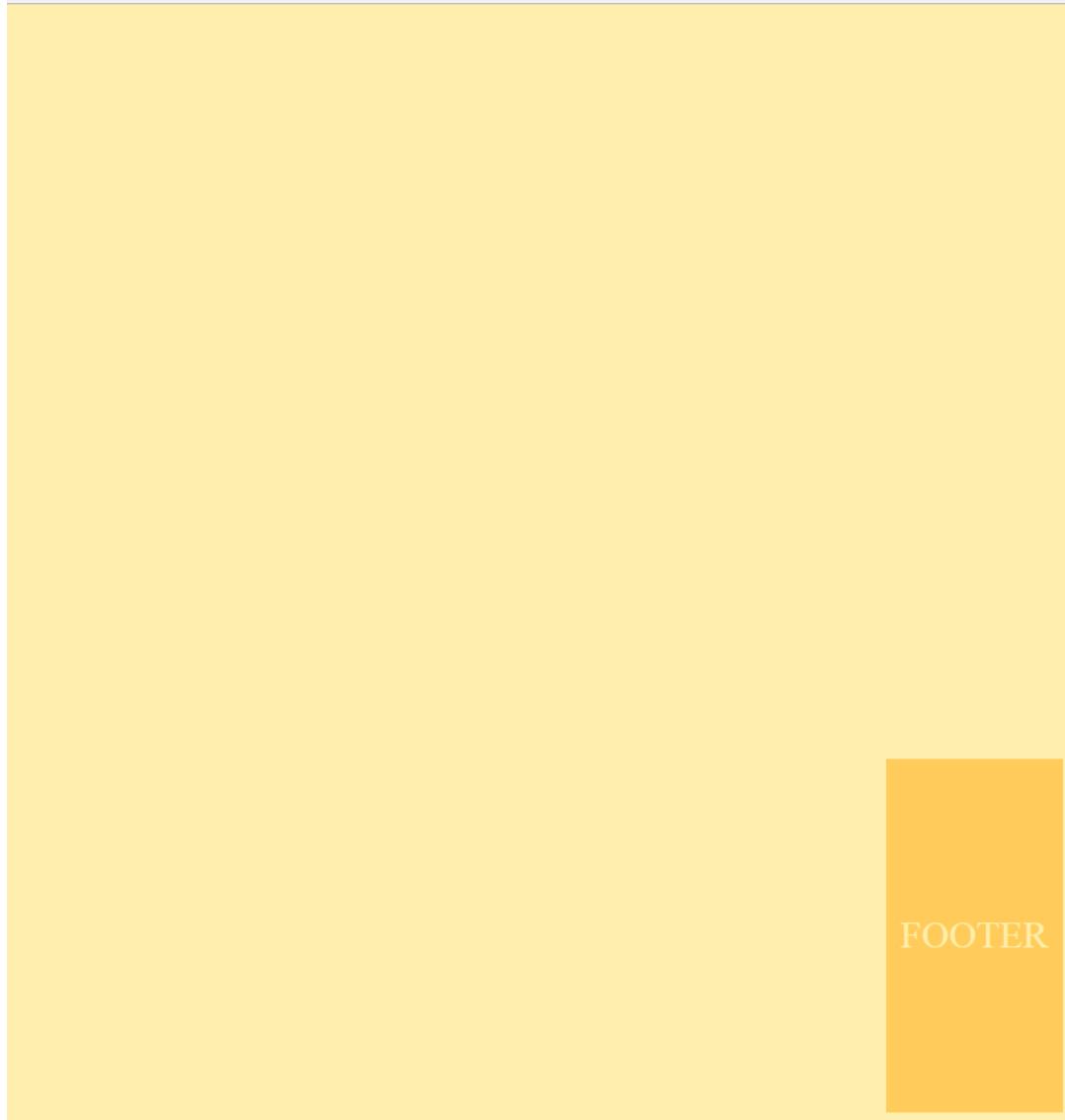
MEDJUTIM TREBA VODITI RACUNA O JEDNOJ STRANI, A TO JE DA CE AREAS BITI VALIDNA SAMO ONDA KADA SU PRAVOUGAONICNI (RECTANGLED)

DAKLE, OVO NECE FUNKCIONISATI, KAKO TREBA

```
.container {  
    display: grid;  
    grid-template-columns: repeat(18, 1fr);  
  
    grid-template-rows: 42px auto 42px;  
    grid-gap: 4.8px;  
  
    border: green solid 0px;  
  
    height: 100%;  
    padding: 10px;  
    box-sizing: inherit;  
  
    grid-template-areas:  
        "m m m h h h h h h h h h h h h h h"  
        "m m m m c c c c c c c c c c c c c c"  
        "f f f f f f f f f f f f f f f f f f f f"  
};  
}
```



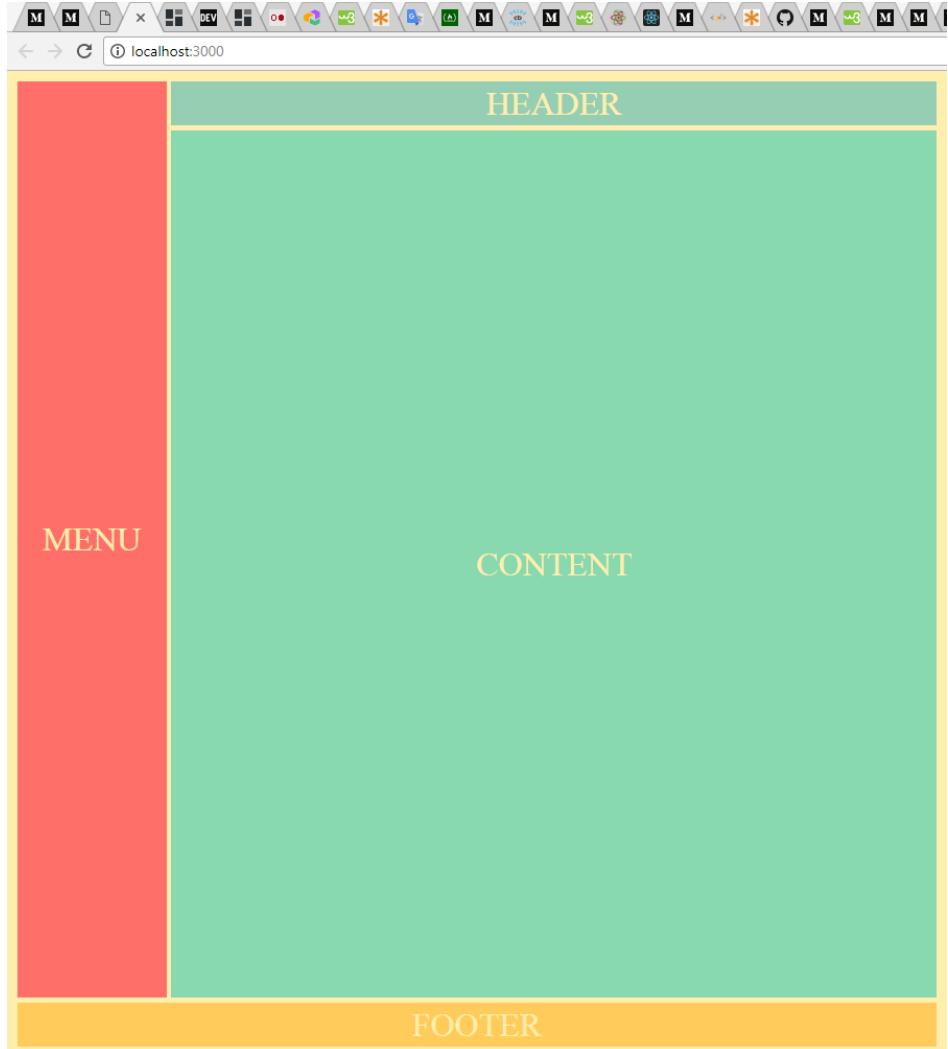
localhost:3000



ALI CE ZATO SLEDECE FUNKCIONISATI:

```
.container {  
  display: grid;  
  grid-template-columns: repeat(18, 1fr);  
  
  grid-template-rows: 42px auto 42px;  
  grid-gap: 4.8px;  
  
  border: green solid 0px;  
  
  height: 100%;  
  padding: 10px;  
  box-sizing: inherit;  
  
  grid-template-areas:  
    "m m m h h h h h h h h h h h h h"  
    "m m m c c c c c c c c c c c c c"  
    "f f f f f f f f f f f f f f f f f"  
};  
}
```

JER JE PRISUTAN RECTANGLE



KAO STO VIDIM, POMENUTO, JE SUPER-JEDNOSTAVAN (SUPER-SIMPLE) NACIN ZA EKSPERIMENTISANJE SA LAYOUT-IMA

AUTO-FIT I MINMAX (autofit AND minmax)

U SLEDECIM LEKCIJAMA, POCEV SA OVOM CU NAUCITI NEKE NAPREDNE KONCEPTE (ADVANCED CONCEPTS)

VEC SAM NAUCIO O RESPONSIVENESS-U, ALI CU SADA TO DOVESTI DO NOVOG NIVOA

SADA CU IMATI OVAKVU SITUACIJU VEZANU ZA STILOVE, U MOM PRIMERU

```
<div class="container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div>11</div>
  <div>12</div>
</div>
```

```
.container {
  display: grid;
  grid-template-columns: repeat(6, 100px);
  grid-template-rows: repeat(2, 100px);
  grid-gap: 5px;
}
```

```
html, body {
  margin: 10px;
  background-color: #ffeedad;
}

.container > div {
  font-size: 2em;
  color: #ffeedad;
  display: flex;
  justify-content: center;
  align-items: center;
}

.container > div:nth-child(1n) {
  background-color: #96ceb4;
  vertical-align: text-bottom;
}

.container > div:nth-child(3n) {
  background-color: #88d8b0;
}

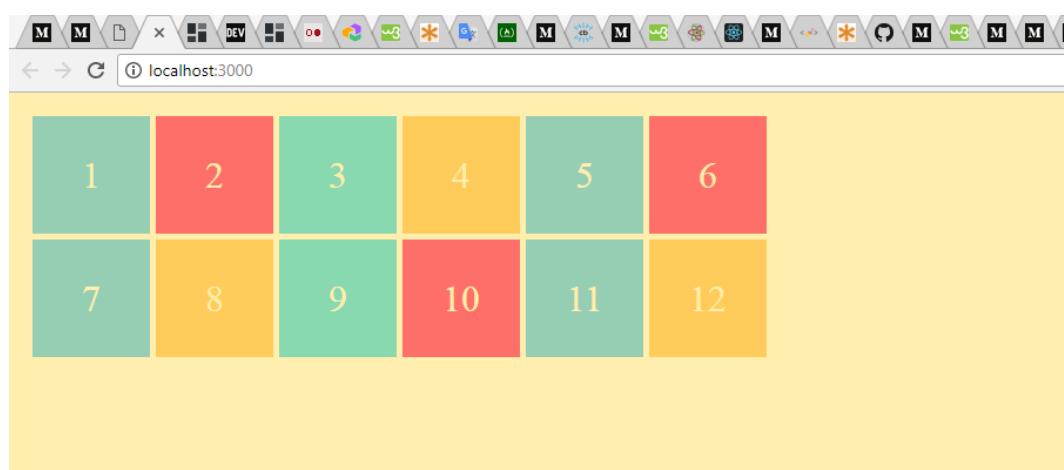
.container > div:nth-child(2n) {
  background-color: #ff6f69;
}

.container > div:nth-child(4n) {
  background-color: #ffcc5c;
}
```

index.html

index.css

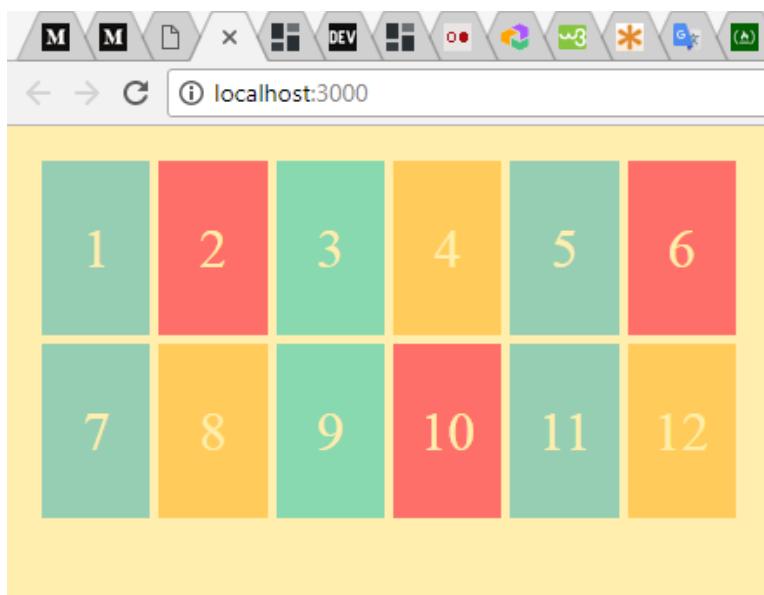
basic.css



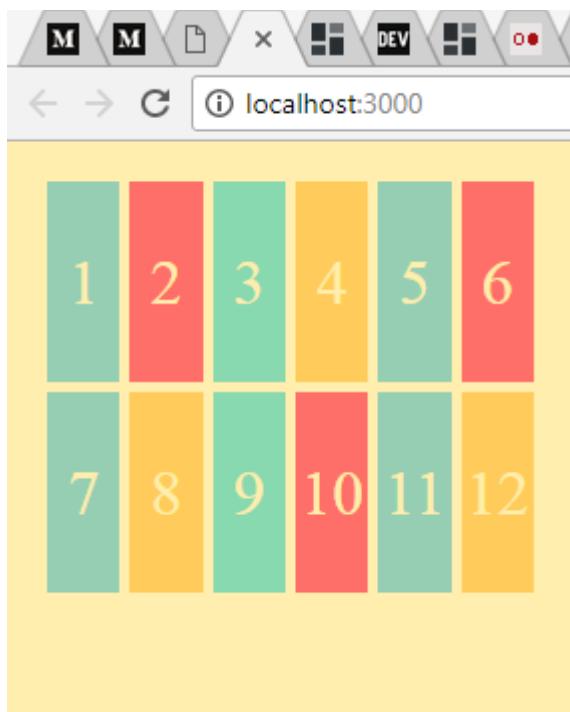
KADA BIH POVECAVAO ILI SMANJIVAO SIZE BROWSERA, VIDEO BI DA OVAJ GRID, ZAISTA NIJE RESPONSIVE

SADA CU DEFINISATI DA POMENUTI GRID BUDE RESPONSIVE U POGLEDU SIRINE

```
.container {  
    display: grid;  
    grid-template-columns: repeat(6, 1fr);  
    grid-template-rows: repeat(2, 100px);  
    grid-gap: 5px;  
}
```



DAKLE SADA SIRINA SVIH GRID ITEMA RASTE SA POVECAVANJEM SIRINE GRID CONTAINER-A (ODNOSNO WINDOW-A BROWSERA), SMANJIVAJUCI CONTAINER I SIRINA GRID ITEMA SE SMANJUJE



ALI, MOZDA JA, NE ZELIM DA BAS UVEK, BROJ KOLONA U MOM GRIDU, BUDE 6, KAO STO JE SADA DEFINISANO

SADA SU ITEMEI, VEOMA STISNUTI IZMEDJU SEBE (SQUEEZED TOGETHER)

A KAO PROSIRIM CONTAINER, ONI CE POSTATI JAKO RAZVUCENI PO SVOJOJ SIRINI

1	2	3	4	5	6
7	8	9	10	11	12

NAIME, JA NA PRIMER ZELIM DA BROJ KOLONA BUDE DVA, KADA SE GRID POSMATRA SA MOBILNOG UREDJAJA, A DA BROJ KOLONA BUDE 6, KADA SE GRID POSMATRA SA "NORMALNOG" EKRANA

DAKLE SADA CU UCINITI DA MOUNT KOLONA VARIRA U ODNOSU NA SIRINU CONTAINER-A

ALI PRE TOGA CU OPET UCINITI DA GRID BUDE UNRESPONSIVE, PO SIRINI

```
.container {  
    display: grid;  
    grid-template-columns: repeat(6, 100px);  
    grid-template-rows: repeat(2, 100px);  
    grid-gap: 5px;  
}
```

1	2	3	4	5	6
7	8	9	10	11	12

MEDJUTIM SADA CU OPET MODIFIKOVATI repeat FUNKCIJU, ODNOSNO SADA CU MODIFIKOVATI, NJEN PRVI ARGUMENT; ODNOSNO MOUNT KOLONA (KOJI JE TRENUTNO 6)

KAO MOUNT ARGUMENT, SADA KORISTICU NESTO DRUGO, A TO JE auto-fit

1	2	3	4	5
6	7	8	9	10
11	12			

TIME CE SE POSTICI DA BROJ KOLONA VARIRA, SA SIRINOM GRIDA

1	2	3	4	5	6	7	8
9	10	11	12				

DAKLE, OVIM CE SE POKUSAVAJU DA SE UKLOPI (TO FIT), STO VECI MOGUCI BROJ KOLONA U RASPOLOZIVU SIRINU CONTAINERA; DAKLE KADA SE U OVOM SLUCAJU POSTIGNE EXTRA-SPACE (SIRINE 100px) NA DESNOJ STRANI CONTAINER-A, UBACICE SE SLEDECI GRID ITEM

1	2	3	4	5	6	7	8	9	10	11	12
---	---	---	---	---	---	---	---	---	----	----	----

TAKO DA CE SE U SLUCAJU VEOMA NARROW SCREEN-A, GRID SASTOJATI OD SAMO NEKOLIKO KOLONA

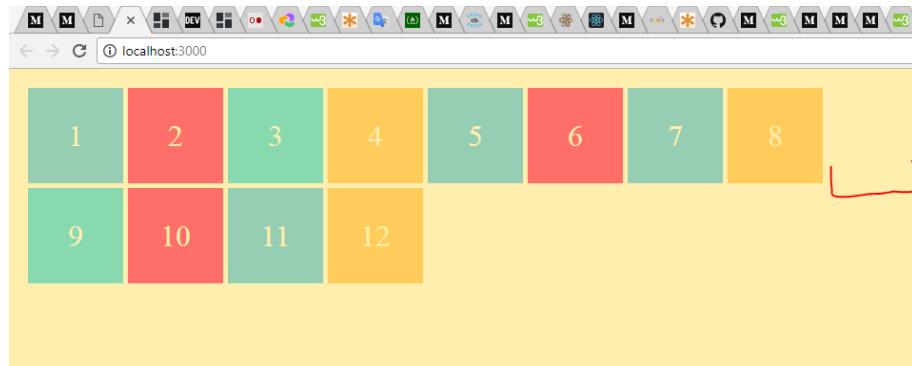
1	2
3	4
5	6
7	8
9	10
11	12

MEDJUTIM OVO MI NE DAJE RESPONSIVENESS, KAKAV SAM ZELEO, IZ SLEDECEG RAZLOGA

KOLICINA KOLONA SIRINE 100px, SE RETKO UMESTI U SVU DOSTUPNU SIRINU CONTAINER-A

NA STA MISLIM PO D TIME?

MISLIM NA SLEDECU SITUACIJU:

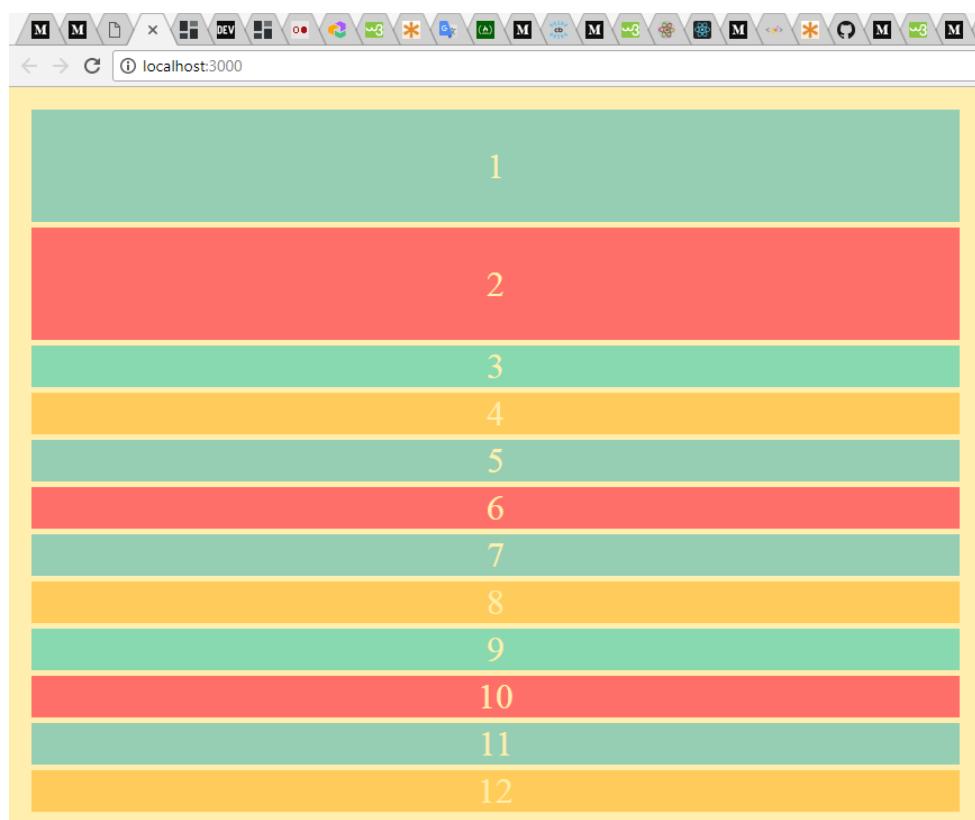


ODNOSNO SITUACIJU, KADA IMAM VEOMA DOSTA PREOSTALOG PROSTORA, NA RIGHT HAND SIDE-U; ALI PREMALO PROSTORA DA BI SE DODALA NOVA KOLONA SIROKA 100px; I TO ZAISTA NE IZGLEDA LEPO

MOZDA MISLIS DA JE RESENJE, ODNOSNO DA JE DOBRA IDEJA DODAVANJE VREDNOSTI FRACTION UNITA, KAO DRUGOG ARGUMENTA repeat FUNKCIJE

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, 1fr);  
    grid-template-rows: repeat(2, 100px);  
    grid-gap: 5px;  
}
```

MEDJUTIM TO NIJE DOBRA IDEJA U STA CU SE I UVERITI



NAIME, DESILO SE DA SADA POSTOJI SAMO JEDNA KOLONA; A TO SE DOGODILO IZ RAZLOGA STO auto-fit POKUSAVA, DA ISPUNI GRID SA STO JE VISE MOGUCE KOLONA, DOK NE BUDE DOVOLJNO PROSTORA DA SE DODA NOVA KOLONA; A KADA KORISTIM FRACTION UNIT-E, DOGADJA SE, ODNOŠNO POCINJE SE SA DODAVANJEM JEDNE KOLONE, MEDJUTIM SVA SE SIRINA ISKORISTI IMMEDIATELY, ZATO STO U TOM TRENUTKU, OBE, I SIRINA CELOG GRID CONTAINER-A I SIRINA PRVE KOLONE, SU PODESENE NA JEDNU FRACTION JEDINICU (1 FRACTION UNIT)

DA BI DOBIO, ONAKVO POANASANJE KAKVO ZELIM, KORISTICU NESTO DRUGO UMESTO FRACTION UNIT VREDNOSTI, KAO DRUGI ARGUMENT repeat FUNKCIJE, A TO JE:

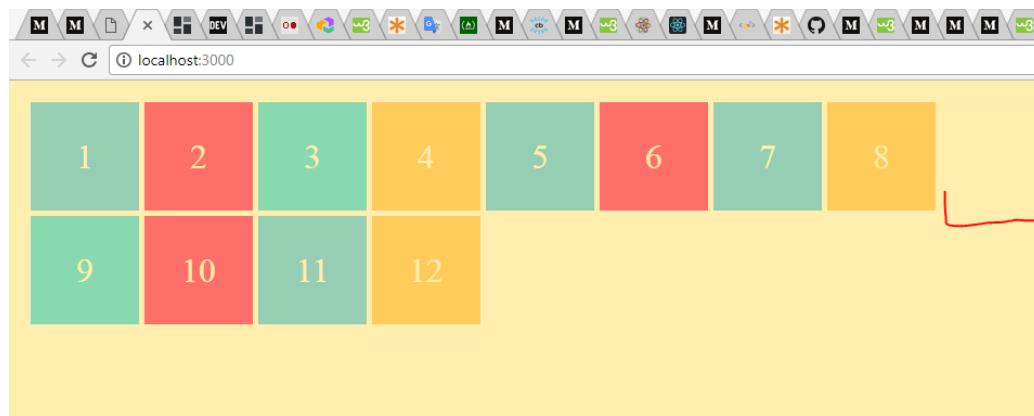
minmax

NAIME, POMENUTIM CU POSTICI DA MINIAMLNA VREDNOSTI SIRINA SVIH KOLONA BUDU 100px

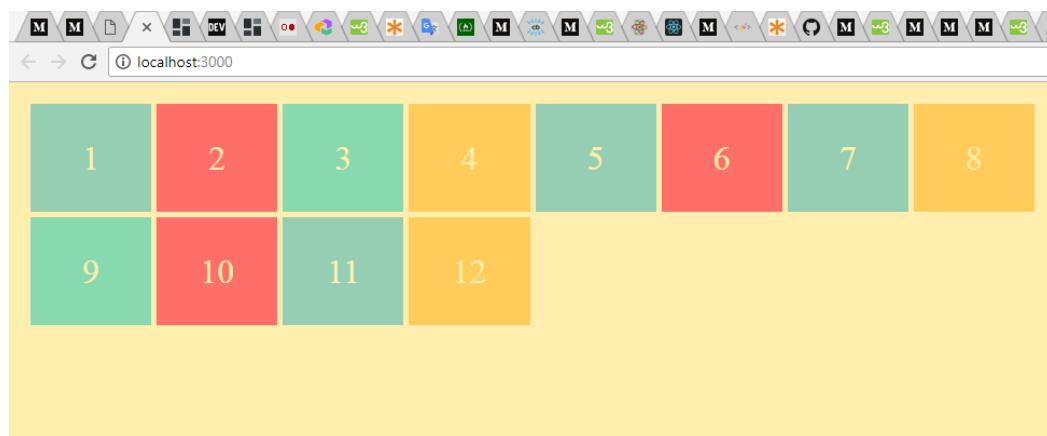
I DA MKSIMALNA VREDNOST SIRINA SVIH KOLONA BUDE 1fr

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: repeat(2, 100px);  
    grid-gap: 5px;  
}
```

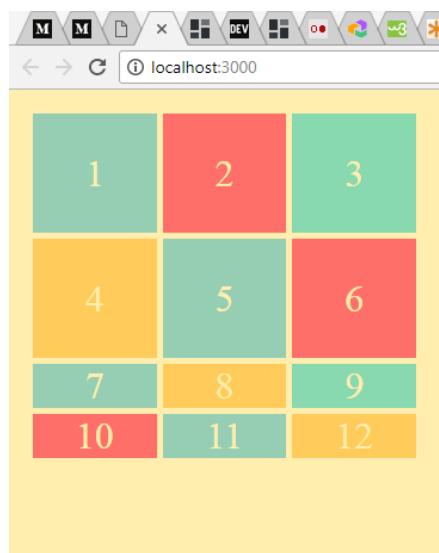
NAIME, RANIJE, KADA JE VREDNOST DRUGOG ARGUMENT repeat FUNKCIJE BILA SAMO 100px POSTOJAO JE POMENUTI PREOSTALI PROSTOR



A SADA, KADA SAM ISKORISTIO minmax, TOG PROSTORA VISE NEMA



I PRI BILO KOJOJ SIRINI CONTAINER-A



DAKLE, SADA KOLONE FIT (UKLAPAJU), U CELOKUPNI GRID, BEZ OBZIRA KOJA JE SIRINA CONTAINER-A; ODNOSNO BEZ OBZIRA, KOLIKA JE TA DOSTUPNA SIRINA SA RIGHT HAND SIDE-A (KOJA NIJE BILA POPUNJENA RANIJE)

ONO STO SE DESAVA U OVOM SLUCAJU JESTE, DA CE SVE KOLONE IMATI SIRINU BAREM 100px, ALI AKO POSTOJI VISE DOSTUPNOG PROSTORA, TAJ PROSTOR CE SE DISRIBUIRATI (DISTRIBUTE), SVIM KOLONAMA, JEDNAKO

DAKLE, SADA JE PRISUTAN DOBAR PRIKAZ I NA MOBILNOM U REDJAJU, ALI I NA WIDE SCREEN-U

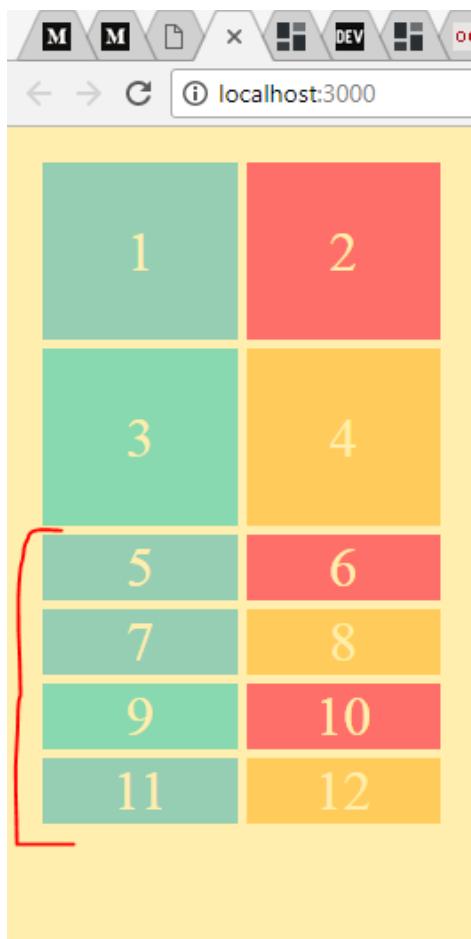
IMPLICITNI REDOVI (IMPLICIT ROWS)

U OVOJ LEKCIJI, NASTAVLJAM SA GRIDOM, KOJI SAM KREIRAO U PROSLOJ LEKCIJI

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: repeat(2, 100px);  
    grid-gap: 5px;  
}
```

NAIME, POSTOJI JOS JEDAN PROBLEM ODNOSNO ISSUE, KOJI JE POTREBNO POPRAVITI

ISSUE SE OGLEDABA U TOME DA, KADA MOJ GRID UCINIM DA BUDE NARROW, I KADA POGLEDAM
ONE REDOVE, KOJI SU ISPOD ONA DVA, KOJE SAM DEFINISAO, VIDEĆU DA SE ONI NE PONASAJU NA
NACIN, KAKO BI JA ZELEO DA SE PONASAJU



NAIME, NJIHOVA VISINA NIJE 100px, KAKO SAM DEFINISAO ZA PRVA DVA REDA, A VISOKI SU SAMO
ONOLIKO KOLIKI IH CONTENT U NJIMA, PRIMORAVA DA BUDU

NAIME TO SE DOGADJA JER, SU SVI OSTALI REDOVI, IZUZEV PRVA DVA, **USTVARI IMPLICITNO
(PRECUTNO) KREIRANI (IMPLICITLY CREATED ROWS)**

KAO STO MOGU DA VIDIM SA SLEDECE SLIKE, SMO SU DVA REDA EKSPLICITNO KREIRANA

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: 100px 100px;  
    grid-gap: 5px;  
}
```

MEDJUTIM, SA OVIM, TRENUTNIM, DEFINISANIM PONASANJEM KOLONA, MORAM PRONACI MESTO GDE CU SMESTITI, POMENUTE GRID ITEME (IZ IMPLICITNIH REDOVA), KOJE NE MOGU DA STANU (CAN'T FIT) U EKSPLICITNOM DELU GRID-A

POCEVSI OD PETOG GRID ITEMA PA NA DALJE, JA MORAM DEFINISATI AUTOMATSKO KREIRANJE REDOVA, KAKO BI NA ZELJENI NACIN MOGAO PRIKAZATI, POMENUTE GRID ITEME

POSTO NISAM REKAO, MOM GRID-U, KAKO DA ZELIM DA BUDU STILIZOVANI REDOVI, KOJI MORAJU DA SE DODAVAJU SAMOM GRID-U, SAM GRID MORA DA ODLUCI KAKO CE TI REDOVI IZGLEDATI

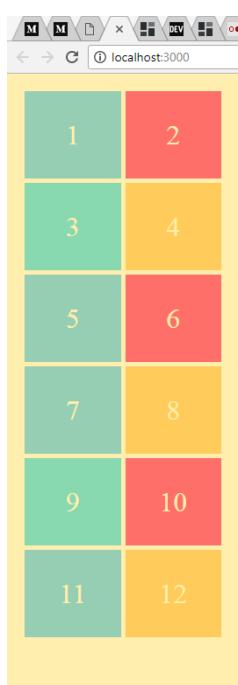
SADA CU TO POPRAVITI

NAIME JA MOGU IZABRATI (TARGET), TE REDOVE, UZ POMOC JEDNOG PROPERTIJA, KOJI SE ZOVE:

grid-auto-rows

KOJEM ONDA MOGU DODELITI VREDNOST, KOJA U OVOM SLUCAJU TREBA DA BUDE 100px

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: 100px 100px;  
    grid-gap: 5px;  
    grid-auto-rows: 100px;  
}
```



KAO STO VIDIM IMPLICITNI REDOVI IZGLEDAJU ONAKO, KAKO SAM ZELEO

A POSTO SAM DEFINISAO grid-auto-rows, MOGU POTPUNO UKLONITI PREDHODNO DEFINISANJE REDOVA

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: 100px 100px;  
    grid-gap: 5px;  
    grid-auto-rows: 100px;  
}
```



```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-gap: 5px;  
    grid-auto-rows: 100px;  
}
```



POSTO MI ODGOVARA, DA U OVOM SLUCAJU SVI REDOVI BUDU IMPLICITNO KREIRANI, I DA IM VISINA BUDE 100px

U SLEDECOJ LEKCIJI, POVEZACU (TIE) ZAJEDNO, SVE KONCEPTE, KOJE SAM NAUCIO DO SADA, U POGLEDU GRID-A, I KREIRACU SUPER COOL IMAGE GRID

AWESOME IMAGE GRID

(****POMINJANJE I
require FUNKCIJE****)

U OVOJ LEKCIJI CU ISKORISTI STO SAM DO SADA NAUCIO, U POGLEDU GRID-A, KAKO BI KREIRAO JEDAN, VERY COOL IMAGE GRID

MEDJUTIM, JA CU REDNER-OVATI SAV HTML, UZ POMOC REACT-A (ODNOSNO JSX-A), JER IZ RAZLOGA STO SAM KREIRAO REACT APLIKACIJU, JA NEMAM MOGUCNOST DIREKTNOG UCITAVANJA SLIKA U img TAGOVE, HTML-A

SLEDECİ DEO BAS NEMA VEZE SA TEMOM OVE LEKCIJE, ILI GRID-OM UOPSTE, ALI JE DOBRO DA ZNAM, KAKO SE SVE STO JE POTREBNO, U SLUCAJU REACT APLIKACIJE, MORA UVODITI, KAO MODUL, BILI TO STILOVI, ILI SLIKE ILI DRUGI JAVASCRIPT FAJLOVI (ILI NJIHOVI DELOVI)

DAKLE, JA MORAM IMPORTOVATI SLIKU PO SLIKU U MOJ JAVASCRIPT FAJL, TAKO DA CU VRSITI IMPORT SLIKE, KAO MODULA

SVA IMPORTOVANJA SAM, RANIJE OBAVLJAO UZ POMOC import KEYWORDA; ALI SADA CU PROBATI DA ISKORISTIM, NESTO STO SE ZOVE require FUNKCIJA

```
function ucitajGaleriju(url, brSlika){  
  
    const reactElementiNiz = [];  
    const imenaKlasa = [  
  
        null,  
        "vertical",  
        "horizontal",  
        null,  
        null,  
        "big",  
        null,  
        "vertical",  
        null,  
        "horizontal",  
        null,  
        "big",  
        null,  
        "horizontal",  
        null,  
        "big",  
        null,  
        "vertical"  
  
    ];  
  
    for(let i = 0; i < brSlika; i++){  
  
        reactElementiNiz.push(  
            <div key={"slika" + i} className={imenaKlasa[i]}>  
                <img alt="slika" src={require(` ${url}${i + 1}.jpg`)} />  
            </div>  
        );  
    }  
  
    return reactElementiNiz;  
}
```

index.js

NECU DODATNO OBJASNJAVAĆI, UPOTREBU require FUNKCIJE (SA SLIKE GORE), SAMO CU RECI DA SE, NJENOM POZIVANJU DODAJE URL STRING KAO ARGUMENT; I RECI CU SAMO DA POMENUTA IMA VEZE SA UCITAVANJEM MODULA; JANO JE DA CE I NJENA POV RATNA VREDNOST BITI POSEBAN URL, KOJI KAO STO MOGU VIDETI SA SLIKE GORE, UPRAVO "UCITAVAM U JSX", ODNOSNO DODELJUJEM src PROP-U, IMAGE ELEMENTA

SA SLIKE GORE, TAKODJE JE JASNO, DA SAM DEFINISAO DODAVANJE className ATRIBUTA, POMENUTOM JSX-U

DAKLE, POV RATNA VREDNOST FUNKCIJE SA SLIKE GORE JESTE NIZ REACT ELEMENATA, "CIJA STRUKTURA" SE SASTOJI OD div TAGOVA, A U SVAKI div, TREBA DA JE NESTED I JEDAN img ELEMENT

DEFINISACU I KOMPONENTU, KOJA CE PREDSTAVLJATI WRAPPER

```
function KontejnerKomponenta(props){  
    return (  
        <div className="container">  
            {props.children}  
        </div>  
    );  
}
```

ZATIM CU SVE OBJEDINITI U NOVU KOMPONENTU KOJA CE SE ZVATI GalerijaKomponenta

```
function GalerijaKomponenta(props){  
    const url = props.url;  
    const brSlika= props.brSlika;  
    const ucitavanje = props.ucitavanje;  
  
    return (  
        <KontejnerKomponenta>  
            {ucitavanje(url, brSlika)}  
        </KontejnerKomponenta>  
    );  
}
```

IAKO TO NIJE PREPORUCIVO, DEFINISACU RENDER-OVANJE U body TAGU (ZASTO TO NIJE PREPORUCIVO OBJASNIO SAM U DOKUMENTU, U KOJEM SAM SE BAVIO React-OM)

```
ReactDOM.render(  
    <GalerijaKomponenta url="./slike/" brSlika={18} ucitavanje={ucitajGaleriju} />,  
    document.getElementsByTagName("body")[0]  
)
```

NAIME, NAJBITNIJE JE DA BUDE RENDER-OVANO SLEDECHE

```
Elements Console Sources Network Performance
... <body> == $0
  <div class="container">
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
    <div>
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
  </div>
</body>
```

DAKLE, OVO SA SLIKE GORE, SAM POGLEDALO U Element SEKCIJI CHROME DEVELOPER TOOLS-A

NAJBITNIJA STAVR JE DA IAM OVAKVU SITUACIJU U POGLEDU HTML-A, NA MOJOJ STRANICI (A SLIKE SU UCITANE U img ELEMENTIMA)

MOGU POSVETITI STILIZOVANJU

DAKLE, I U OVOM SLUCAJU IMPORTOVAO SAM MODULE: index.css | basic.css

SADA CU DEFINISATI STILOVE, U POMENUTIM MODULIMA, ODNOSNO FAJLOVIMA

```

html, body {
    background-color: #fffeead;
    margin: 10px;
}

.container > div {
    display: flex;
    justify-content: center;
    align-items: center;

    font-size: 2em;
    color: #fffeead;
}

.container > div > img {
    width: 100%;
    height: 100%;

    object-fit: cover;
}

```

basic.css

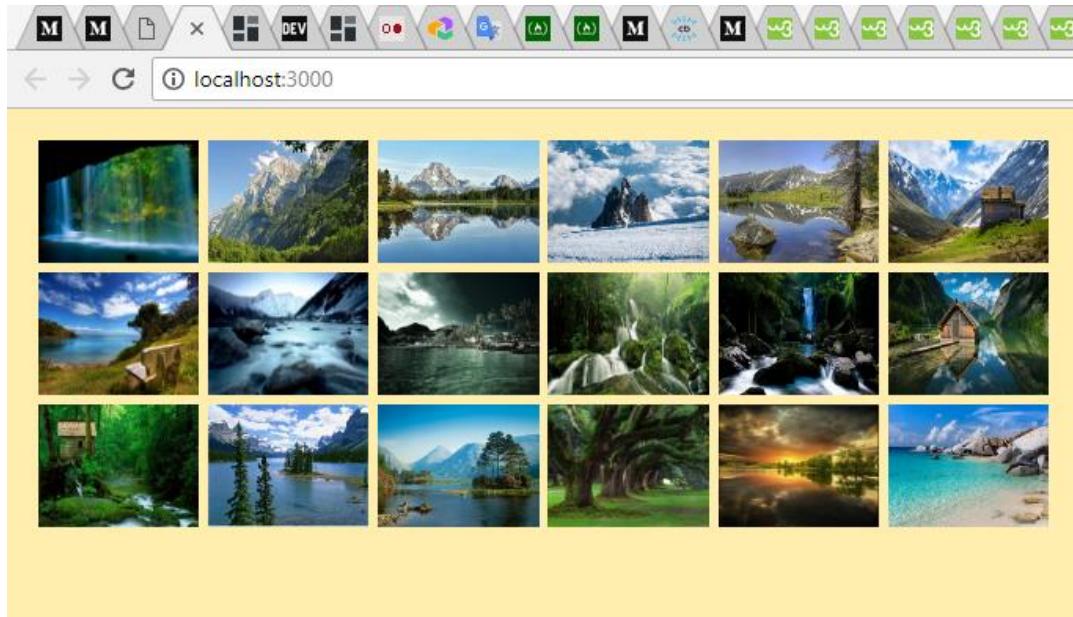
```

.container {
    display: grid;
    grid-gap: 6px;
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
    grid-auto-rows: 78px;
}

```

index.css

ZA SADA MOJ GRID IZGLEDA OVAKO, I BICE RESPONSIVE, JER SAM KORISTIO TEHNIKU SA auto-fit i minmax



SMANJENJEM I POVECANJEM PROZORA, BROJ KOLONA VARIRA; A REDOVI SU IMPLICITNO KREIRANI, I VISOKI SU PO 78px

IVO JE PRILICNO DOBAR GRID, ALI NE SVE OD SLIKA STAJU (FIT), JEDNAKO DOBRO U SVOJE, DATE FRAME-OVE

NAIME, NEKE OD SLIKA SU MOZDA VISE HORIZONTALNE, NEGOT VERTIKALENE, A DRUGE IMAJU OBRNUTO STANJE

A POSTOJI NEKOLIKO FAVORITE SLIKA, KOJIMA JE POTREBAN EXTRA ATTENCION (NAIME, POTREBNO JE DA BUDU PRIKAZANE VECE OD DRUGIH)

SVE TO STO SAM POMENUO JE ODREDJENO I PUTEM KLASA, KOJE SAM DODELIO FRAME-OVIMA
(div ELEMENTIMA, KOJI SU PARENT-I, POMENUTIH img ELEMENATA)

NAIME, NEKI ELEMENTI, NEMAJU class ATRIBUT

NEKI ELEMENTI IMAJU class ATRIBUT SA VREDNOSCU "horizontal"

NEKI ELEMENTI IMAJU class ATRIBUT SA VREDNOSCU "vertical"

A NEKI ELEMENTI IMAJU class ATRIBUT SA VREDNOSCU "big"

PREDHODNA TRI PREDSTAVLJAju IMAGES, ZA KOJE NE MISLIM DA DOBRO STOJE U SVOJIM GRID CELIJAMA (div ELEMENTIMA)

DEFINISACU SADA DA "HORIZONTALNE" SLIKE BUDU DUPLO SIRE, A DA "VERTIKALNE" SLIKE BUDU DUPLO VISE; I DA "VELIKE" SLIKE BUDU DUPLO VISEM, A TAKODJE I DUPLO SIRE

DAKLE, JASNO JE DA U OVOM SLUCAJU MORAM DEFINISATI grid-row I grid-column PROPERTIJE

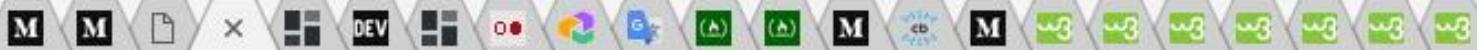
RANIJE SAM TE PROPERTIJE DEFINISAO TAKO STO SAM DODELJIVAO, PRVO BROJ COLUMN LINE-A, OD KOJEG TREBA DA POCINJE GRID ITEM, PA ZNAK "KROZ", PA BROJ COLUMN LINE-A DO KOJEG SE ZAVRSAVA, POMENUTI GRID ITEM (ISTO I U SLUCAJU REDOVA SAMO STO SAM TU POSMATRAO ROW LINES)

POSTO JE grid-column (ILI grid-row), USTVARI SHORTHAND TADA SAM, KAO VREDNOST ZA grid-column-end KORISTIO I JEDNU DRUGACIJU VREDNOST, A TO JE span; KOJIM SAM DEFINISAO PREKO KOLIKO CELIJA (U POGLEDU KOLONE ILI REDA) TREBA DA SE PROSTIRE, ODREDJENI GRID ITEM

KAKO NE BIH DEFINISAO GDE POCINJE ODREDJENI GRID ITEM U OVOM SLUCAJU, JER MI TO NIJE NI POTREBNO, JA MOGU KORISTITI I SAMO span VREDNOST, ZA grid-column (ILI grid-row) PROPERTI; U TOM SLUCAJU GRID ITEM CE POCINJATI OD SVOG DEFAULT COLUMN LINE-A, ILI ROW LINE-A

```
.horizontal {  
    grid-column: span 2;  
}  
  
.vertical {  
    grid-row: span 2;  
}  
  
.big {  
    grid-row: span 2;  
    grid-column: span 2;  
}
```

POGLEDACU, KAKO CE SADA IZGLEDATI, MOJ GRID



← → C

localhost:3000



ONO STO MORAM POPRAVITI, U VEZI GRIDA SA SLIKE GORE, JESU PRAZNA MESTA, JER TO NE
IZGLEDA DOBRO

ZASTO SE POMENUTA PRAZNA MESTA POJAVLJUJU?

KAKO BI TO SHVATIO POCENJU POSMATRANJE GRID-A OD NJEGOVE PRVE CELIJE (A POSMTRACU I
REDOSLED class ATRIBUTA, I NJIHOVIH VREDNOSTI, U HTML-U)



```

...<body> == $0
  <div class="container">
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
    <div>
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="horizontal">
      
    </div>
    <div>
      
    </div>
    <div class="big">
      
    </div>
    <div>
      
    </div>
    <div class="vertical">
      
    </div>
  </div>
</body>

```

DAKLE, GLEDAM PRVI RED:

PRVA SLIKA, DRUGA SLIKA, TRECA, CETVRTA, PETA

SADA GRID SE NASTAVLJA U SLEDECDEM REDU, ALI PROVERICU, KOJA JE VREDNOST class ATRIBUTA, SESTOG FRME-A SLIKE (ODNOSNO SESTOG GRID ITEMA)

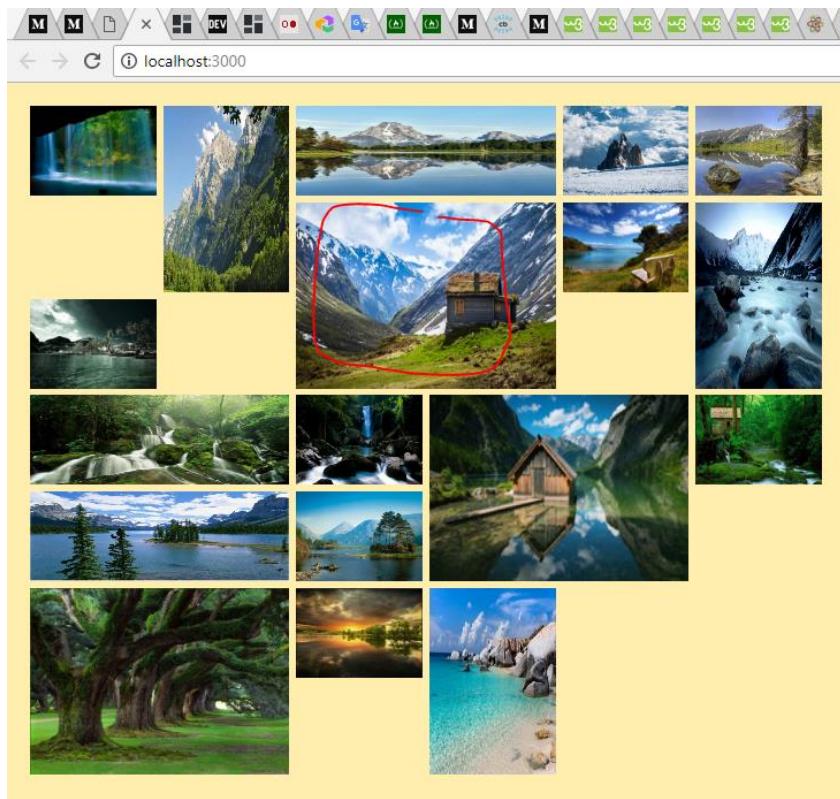
TO JE UPRAVO "big"

SESTA SLIKA JE POKUSALA DA SE UMETNE TAMO GDE JE PRVA CELIJA DRUGOG REDA; ALI KAO STO VIDIM TO NIJE MOGUCE, JER JE SLIKA PREVELIKA, JER JE TO SLIKA KOJA TREBA DA SE PROSTIRE PREKO DVE CELIJE VISINE I DVE CELIJE SIRINE, A DOSTUPNA JOJ JE SAMO JEDNA CELIJA NA POCETKU DRUGOG REDA, I NEMA SANSE DA SE TAMO SMESTI

NAIME, ONDA SE NASTAVLJA TRAZENJE

DRUGA CELIJA DRUGOG REDA JE VEC ZAUZETA

POCEVSI OD TRECE CELIJE DRUGOG REDA PRONADJEN JE DOVOLJAN PROSTOR, GDE SE POMENUTI GRID ITEM MOZE SMESTITI



SADA SE NAKON POMENUTE SESTOG GRID ITEMA, NAIME, NAKON ONOG MESTE, GDE JE ON SADA, NASTAVLJA SA LAYING OUT-OM, SLEDECIH GRID ITEMA, A POMENUTO PRAZNO MESTO, ODNOSNO PRAZNA CELIJA, OSTAJE NEPOPUNJENA

MEDJUTIM, POSTOJI NACIN, NA KOJI SE MOGU POPUNITI TA PRAZNA MESTA, ODNOSNO TE PRAZNE CELIJE GRIDA

TO CU NAIME UCINITI UZ POMOC PROPERTIJA:

grid-auto-flow

```
.container {  
    display: grid;  
    grid-gap: 5.8px;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-auto-rows: 78px;  
  
    grid-auto-flow: row;  
}
```

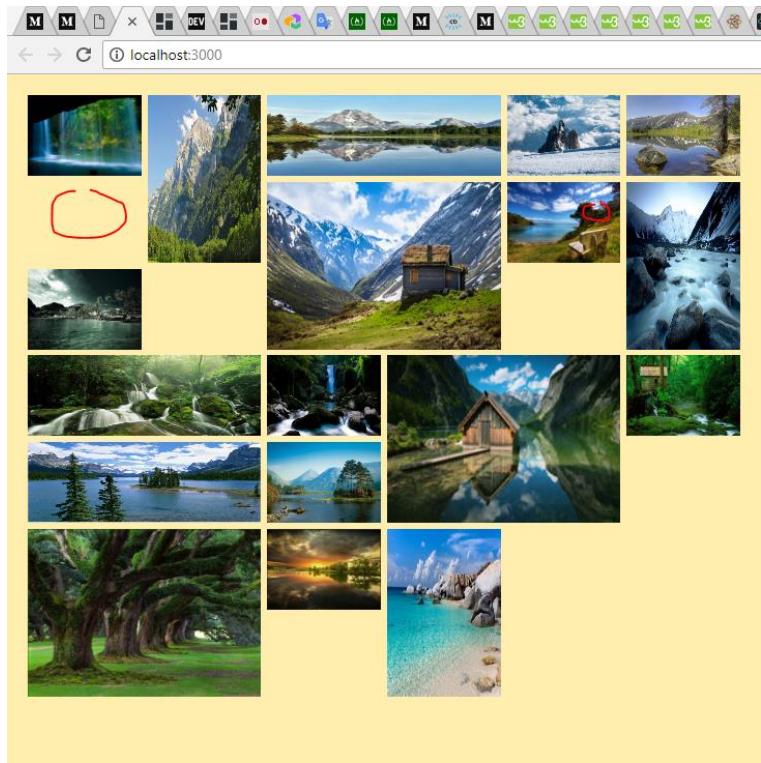
DEFAULT, POMENUTOG PROPERTIJA, JESTE VREDNOST row, STO ZNACI DA SE GRID ITEMI, BIVAJU POLOZENI U JEDAN PO JEDAN RED (UPRAVO, KAKO SAM, GORE I OBJASNIO)

A AKO OVOM PROPERTIJU DODELIM VREDNOST: dense

STO "U PREVODU" ZNACI ALGORITAM GUSTOG PAKOVANJA (**DENSE PACKING ALGORITHM**), KOJI POKUSAVA DA ISPUNI RUPE U GRID-U, I AKO POSTOJE RUPE U GRIDU, ONI SMALLER GRID ITEM-Y, NA CIJI PLACEMENT JESTE DOSAO RED, BICE POMERENI NAZAD, KAKO BI ISPUNILI PRAZNA MESTA

POSTO SAM SAZNAO LOGIKU, IZA POMENUTOGA ALGORITMA, POGLEDACU, KOJI TO GRID ITEM BI MOGAO ISPUNITI PRVU RUPU, NAKON STO POMENUTOM PROPERTIJU DODELIM dense VREDNOST

PREDPOSTAVLJAM DA BI TO MOGAO BITI SLEDECI GRID ITEM:

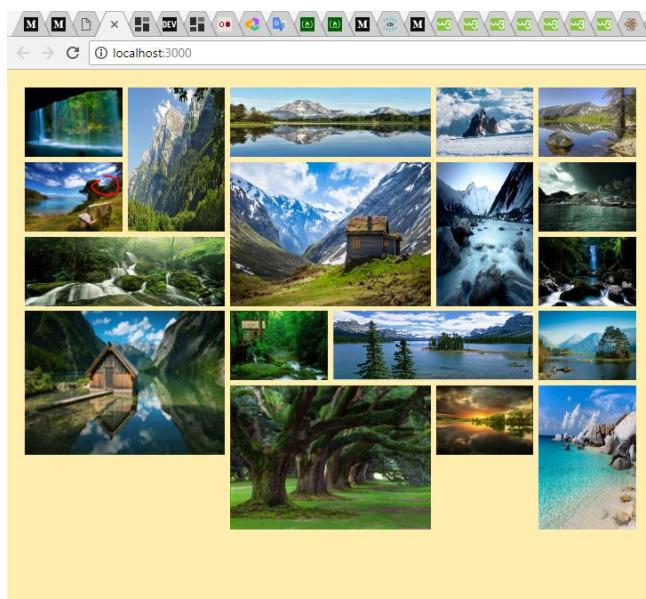


```
.container {  
    display: grid;  
    grid-gap: 5.8px;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-auto-rows: 78px;  
  
    grid-auto-flow: row;  
}
```

→→→→

```
.container {  
    display: grid;  
    grid-gap: 5.8px;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-auto-rows: 78px;  
  
    grid-auto-flow: dense;  
}
```

→→→→



I ZAISTA, POMENUTA PRAZNA CELIJA JESTE ISPUNJENA SA POMENUTIM GRID ITEM-OM

I OVO SVE FUNKCIONISE, BEZ OBZIRA KOJA JE SIRINA GRIDA

OVAKAV GRID, JE VEOMA TESKO KREIRATI, A OVDE JE TO POSTIGNUTO UZ POMOC SAMO DVADESET REDOVA CSS CODE-A

ONO STO, JOS TREBA POMENUTI, JESTE DA JE dense ODLIKA, USTVARI PRIMER NECEGA STO SE ZOVE:

SOURCE ORDER INDEPENDENCE (NEZAVISNOST PORETKA)

A TO PREDSTAVLJA, VELIKU POBEDU ZA CSS, STO ZNACI DA GRID MOZE DA RE-ARANGE, SVOJE GRID ITEM-E, BEZ OBZIRA KAKO SU LAYED OUT, U MARKUP-U (HTML-U)

U OVOM PRIMERU, JA SAM KORISTIO MARKUP, UPRAVO KAO MARKUP ZA CONTENT (ZA SADRZINU); DOK SAM CSS KORISTIO, UPRAVO ZA ONO ZA STA SE CSS I TREBA KORISTITI, A TO JE STILIZOVANJE

I NISAM OGRANICEN, REDOM KOJIM SU INICIJALNO, GRID ITEMI LAYED OUT, TAKO DA SOURCE ORDER INDEPENDENCE, DAJE VELIKU FLEKSIBILNOST

NAMED LINES

ONO STO JE INTERESANTNO, JESTE DA POSTOJI MOGUCNOST I IMENOVANJA COLUMN LINE-OVA I ROW LINE-OVA

PRE NEGO STO SE POZABAVIM POMENUTIM, KREIRACU OPET PRIMER

SET-UP CE, NAIME IZGLEDATI OVAKO

```
<div class="container">
  <div class="header">HEADER</div>
  <div class="menu">MENU</div>
  <div class="content">CONTENT</div>
  <div class="footer">FOOTER</div>
</div>
```

(NESTED U body TAGU)

index.html

```
html, body {
  background-color: #fffeead;
  margin: 0px;
  padding: 10px;
  height: 100%;

  box-sizing: border-box;
}

.container > div {
  font-size: 2em;
  color: #fffeead;

  display: flex;
  justify-content: center;
  align-items: center;
}

.container > div:nth-child(1n) {
  background-color: #96ceb4;
}

.container > div:nth-child(3n) {
  background-color: #88d8b0;
}

.container > div:nth-child(2n) {
  background-color: #ff6f69;
}

.container > div:nth-child(4n) {
  background-color: #ffcc5c;
}
```

```
.container {
  height: 100%;

  display: grid;
  grid-gap: 4px;
  grid-template-columns: 1fr 5fr;
  grid-template-rows: 42px auto 42px;
}

.header {
  grid-column: 1 / 3;
}

.menu {

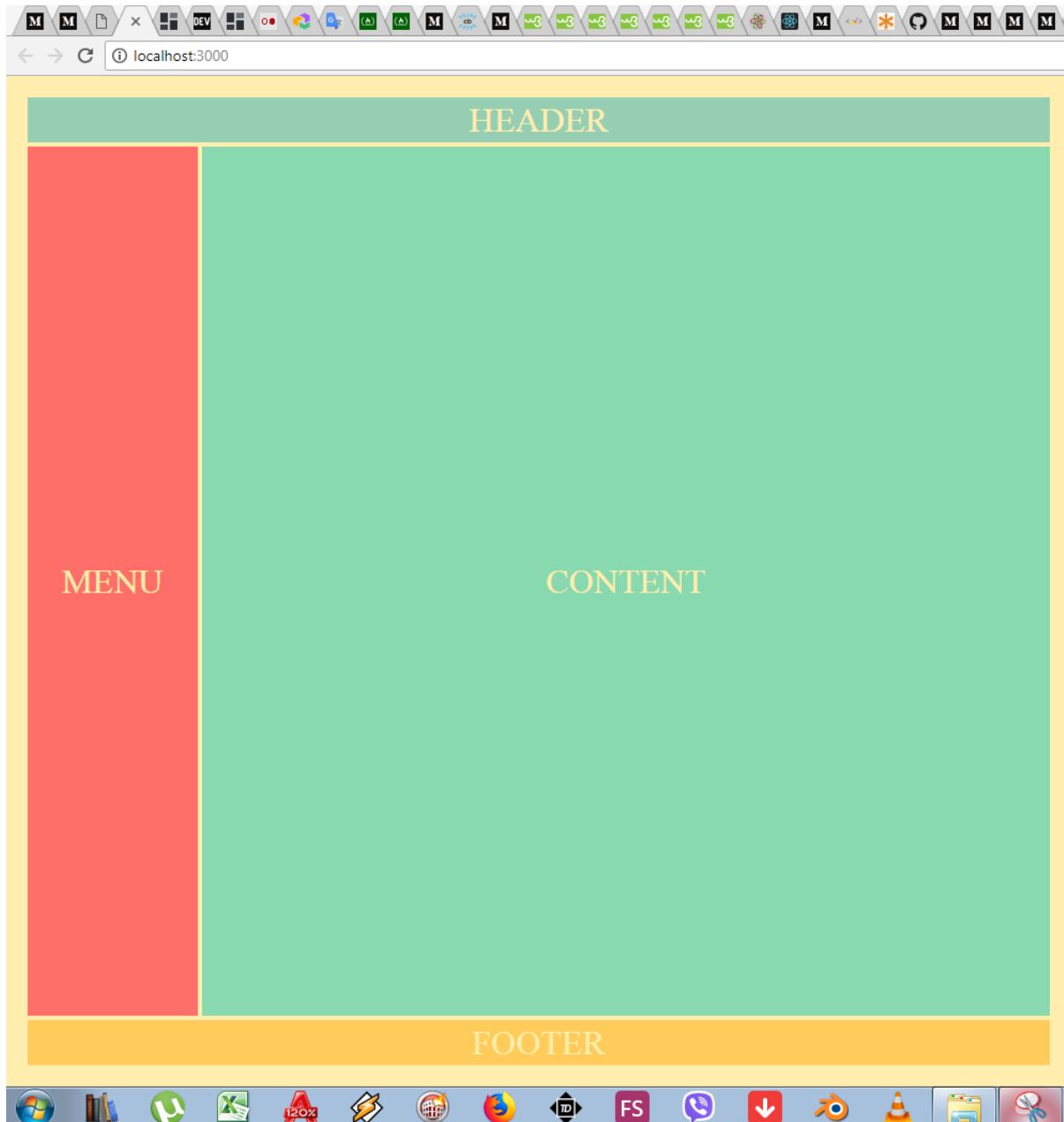
}

.content {
  grid-column: 2 / 3;
}

.footer {
  grid-column: 1 / 3;
}
```

basic.css

index.css



IMENOVANJE COLUMN LINE-OVA, DEFINISEM KAO DEO VREDNOST PROPERTIJA, KOJIM SAM SE VEC BAVIO:

grid-template-columns

ISTO TAKO IMENOVANJE ROW LINE-OVA, DEFINISEM KAO DEO VREDNOST PROPERTIJA, KOJIM SAM SE VEC BAVIO:

grid-template-rows

IMENA LINIJA KOLONA ILI LINIJA REDOVA, DEFINISEM TAKO STO IME, KOJE TREBA DA BUDE OGRANICENO UGLASTIM ZAGRADAM POSTAVIM, PORED VREDNOSTI, KOJA DEFINISE VISINU REDA (ILI SIRINU KOLONE, U SLUCAJU, GORE PRVOG PROPERTIJA); UPROSTENO RECENO, IMENOVANJA LINIJA VRSIM IZMEDJU VREDNOSTI DIMENZIJA ZA REDOVE (ILI KOLONE), I NE TREBA DODATNO OBJASNjavati zasto, jer je sama, ta radnja poprilično sugestivna u pogledu mesta, gde se to nalaze linije, jer jednostavno gledajuci vrednost propertija i gledajuci sam grid, jasno je gde se definisu imenovanja linija

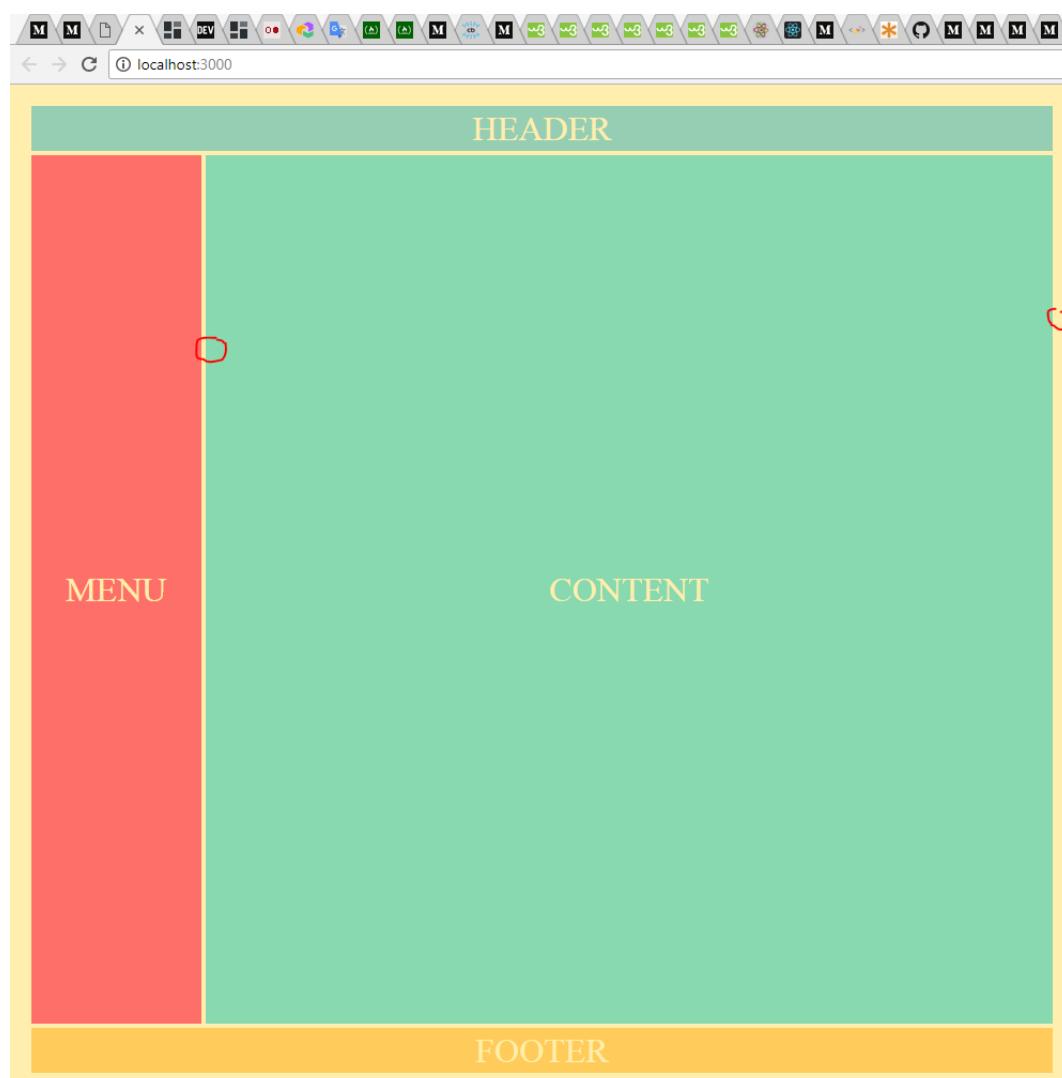
SADA CU DEFINISATI IMENA, NEKIH LINIJA; ODNOSNO PRVO STO CU DEFINISATI JESTE IME COLUMN LINIJE OD KOJE POCINJE CONTENT ITEM, I IME COLUMN LINIJE DO KOJE SE ZAVRSVA CONTENT ITEM

```
.container {  
    height: 100%;  
  
    display: grid;  
    grid-gap: 4px;  
    grid-template-columns: 1fr [content-start] 5fr [content-end];  
  
    grid-template-rows: 42px auto 42px;  
}
```

POSTO SAM IMENOVAO, POMENUTE LINIJE, SADA MOGU UPOTREBITI, UPRAVO IMENA POMENUTIH LINIJA, KAO VREDNOSTI, PROPERTIJA `grid-column`, CONTENT ELEMENTA, ODNOSNO GRID ITEM-A; UMESTO DA KORISTIM KONKRETAN BROJ LINIJE, STO JE SLUCAJ BIO RANIJE

```
.content {  
    grid-column: 2 / 3;  
}  
→→→→ .content {  
    grid-column: content-start / content-end;  
}
```

POSTO JE REC O ISTIM LINIJAMA, POMOCU KOJIH JE DEFINISANO PROSTIRANJE CONTENT ELEMENTA, NISTA SE NECE PROMENITI (SAMO CU NA SLEDECOJ SLICI, OZNACITI TE LINIJE)



DA SAM ISTO ZELEO DA URADIM I U SLUCAJU HEADER ELEMENTA, JASNO MI JE GDE SAM MORAO IMENOVATI LINIJU OD KOJE POCINJE

ALI, NJBOLJE JE DA TOJ LINIJI DAM IME `glavno-start`, ILI `main-start` (OZNACENO CRVENOM NA SLEDECOJ SLICI), UMESTO DA JE IMENUJEM KAO POCETAK, SAMOG HEADER-A; TO GOVORIM IZ RAZLOGA STO TA LINIJA, NIJE SAMO POCETAK HEADERA, VEC TREBA DA BUDE POCETAK I, DRUGIH OSTALIH GRID ITEMA

ISTI SLUCAJ BI BIO I ZA LINIJU DO KOJE SE ZAVRSVA FOOTER ITEM, I NJOJ CU DATI IME `main-end` (OZNACENO ZELENOM NA SLEDECOJ SLICI)

ALI AKO POGLEDEM, JOS JEDNOM VREDNOST, `grid-template-column` PROPERTIJA, VIDECU DA SAM VEC DODELIO IME POPMENUTOJ LINIJI DO KOJE SE ZAVRSVA FOOTER, JER TO JE I LINIJA DO KOJE SE ZAVRСAVA I CONTENT

NE MARI, ZATO STO JEDNA LINIJA, MOZE IMATI I VISE IMENA; IZMEDJU UGLASTIH ZAGRADA, MOZE STAJATI VISE IMENOVANJA, SAMO IZMEDJU NJIH TREBA DA SE NALAZI RAZMAK, ODNOSNO SPACE

```
.container {  
    height: 100%;  
  
    display: grid;  
    grid-gap: 4px;  
    grid-template-columns: [main-start] 1fr [content-start] 5fr [content-end main-end];  
  
    grid-template-rows: 42px auto 42px;  
}
```

SADA CU I ZA FOOTER, UMESTO UZ POMOC REDNIH BROJAVA COLUMN LINIJA, DEFINISATI GDE SE POMENUTI GRID ITEM PROSTIRE UZ POMOC IMENOVANJA, POMENUTIH LINIJA

```
.header {  
    grid-column: 1 / 3;  
}  
→→→→ .header {  
    grid-column: main-start / main-end;  
}
```

NECU PRIKAZIVATI LAYOUT, JER SE NISTA NECE PROMENITI, JER I U OVOM SLUCAJU, REC JE O ISTIM LINIJAMA, SAMO STO SU PRVI PUT REFERENCIRANE UZ POMOC SVOJIH REDNIH BROJAVA, A DRUGI PUT UZ POMOC IMENOVANJA, KOJE SAM I M JA DODELIO

I FOOTER MOZE DOBITI, ISTE VREDNOSTI

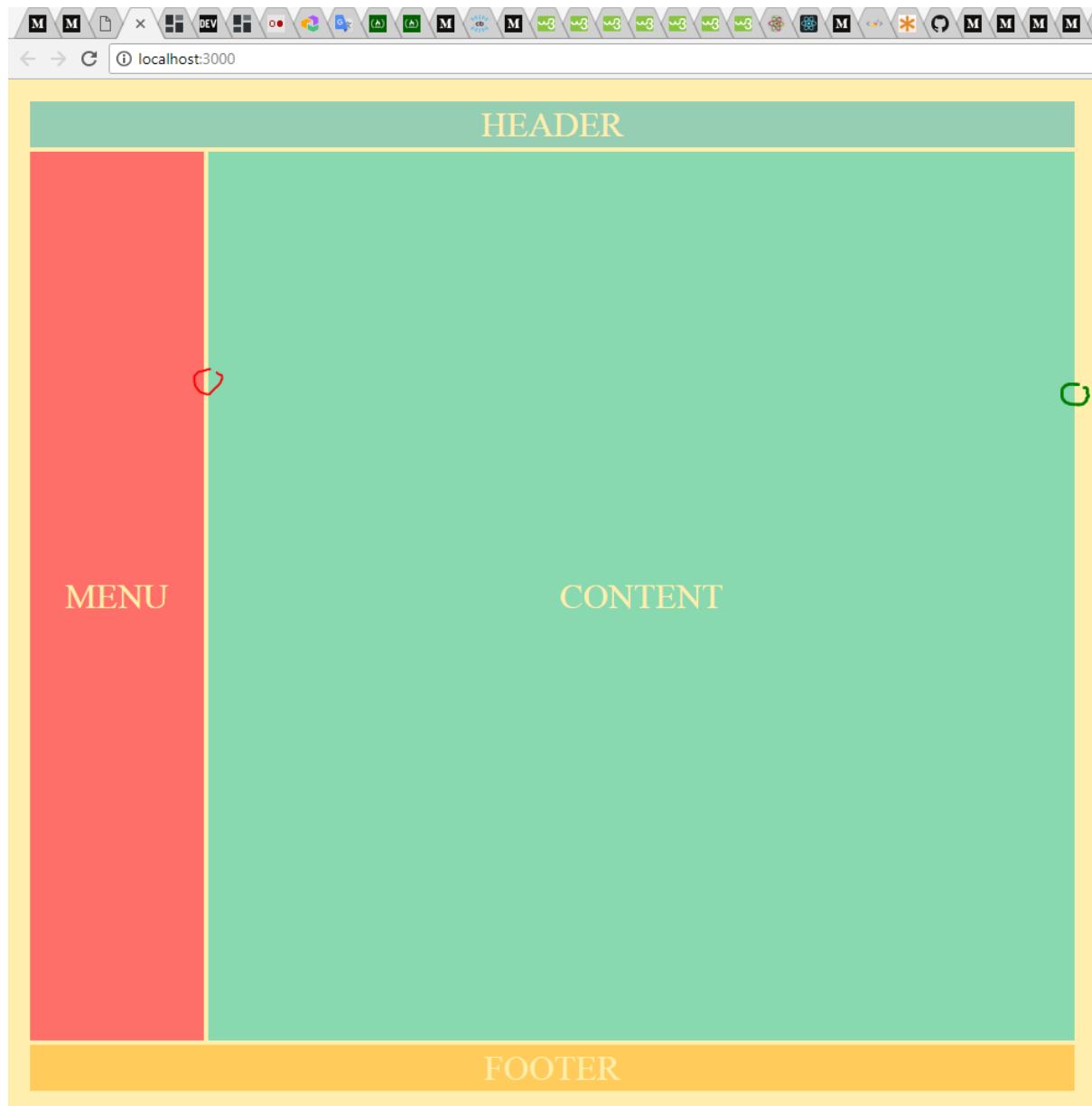
```
.footer {  
    grid-column: main-start / main-end;  
}
```

ISTU KONVENCIJU, MOGU ISKORISTITI I U SLUCAJU LINIJA REDOVA

TAKO DA CU SADA TO URADITI

```
.container {  
    height: 100%;  
    display: grid;  
    grid-gap: 4px;  
    grid-template-columns: [main-start] 1fr [content-start] 5fr [content-end main-end];  
    grid-template-rows: [main-start] 42px [content-start] auto [content-end] 42px [main-end];  
}
```

VEROVATNO SE PITAM, ZASTO JE DASH (CRTICA), DEO SVAKOG IMENOVANJA LINIJE
E UPRAVO ZATO STO TA CRTICA, PRUZA NESTO MAGIJE
NAIME, POGLEDACU OPET, MOJ LAYOUT



NAIME, POSTO SAM LINIJU, OZNACENU CRVENOM NA GORNJOJ SLICI, NAZVAO content-start, I
POSTO SAM LINIJU, ONZNACENU ZELOMOM NA GORNJOJ SLICI, NAZVAO content-end; GRID
PREPOZNAJE IZMEDJU TIH LINIJA UPRAVO content KOLONA, ILI KOLONE (AKO JE TAKAV SLUCAJ)
TAKO DA SAM VREDNOST grid-column PROPERTIJA, POMENUTOG CONTENT ITEMA, USTVARI
MOGAO DEFINISATI, KAO SAMO content

```
.content {  
    grid-column: content-start / content-end;  
}
```



```
.content {  
    grid-column: content;  
}
```

A POSTO SE I HEADER I FOOTER SPAN-UJU OD main-start COLUMN LINE-A, PA DO main-end COLUMN LINE-A, NA ISTI NACIN SAM MOGAO DEFINISATI I NJIHOVE grid-column PROPERTIJE; ODNOSNO POMENUTOM PROPERTIJU CU DODELITI VREDNOST main, U SLUCAJU HEADER-A I U SLUCAJU FOOTER-A

```
.header {  
    grid-column: main;  
}  
  
.menu {  
}  
  
.content {  
    grid-column: content;  
}  
  
.footer {  
    grid-column: main;  
}
```

A KAKO BI STVAR ODVEO JOS KORAK DALJE, JA SAM MOGAO KORISTITI I PROPERTI grid-area, U SLUCAJU JEDNOG OD GRID ITEM-A

NAIME, POSTO JE CONTENT GRID ITEM, OKRUZEN SA SLEDECIM LINIJAMA:

U SLUCAJU LINIJA KOLONA: content-start content-end

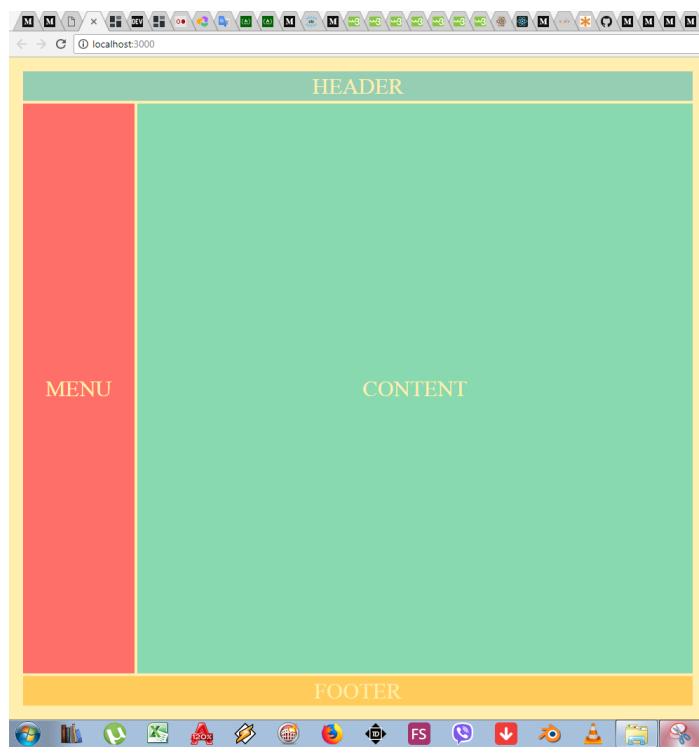
U SLUCAJU LINIJA REDOVA: content-start content-end

MOGAO SAM DEFINISATI SLEDECE:

```
.content {  
    grid-column: content;  
}
```



```
.content {  
    grid-area: content;  
}
```



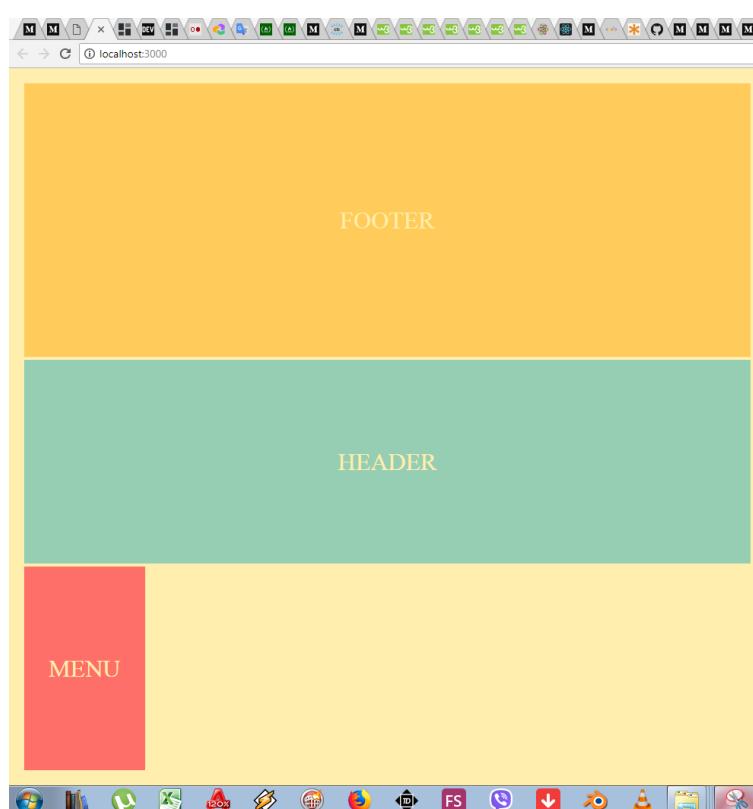
JA MOGU BITI DOVEDEN U ISKUSENJE DA NA ISTI PROPERTI DEFINISEM I ZA FOOTER ELEMENT

```
.footer {  
    grid-column: main;  
}  
  
→→→→ .footer {  
    grid-area: main;  
}
```

ALI TADA CE CELI LAYOUT BITI POREMECEN (BROKEN), JER KAO STO MOGU DA PRIMETIM FOOTER JE OGRANICEN SLEDECIM LINIJAMA:

ZA COLUMN LINES: main-start main-end

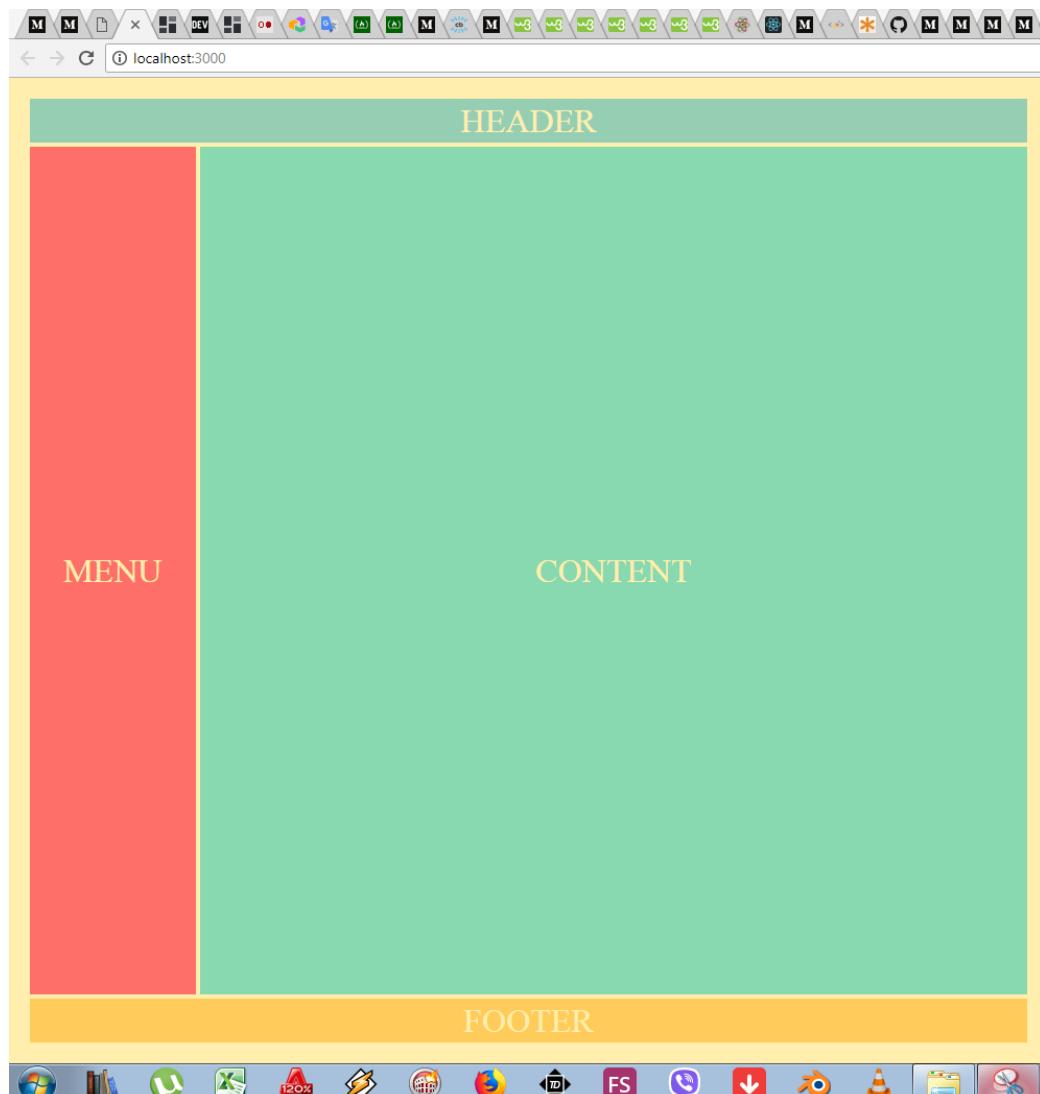
ZA ROW LINES: content-end main-end



DAKLE, DA BIH POMENUTO BILO DEFINISANO NA NACIN KAO STO SAM ZELEO, GRID ITEM BI MORAO BITI OGRANICEN OD STRANE CETIRI main LINIJE, STO NIJE SLUCAJ

SADA CU ISPRAVITI, PREDHODNU GRESKU

```
.container {  
    height: 100%;  
  
    display: grid;  
    grid-gap: 4px;  
    grid-template-columns: [main-start] 1fr [content-start] 5fr [content-end main-end];  
  
    grid-template-rows: [main-start] 42px [content-start] auto [content-end] 42px [main-end];  
}  
  
.header {  
    grid-column: main;  
}  
  
.menu {  
}  
  
.content {  
    grid-area: content;  
}  
  
.footer {  
    grid-column: main;  
}
```



justify-content I align-content

U OVOJ LEKCIJI, SAZNACU KAKO DA JUSTIFY, CELI CONTENT GRID-A (DAKLE SVE GRID ITEME)

ZA POCETAK CU IMATI OVAKAV SET UP

```
<div class="container">
<div>1</div>
<div>2</div>
<div>3</div>
<div>4</div>
<div>5</div>
<div>6</div>
</div>
```

index.html

```
html, body {
    background-color: #ffeedad;
    margin: 0px;
    padding: 10px;
    height: 100%;

    box-sizing: border-box;
}

.container > div {
    font-size: 2em;
    color: #ffeedad;

    display: flex;
    justify-content: center;
    align-items: center;
}

.container > div:nth-child(1n) {
    background-color: #96ceb4;
}

.container > div:nth-child(3n) {
    background-color: #88d8b0;
}

.container > div:nth-child(2n) {
    background-color: #ff6f69;
}

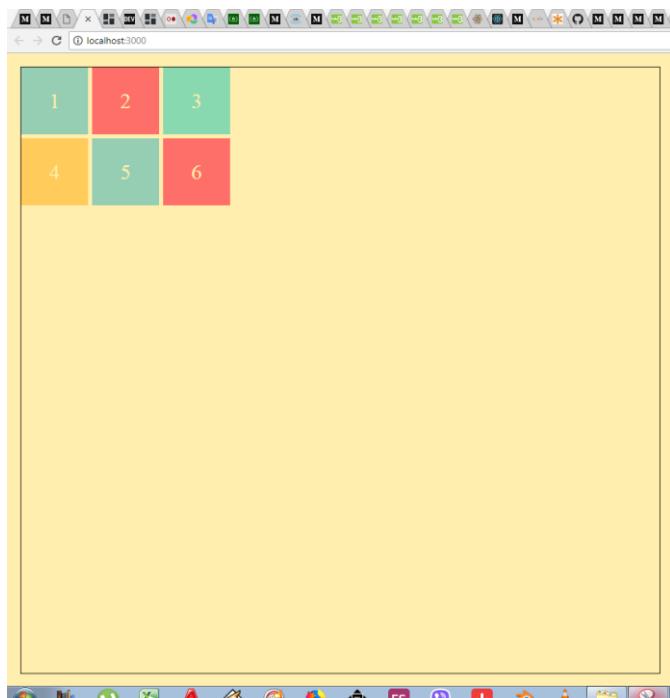
.container > div:nth-child(4n) {
    background-color: #ffcc5c;
}
```

basic.css

```
.container {
    border: black solid 1px;
    height: 100%;

    display: grid;
    grid-gap: 6px;
    grid-template-columns: repeat(3, 100px);
    grid-template-rows: repeat(2, 100px);
}
```

index.css



GRID, JE NAIME OGRANICEN CRNIM BORDER-OM, KAO STO SE VIDI NA SLICI GORE; A TO SAM URADIO KAKO BI BILE VIDLJIVE IVICE GRID CONTAINER-A

TAKODJE SAM SE POSTARAO DA JE GRID POTPUNO RESPONSIVE I DA POSTOJI MALO PROSTORA OKO NJEGA

KAO STO VIDIM SA SLIKE GORE, GRID ITEMI, ZA SADA ZAUZIMAJU (OCCUPY), SAMO MALI DEO CELOKUPNOG GRID CONTAINER-A

POMENUTO MI JE VAZNO, KAKO BIH MOGAO POCETI UCITI KAKO DA IZVRSIM ALIGNMENT I JUSTIFICATION, GRID ITEM-A, ODNOSNO CONTENT-A, GRID CONTEINER-A

POCECU SA OBJASNJENEM ZA **JUSTIFICATION-ON**, KOJI DEFINISE DITRIBUIRANJE PROSTORA, IZMEDJU I OKO GRID ITEM-A, U ODNOSU NA ROW AXIS-A (SMER I PRAVAC POMENUTE OSE JESU **LEFT-TO-RIGHT HORIZONTALA**)

POMENUTO MOGU OSTVARITI UZ POMOCU `justify-content` PROPERTIJA

VREDNOSTI PROPERTIJA:

start (DEFAULT) end center space-between space-around space-evenly

A **ALIGNMENT DEFINISE DITRIBUIRANJE PROSTORA, IZMEDJU I OKO GRID ITEM-A, U ODNOSU NA COLUMN AXIS-A (SMER I PRAVAC POMENUTE OSE JESU TOP-TO-BOTTOM VERTIKALA)**

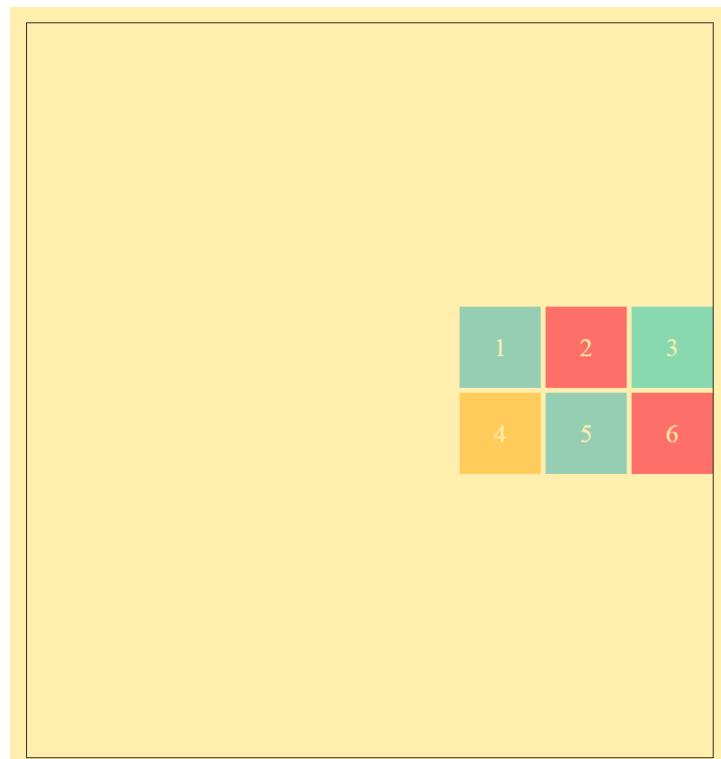
POMENUTO MOGU OSTVARITI UZ POMOCU `align-content` PROPERTIJA

VREDNOSTI PROPERTIJA:

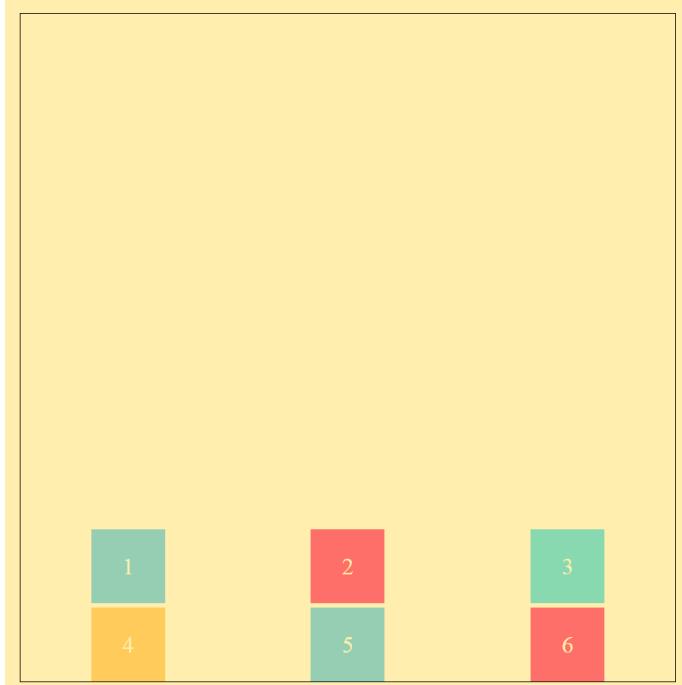
start: (DEFAULT) end center space-between space-around space-evenly

NECU DODATNO OBJASNJAVAĆI STA CE SE DOGADJATI, SA GRID ITEMIMA U GRID CONTAINERU, JER SAM SE SLICNIM BAVIO, KADA SAM SE UPOZNAVAO SA JUSTIFICATIONOM I ALIGNMENTOM FLEX ITEMA U FLEX CONTAINER-U; VEC CU SAMO PUSTITI PRIMERE DA GOVORE SAMI ZA SEBE

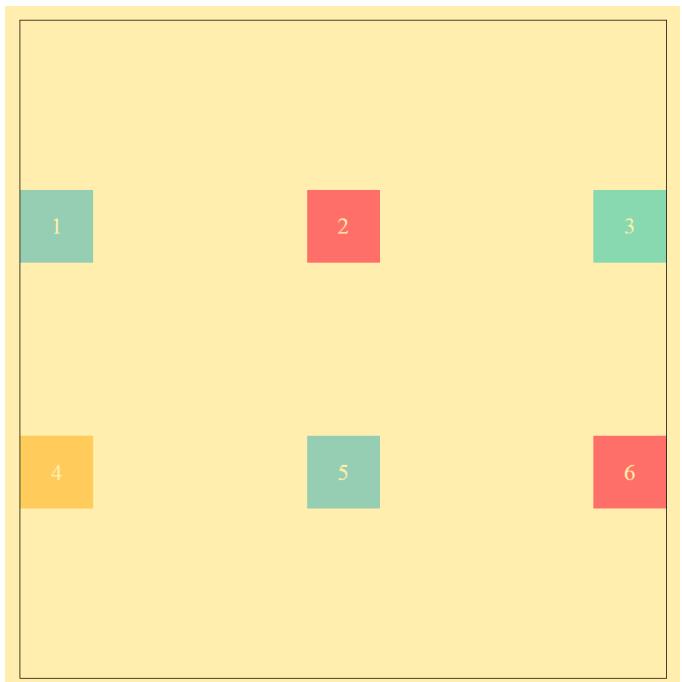
```
.container {  
    border: black solid 1px;  
    height: 100%;  
  
    display:grid;  
    grid-gap: 6px;  
    grid-template-columns: repeat(3, 100px);  
    grid-template-rows: repeat(2, 100px);  
  
    justify-content: end;  
    align-content: center;  
}
```



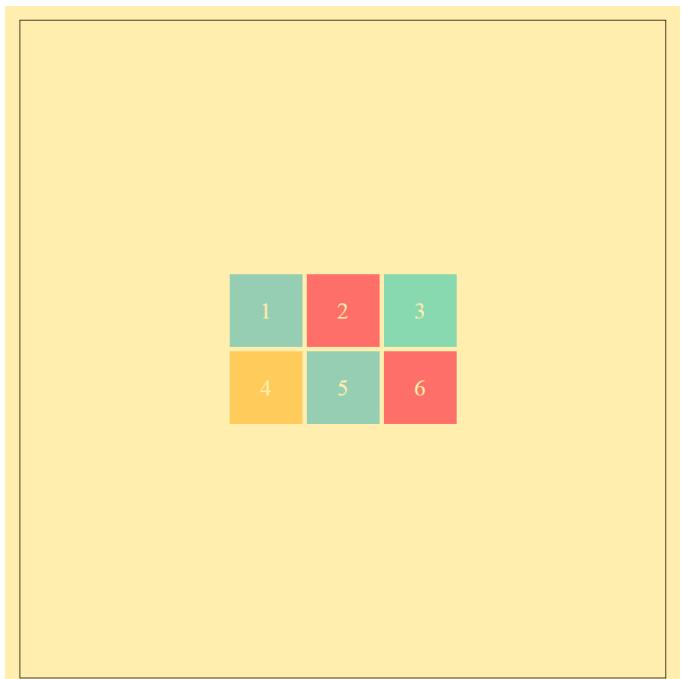
```
.container {  
  border: black solid 1px;  
  height: 100%;  
  
  display:grid;  
  grid-gap: 6px;  
  grid-template-columns: repeat(3, 100px);  
  grid-template-rows: repeat(2, 100px);  
  
  justify-content: space-around;  
  align-content: end;  
}
```



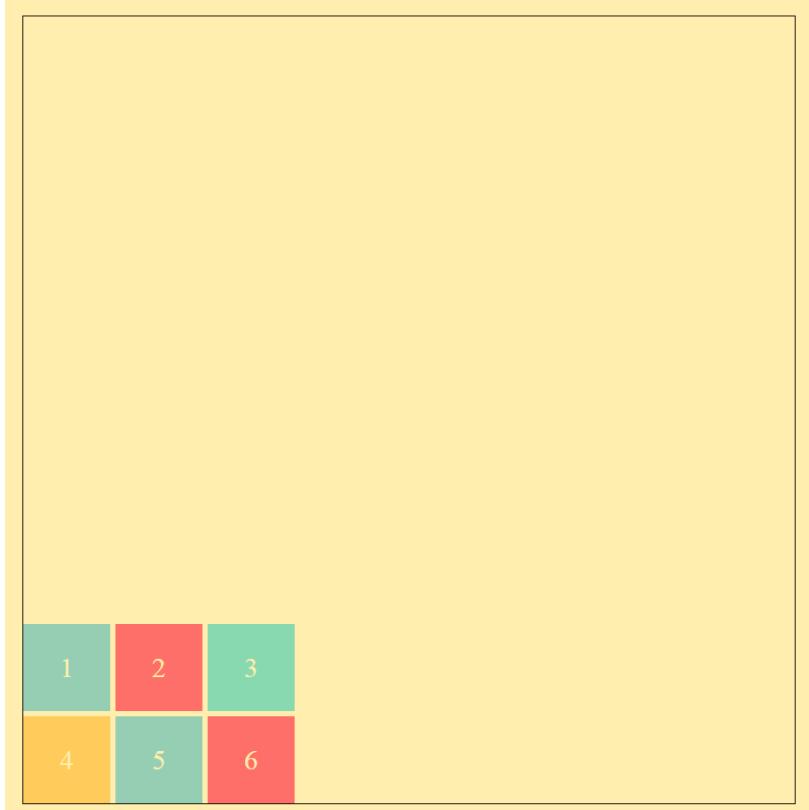
```
.container {  
  border: black solid 1px;  
  height: 100%;  
  
  display:grid;  
  grid-gap: 6px;  
  grid-template-columns: repeat(3, 100px);  
  grid-template-rows: repeat(2, 100px);  
  
  justify-content: space-between;  
  align-content: space-evenly;  
}
```



```
.container {  
  border: black solid 1px;  
  height: 100%;  
  
  display:grid;  
  grid-gap: 6px;  
  grid-template-columns: repeat(3, 100px);  
  grid-template-rows: repeat(2, 100px);  
  
  justify-content: center;  
  align-content: center;  
}
```



```
.container {  
    border: black solid 1px;  
    height: 100%;  
  
    display: grid;  
    grid-gap: 6px;  
    grid-template-columns: repeat(3, 100px);  
    grid-template-rows: repeat(2, 100px);  
  
    /*justify-content: center;*/  
    align-content: end;  
}
```



justify-items I align-items (justify-self align-self)

I POMENUTO CU OBJASNITI PREKO PRIMERA, A ZA POCEAK CU IMATI SLEDECI SET-UP

```
<div class="container">
  <div class="header">HEADER</div>
  <div class="menu">MENU</div>
  <div class="content">CONTENT</div>
  <div class="footer">FOOTER</div>
</div>
```

index.html

```
html, body {
  background-color: #ffeedad;
  margin: 0px;
  padding: 10px;
  height: 100%;

  box-sizing: border-box;
}

.container > div {
  font-size: 2em;
  color: #ffeedad;
}

.container > div:nth-child(1n) {
  background-color: #96ceb4;
}

.container > div:nth-child(3n) {
  background-color: #88d8b0;
}

.container > div:nth-child(2n) {
  background-color: #ff6f69;
}

.container > div:nth-child(4n) {
  background-color: #ffcc5c;
}
```

basic.css

```
.container {
  height: 100%;

  display: grid;
  grid-gap: 5px;
  grid-template-columns: repeat(12, 1fr);
  grid-template-rows: 148px auto 148px;
}

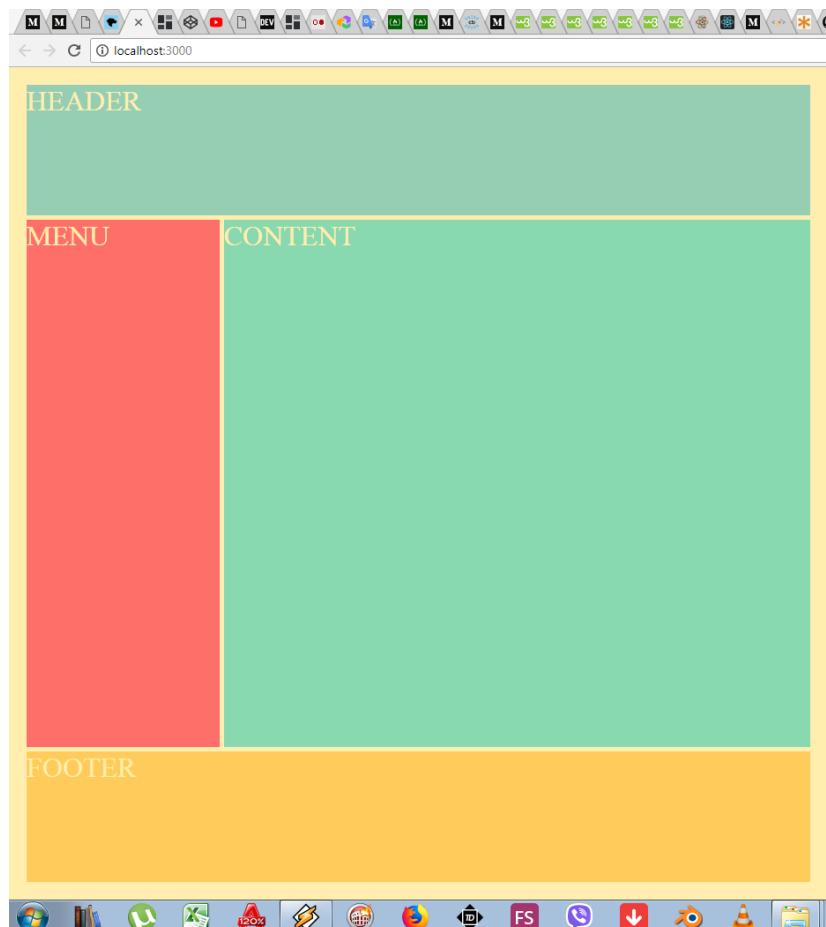
.header {
  grid-column: 1 / -1;
}

.menu {
  grid-column: 1 / 4;
}

.content {
  grid-column: 4 / -1;
}

.footer {
  grid-column: 1 / -1;
}
```

index.css



NAIME, ONO O CEMU CE SE RADITI U OVOJ LEKCIJI JESTE NESTO STO SE ZOVE BOX ALIGNMENT U CSS GRID-U; SA POMENUTIM SAM SE TAKODJE SUSREO I PRILIKOM UPOZNAVANJA SA FLEXBOX MODELOM

ONO CIME SAM SE TREBAO BAVITI SADA JESU POMENUTI PROPERTIJI:

justify-items align-items

ALI JA SE NECU, ODMAH BAVITI POMENUTIM PROPERTIJIM, KOJI SU PROPERTIJI GRID CONTAINERA STO ZNACI DA SE, GORE POMENUTI PROPERTIJI PRIMENUJU NAD SVIM GRID ITEMIMA JA CU SE, PRVO POZABAVITI PROPERTIJIMA, KOJI SU PROPERTIJI, SAMOG GRID ITEM-A, I KOJIMA SE POSTIZE ISTO ONO STO SE POSTIZE, I UZ KORISCENJE GORE NAVEDENIH PROPERTIJA; STO ZNACI DA CE SLEDECIM PROPERTIJIMA:

justify-self align-self

BITI APLICIRANI STILOVI NAD ODREDJENOM GRID ITEMU, ZA KOJI BUDEM DEFINISAO VREDNOSTI, POMENUTIH PROPERTIJA

VREDNOSTI PROPERTIJA:

stretch (DEFAULT) start end center baseline first baseline
last baseline

NAIME, NISAM REKAO KAKO SE VRSI ALIGNMENT ILI JUSTIFICATION GRID ITEM-A, U SLUCAJU POMENUTIH PROPERTIJA

NAIME, ONO KAKO JA RAZUMEM, POMENUTE PROPERTIJE, MOGU RECI DA SE POMENUTI PROPERTIJIMA POSTIZE ALIGNMENT GRID ITEMA, U ODNOSU "NJIH SAME", ODNOSNO, BLIZE RECENO U ODNOSU NA NEKE KARAKTERISTIKE, SAMOG GRID ITEM-A, NAD KOJIM VRSIM JUSTIFICATION ILI ALIGNMENT; ALI PORED TOGA I U ODNOSU NA COLUMN AXIS ODNOSNO ROW AXIS

DAKLE, KAO STO SAM REKAO, KAKO BI NAJBOLJE RAZUMEAO POMENUTE PROPERTIJE, I NJIHOVE VREDNOSTI, DEFINISACU IH, ZA POCETAK SAMO ZA CONTENT ELEMENT, I NECU IH DODATNO OBJASNJAVAATI, JER MISLIM DA CE PRIMERI GOVORITI SAMI ZA SEBE

POSTO ZNAM DA JE VREDNOST stretch DEFAULT VREDNOST, NECU POCETI SA POMENUTOM VREDNOSCЮ

POCECU SA center VREDNOSCЮ ZA justify-self PROPERTI

```
.content {  
    grid-column: 4 / -1;  
    justify-self: center;  
}
```

HEADER

MENU

CONTENT

FOOTER

SADA CU DEFINISATI I align-self PROPERTI, KOJEM CU DODELITI ISTO center VREDNOST

```
.content {  
    grid-column: 4 / -1;  
    justify-self: center;  
    align-self: center;  
}
```

HEADER

MENU

CONTENT

FOOTER

MISLIM DA JE JASNO NA KOJI SE NACIN SE POZICIONIRAO CONTENT ELEMENT, U OVOM SLUCAJU; ZATO NECU DODATNO OBJASNJAVAĆI ONO STO SE DOGODILO, VEC CU ISPROBATI I OSTALE VREDNOSTI, ZA POMENUTE PROPERTIJE

HEADER

MENU

CONTENT

FOOTER

```
.content {  
  grid-column: 4 / -1;  
  justify-self: end;  
  align-self: start;  
}
```

HEADER

MENU

CONTENT

FOOTER

```
.content {  
  grid-column: 4 / -1;  
  justify-self: end;  
  align-self: stretch;  
}
```

HEADER

MENU

CONTENT

FOOTER

```
.content {  
    grid-column: 4 / -1;  
    justify-self: baseline;  
}
```

HEADER

MENU

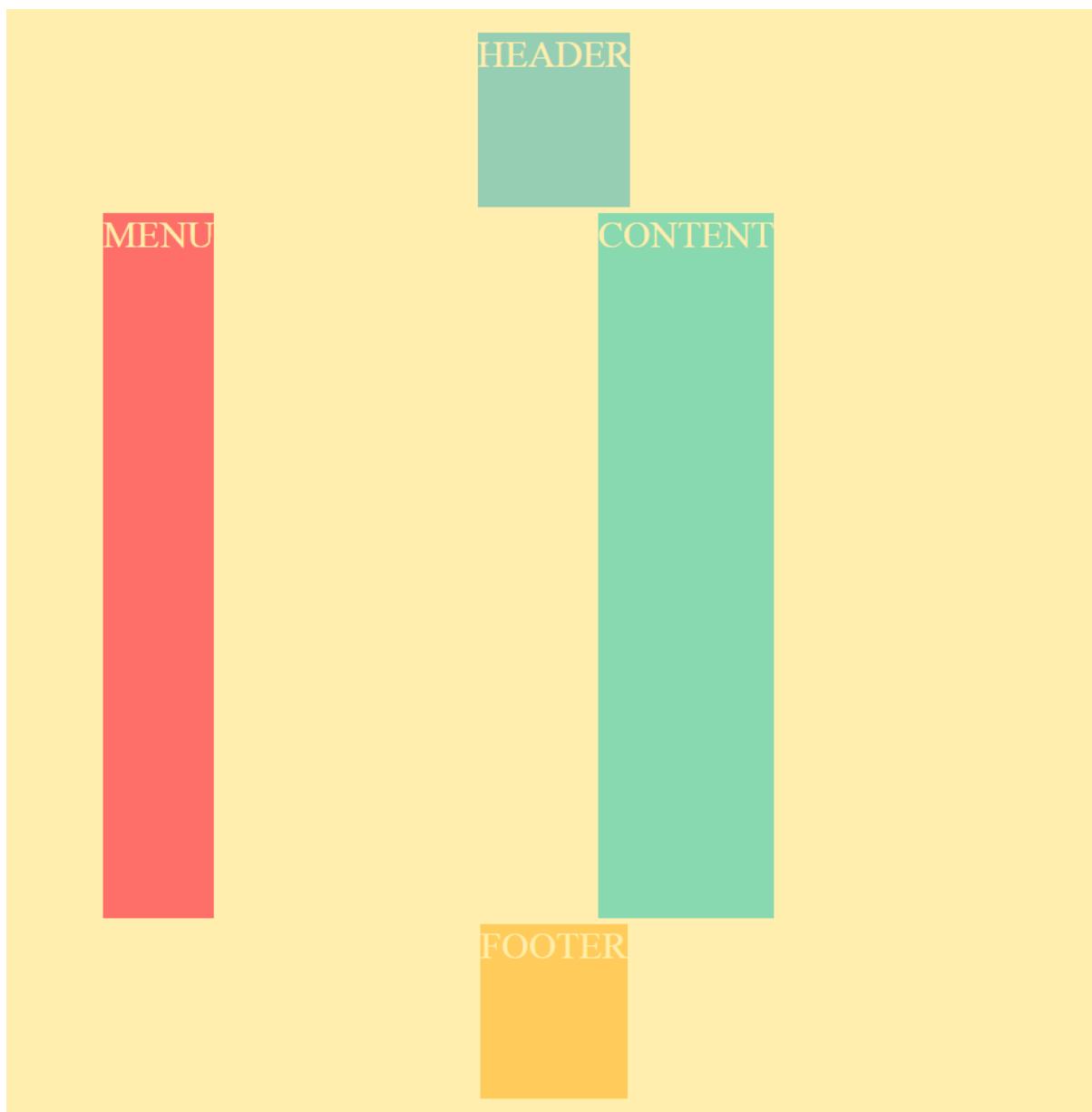
CONTENT

FOOTER

```
.content {  
    grid-column: 4 / -1;  
    align-self: baseline;  
}
```

SADA CU PROBATI, PREDHODNE STILOVE APLICIRAM NA SVE GRID ITEME ZAJEDNO, TAKO STO CU DEFINISATI align-items I justify-items PROPERTIJE, ZA GRID CONTAINER

```
.container {  
  height: 100%;  
  display: grid;  
  grid-gap: 5px;  
  grid-template-columns: repeat(12, 1fr);  
  grid-template-rows: 148px auto 148px;  
  
  justify-items: center;  
  align-items: stretch;  
}
```



OVERLAPPING ITEMS

TRUDICU SE DA BUDEM STO SAZETIJI I DA SVE PRIKAZEM PREKO PRIMERA, U CILJU USTEDE VREMENA

ZA POCETAK CU IMATI SLEDECİ SET-UP

```
<div class="grid-container">
  <div class="item-1">1</div>
  <div class="item-2">2</div>
</div>
```

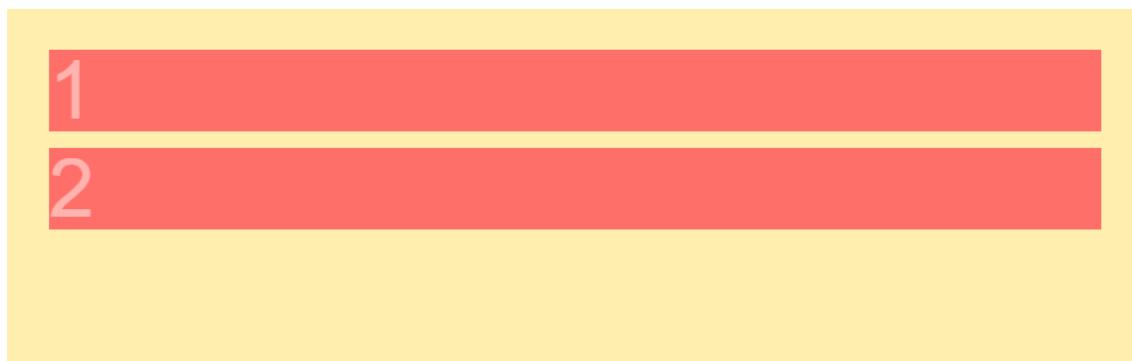
index.html

```
.grid-container > div {
  font-size: 2em;
  color: rgba(255, 255, 255, 0.48);
  background-color: #ff6f69;
  margin: 0.2em;

  font-family: 'Overpass', sans-serif;
  font-size: 4em;

  line-height: 1;
}
```

index.css



UCINICU DA ELEMENT, KOJI IMA CSS class ATRIBUT, SA VREDNOSU .grid-container, ZAISTA POSTANE GRID CONTAINER

```
.grid-container {
  display: grid;
  grid-template-columns: [pocetak] 4fr [srednja] 2fr [kraj];
  grid-template-rows: [pocetak] 86px [srednja] 98px [kraj];
  grid-gap: 18px;
}
```



SADA CU UZ POMOC grid-column-start I grid-column-end, ODNOSNO grid-row-start I grid-row-end PROPERTIJA (USTVARI NJIHOVIH SHORTHAND PROPERTIJA) DEFINISATI GDE CE SE GRID ITEMI PROSTIRATI

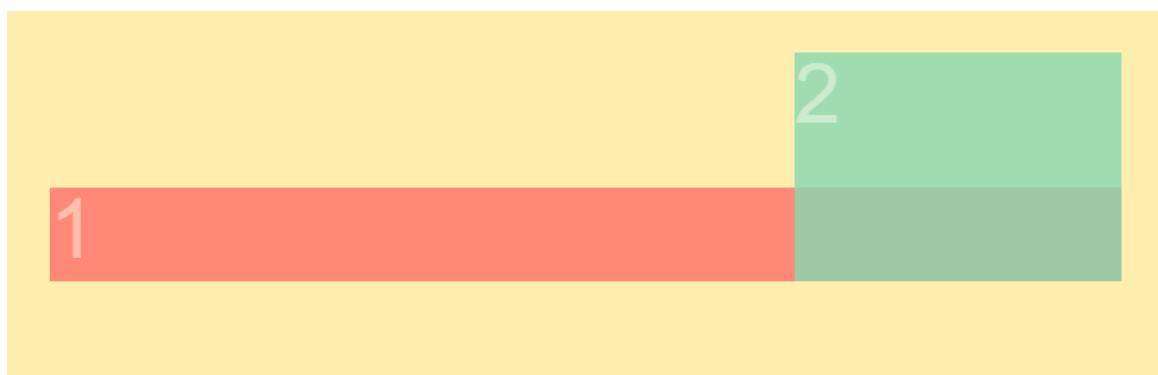
MEDJUTIM, PRE TOGA, JEDINO STO CU URADITI, JESTE UKLANJANJE background-color PROPERTIES IZ .grid-container > div SELEKTORA, KAKO BI SPRECIO MOGUCI OVERWRITING BOJE POZADINE

```
.grid-container > div {  
    font-size: 2em;  
    color: rgba(255, 255, 255, 0.48);  
    /*background-color: #ff6f69;*/  
    margin: 0.2em;  
  
    font-family: 'Overpass', sans-serif;  
    font-size: 4em;  
  
    line-height: 1;  
}
```

ONO STO CU, PORED POZICIONIRANJA GRID ITEM-A, TAKODJE DEFINISATI JESTE I NJIHOV opacity

```
.item-1 {  
  
    background-color: #ff6f69;  
    opacity: 0.8;  
  
    grid-row: srednja / kraj;  
    grid-column: pocetak / kraj;  
}  
  
.item-2 {  
  
    background-color: #88d8b0;  
    opacity: 0.8;  
  
    grid-column: srednja / kraj;  
    grid-row: pocetak / kraj;  
}
```

I SADA CU IMATI SLEDECU SITUACIJU U POGLEDU GRID-A



KAO STO SE VIDI, JEDAN ELEMENT OVERLAP (PREKLAPA), DRUGI

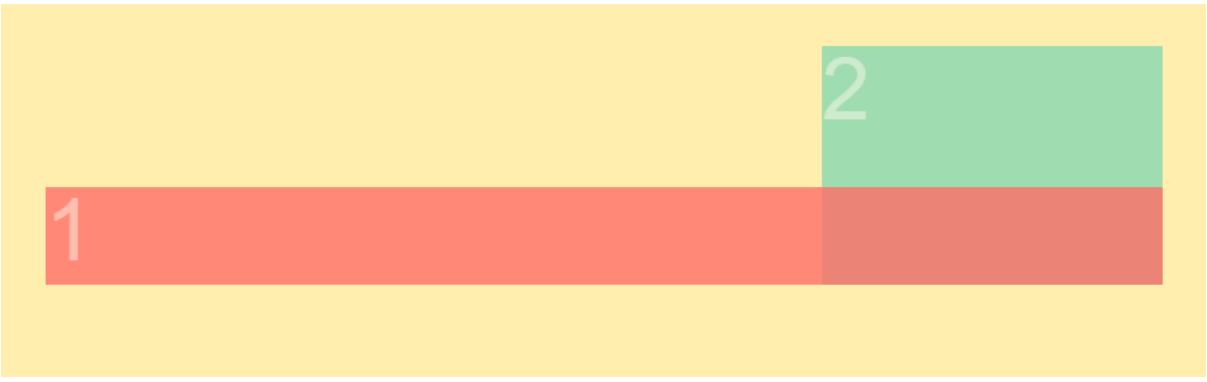
ONO STO JE BITNO SHVATITI JESTE, PO CEMU SE ODLUCUJE KOJI CE ELEMENT BITI PREKLOPLJEN, A
PO CEMU SE ODLUCUJE, KOJI CE ELEMENT PREKLAPATI DRUGI

ODGOVOR CE MI DATI HTML, JER CU URADITI SLEDECE

```
<div class="grid-container">  
    <div class="item-1">1</div>  
    <div class="item-2">2</div>  
</div>
```

```
<div class="grid-container">  
    <div class="item-2">2</div>  
    <div class="item-1">1</div>  
</div>
```

KAO STO SE VIDI SA SLIKE, ZAMENIO SAM MESTA DVEMA GRID ITEM-IMA



2

1

ONO STO MOGU ZAKLJUCITI JESTE DA CE BITI PREKLOPLJEN ONAJ GRID ITEM, KOJI SE U GRID CONTAINER-U (MISLIM NA HTML), NALAZI U REDU PRE, NEKOG DRUGOG GRID ITEM-A

auto-fit VS auto-fill

U OVOJ LEKCIJI CU SAZNATI, KOJA JE TO RAZLIKA IZMEDJU auto-fit VREDNOSTI, SA KOJOM SAM SE VEC SUSREO, I auto-fill VREDNOSTI, SA KOJOM SE PRVI PUT SUSRECEM (OBE VREDNOSTI SU ARGUMENTI repeat FUNKCIJE)

OPET CU KREIRATI PRIMER; I SET-UP CE IZGLEDATI OVAKO

```
html, body {  
    background-color: #ffeedad;  
    padding: 10px;  
}  
  
.container > div, .container2 > div {  
    font-size: 2em;  
    color: #ffeedad;  
  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.container > div:nth-child(1n), .container2 > div:nth-child(1n) {  
    background-color: #96ceb4;  
}  
  
.container > div:nth-child(3n), .container2 > div:nth-child(3n) {  
    background-color: #88d8b0;  
}  
  
.container > div:nth-child(2n), .container2 > div:nth-child(2n) {  
    background-color: #ff6f69;  
}  
  
.container > div:nth-child(4n), .container2 > div:nth-child(4n) {  
    background-color: #ffcc5c;  
}  
  
.title {  
    color: #ff6f69;  
    background-color: #ffeedad;  
    text-align: center;  
    padding: 10px 0;  
    margin: 0px;  
}
```

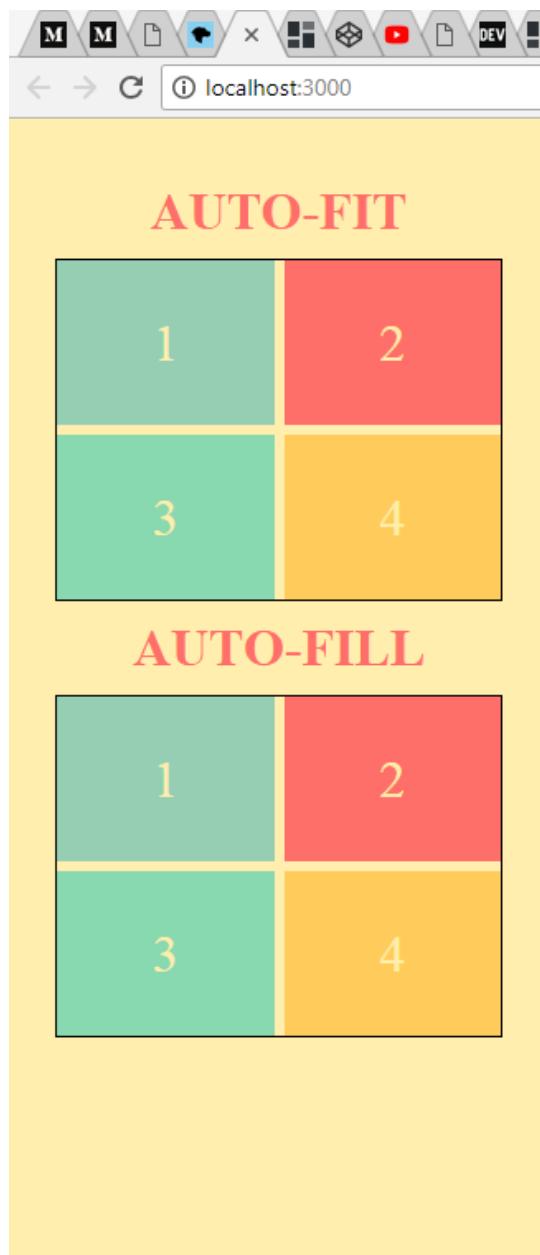
index.html

basic.css

```
.container {  
    display: grid;  
    grid-gap: 6px;  
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
    grid-template-rows: 100px 100px;  ( )  
  
    border: black solid 1px;  
}  
  
.container2 {  
    display: grid;  
    grid-gap: 6px;  
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));  
    grid-template-rows: 100px 100px;  ( )  
  
    border: black solid 1px;  
}
```

index.css

KAO STO VIDIM SA SLIKE GORE, OZNACIO SAM GDE SAM KORISTIO auto-fit, I GDE SAM KORISTIO auto-fill VREDNOST



(NAMERNO SAM NA POČETKU SKUPIO, BROWSEROW WINDOW)

INTERPRETIRACU SADA VREDNOST `grid-template-columns` PROPERTIJA

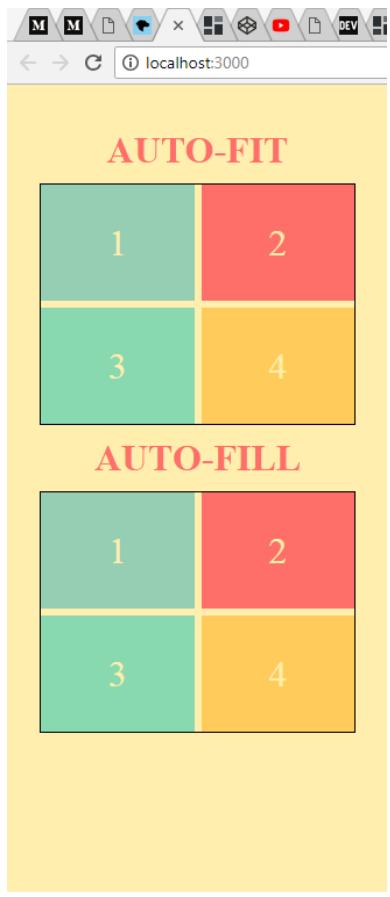
DAKLE, POSTO JE VREDNOST POMENUTOG PROPERTIJA, USTVARI `repeat` FUNKCIJA, A NJENI ARGUMENTI SU `auto-fit` I `minmax` FUNKCIJA;

U GRID-U CE SE UBACITI (FIT) STO JE VISE KOLONA MOGUCE, STO JE POSLEDICA `auto-fit` VREDNOSTI

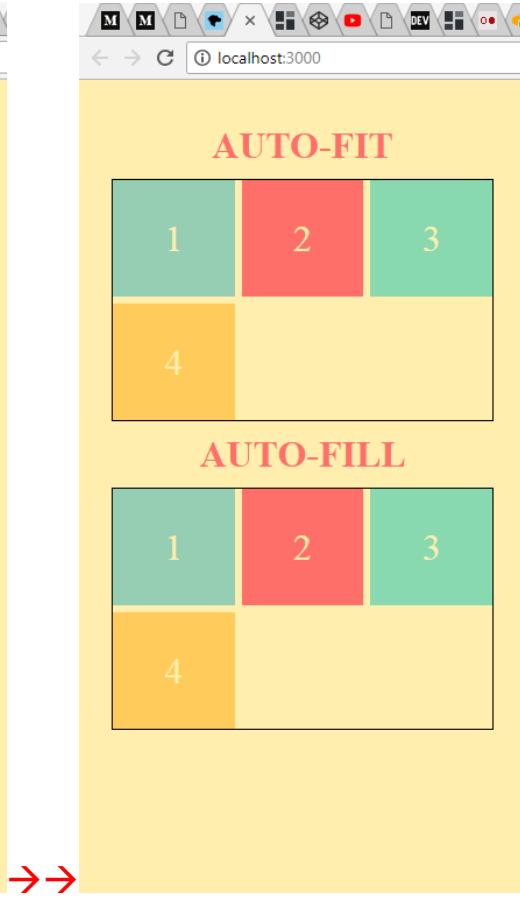
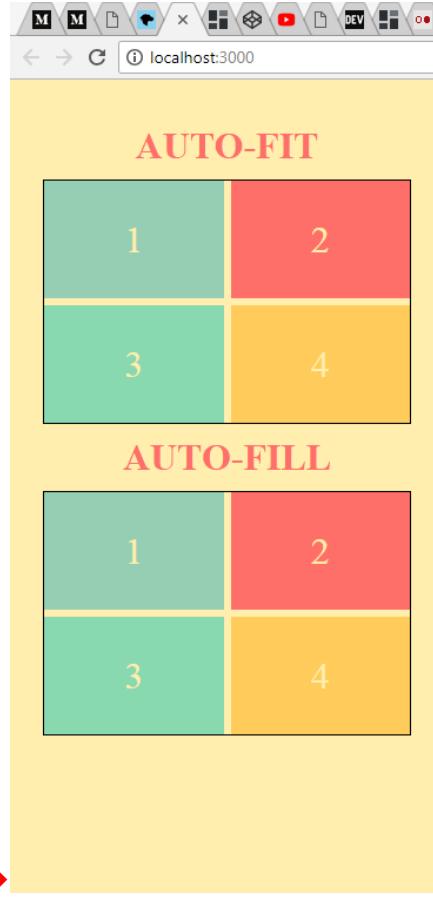
ALI, UZ OGRANICENJE DA NAJMANJA MOGUCA SIRINA GRID ITEM-A, JESTE 100px, A NJAVECA JESTE 1fr

IVO DAJE LEPO RESPONSIVE PONASANJE

SADA CU POVECAVATI VELICINU BROWSER-OVOG WINDOW-A, KAKO BIH VIDEO STA CE SE TO DESAVATI



→→



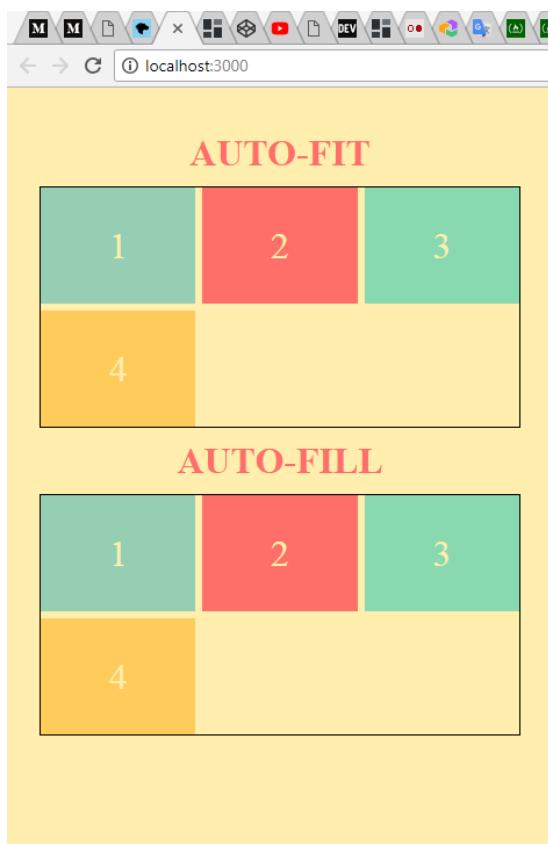
→→

KAO STO VIDIM, BROJ KOLONA RASTE, SA POVECAVANJEM GRID-A

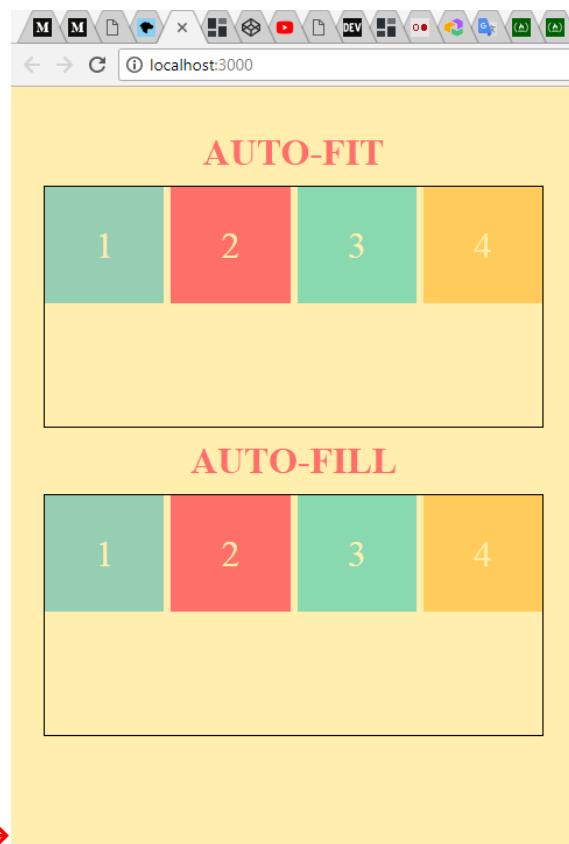
SA SLIKE GORE MOGU VIDETI DA JE GRID, SIRECI SE, PRESAO TRESHOLD (PRAG) OD 300px, I DA SADA MOGU STATI TRI KOLONE

AKO OPET POCNEM DA SIRIM GRID, KOLONE CE OPET POSTATI SIROKE JEDNU FRACTION JEDINICU

I SIRECI GA I DALJE, PRECI CE I TRESHOLD OD 400px, I ONDA CE SE GRID PROSTIRATI U 4 KOLONE

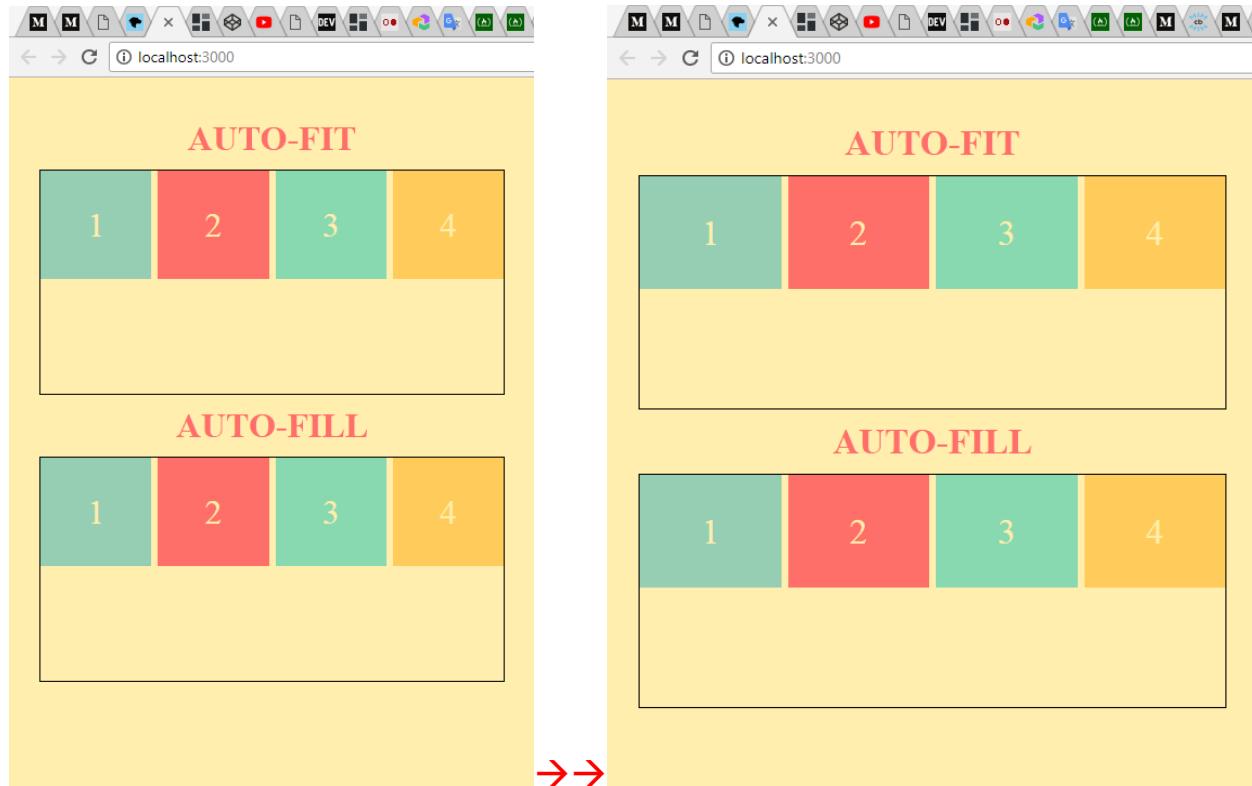


→→→→

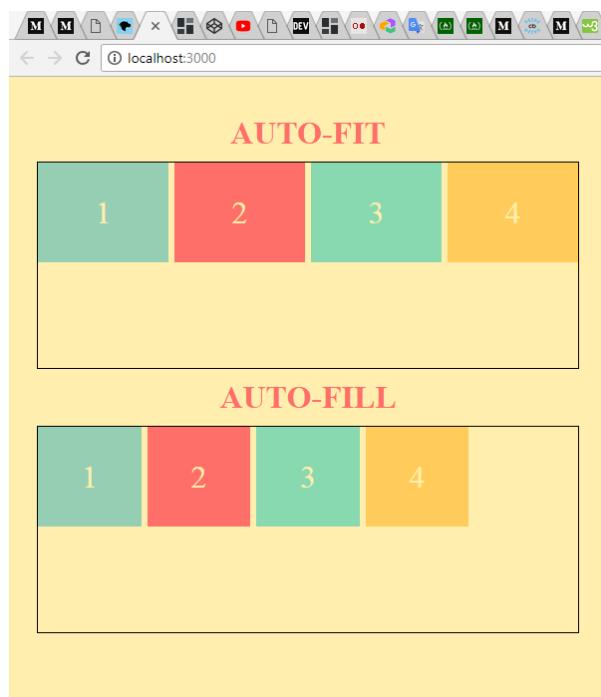


OBRATI PAZNJU DA SE ISTO DOGADJA U OBA GRIDA, I ONOG CIJE SU KOLEONE DEFINISANE UZ POMOC auto-fit VREDNOSTI, ALI I ONOG CIJE SU KOLONE DEFINISANE UZ POMOC auto-fill VREDNOSTI

U CEMU JE RAZLIKA IZMEDJU POMENUTE DVE VREDNOSTI, MOGU VIDETI, AKO POKUSAM, CAK JOS DA POVECAM, OBA GRID-A, MOG PRIMER-A



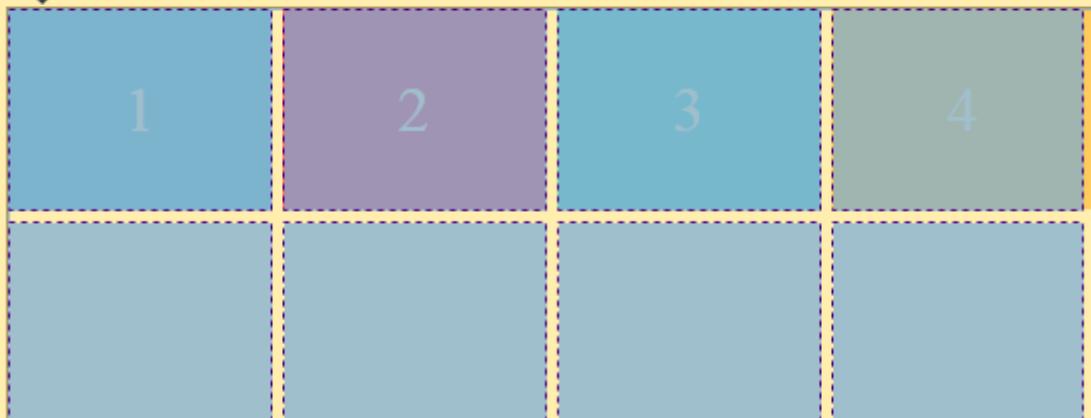
NAIME NAKON STO, OBA GRID-A, U POGLEDU SIRINE, PREDJE TRESHOLD OD 500px, DESICE SE SLEDECE



KAO STO VIDIM, GRID ITEMI PRVOG GRIDA SE I DALJE SIRE SA POVECAVANJEM SIRINE GRIDA, DOK SU GRID ITEMI DRUGOG GRIDA PRESTALI DA RASTU; UMESTO TOGA, DODATA JE NOVA KOLONA, DRUGOM GRIDU, STO LAKO MOGU VIDETI U Element SEKCIJI, CHROME DEVELOPER TOOLS-A

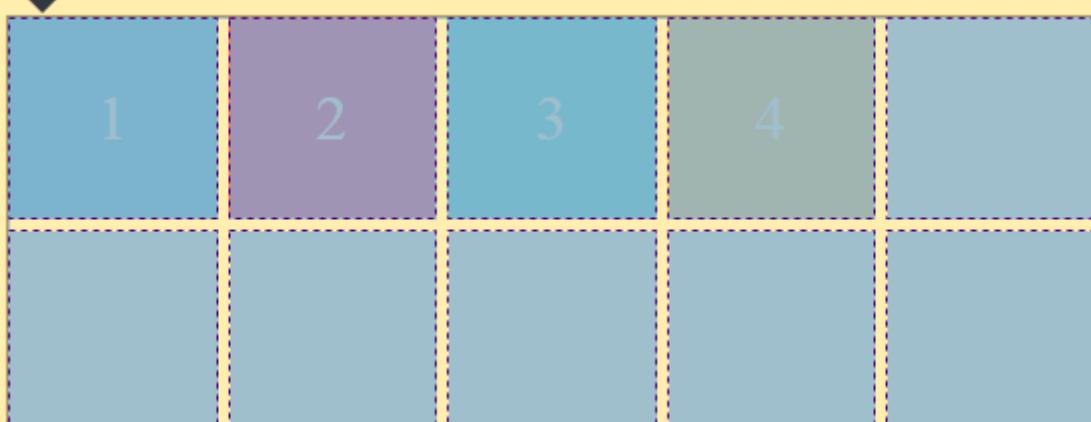
`div.container | 544 × 208`

AUTO-FIT



`div.container2 | 544 × 208`

AUTO-FILL



DAKLE, DRUGOM GRIDU JE DODATA PETA KOLONA, IAKO GRID NE OBUHVATA 5 GRID ITEMA (OBUHVATA 4)

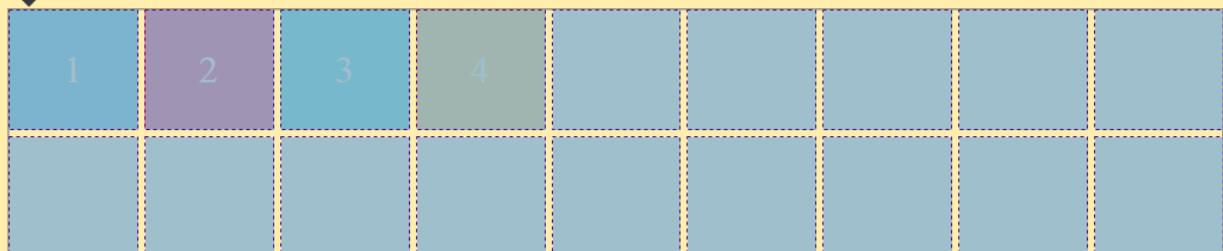
U SLUCAJU PRVOG GRID-A, auto-fit SE POSTARAO, DA 4 GRID ITEMA STAJU U CEO GRID; MEDJUTIM I U NJEGOVOM SLUCAJU DOSLO JE DO KREIRANJA NOVE KOLONE, MEDJUTIM TA KOLONA JE SRUSENA (COLAPSED)

DAKLE, auto-fit, DALJIM SIRENJEM GRID-A, BI RUSIO, SVE NOVE KOLONE KOJE SU KREIRANE; KAKO BI SVAKI OD GRID ITEMA BIO SIROK PO JEDNU FRACTION JEDINICU, JER SAM JA TAKO DEFINISAO

A U SLUCAJU auto-fill VREDNOSTI, SIRENJEM GRID-A, POMENUTE, NOVE KOLONE, NECE BITI RUSENE, VEC SVAKIM PRELASKOM, NOVOG TRESHOLDA (U OVOM SLUCAJU + 100PX), NOVA KOLONA CE BITI DODATA GRID-U, I NECE BITI COLAPSED

`div.container2 | 1021 × 208`

AUTO-FILL



KREIRANJE ARTICLE LAYOUT-A UZ POMOC CSS GRID-A

HTML OVOG PRIMERA SE SASTOJI OD article TAGA, KOJI JE NESTED U body ELEMENT

DAKLE REC JE O article TAGU, KOJI CE BITI STILIZOVAN NA SVOJ NACIN, I KOJI NECU DODATNO KOMENTARISATI, JER TO ISKACE IZ TEME OVOG TUTORIJALA

NECU ZATO PRIKAZIVATI SET-UP (CODE HTML I CSS FAJLOVA)

JEDINO STO CU RECI ZA POCETAK JESTE DA OVAJ ARTICLE, ZA SADA NIJE GRID CONTAINER

What is involved in supporting NPM?

There are two very distinct steps in supporting NPM: Dependency resolution and module loading.

Dependency resolution is about going from a dependency requirement,

```
1 "dependencies": {  
  2   "react": "^16.0.0",  
  3   "react-dom": "^16.0.0"  
  4 }
```

to a concrete description of what packages needs to be installed:

```
node_modules/react      react@16.0.0  
node_modules/react-dom  react-dom@16.0.0  
node_modules/fbjs       fbjs@0.8.16  
...
```

Module loading is more involved and includes:

- Figuring out that `require("react")` in app.js refers to node_modules/react/index.js
- Making sure dependencies are available before the module body is being executed
- Providing `exports`, `module`, `process` variables inside the module body

Dependency resolution

[Yarn](#) is a package manager for NPM which you can use a library. There is not a public API available at the moment, but as long as you depend on a fixed version everything works out nicely. Yarn has a `PackageResolver` which resolves all dependencies, and with the `PackageHoister` you can find the paths where packages needs to be installed. Yarn is decently decoupled and although it lacks documentation about the internal classes it's quite easy to figure things out.

Module loader: Webpack

There are some online coding sandboxes ([Webpackbin](#) and [CodeSandbox](#)) which supports NPM packages through server-side Webpack. Their approach is essentially as follows:

First they have a `packager` which takes a list of fully resolved packages (e.g. `react@16.0.0,react-dom@16.0.0`), generates a `package.json`, install packages and dependencies with `yarn`, and then uses Webpack to build a `DLL package`. A DLL package is a way of bundling all files into a single Webpack bundle. Packaging can take a while (>10s), but it's only needed once per request and you can cache the produced package indefinitely.

When you want to execute your `app.js` you send it to the server, generate a Webpack configuration (which includes a reference to the DLL package) and Webpack will resolve everything for you.

There are two big disadvantages with this approach:



GORE SAM PRIKAZAO NEKE DELOVE ARTICLE-A; A SAMO CU DODATI DA SE CEO PRIMER ZA OVU LEKCIJU NALAZI NA SLEDECOJ WEB STRANICI

<https://scrimba.com/p/pWqLHa/cdp76sD>

SADA CU DEFINISATI DA POMENUTI CLANAK BUDE I GRID CONTAINER, KOJI CE SE PROSTIRATI U TRI KOLONE, OD KOJIH CE SREDNJA ZAUZIMATI NAJVISE MESTA

A DEFINISACU I DA SVI GRID ITEM-I, BUDU U SREDNJOJ KOLONI, JER ZA SADA NE ZELIM DA SE IKAKVI ELEMENTI PROSTIRU, PO PRVOJ I TRECOJ KOLONI

```
article {  
  display: grid;  
  grid-column-gap: 10px; /*grid-gap JE TAKODJE BIO SHORTHAND (VEOMA VAZNO, jer to nisam znao ranije)*/  
  grid-template-columns: 80px 1fr 80px;  
}  
  
article > * {  
  grid-column: 2; /*grid-column-start*/ /*svaki grid item zauzima po jednu grid celiju*/  
  min-width: 0px; /*zbog responsiveness-a*/  
  ... ... ... ... /*elementi ce se smanjivati po sirini dokle god to bilo moguce*/  
}
```



Running any NPM pacakge in the browser locally

JavaScript has never had any official solution for distributing packages, and every web platform (Rails, Django etc) has their own idea of how to structure and package JavaScript. In the last few years NPM has started becoming the canonical way of distribution, with Webpack as the build system, but there's no way to load NPM packages in the browser without a server-side component.

Scrimba is a platform for interactive coding screencast where you can run the code at any moment in time. Being able to use NPM packages has been a frequently requested feature, but it turns out to be surprisingly technically difficult to implement. In this article I will explain how we tackled the problem and the final solution we ended up with.



Credits: Wesley Burke

Our solution (inspired by StackBlitz) runs almost entirely on the client-side. Dependency resolution is done server-side, but fetching NPM files, parsing `require()`, resolving require paths and creating a bundle is all done at the client. This (combined with IndexedDB caching) gives a smooth experience with very low latency.

U SUSTINI ONO STO JE DEFINISANO U OVOM SLUCAJU JESTE DA OVAJ CLANAK IMA PRAZNA MESTA, SA STRANA

MEDJUTIM, ONO STO ZELIM DA DEFINISEM JESTE DA SE SLIKA (LOADOVANA, ODNOŠNO PREDSTAVLJENA PUTEM figure TAGA), PROSTICE CELOM SIRINOM STRANICE, A DODACU JOJ IZNAD, I MALO PROSTORA



Running any NPM pacakge in the browser locally

JavaScript has never had any official solution for distributing packages, and every web platform (Rails, Django etc) has their own idea of how to structure and package JavaScript. In the last few years NPM has started becoming the canonical way of distribution, with Webpack as the build system, but there's no way to load NPM packages in the browser without a server-side component.

Scrimba is a platform for interactive coding screencast where you can run the code at any moment in time. Being able to use NPM packages has been a frequently requested feature, but it turns out to be surprisingly technically difficult to implement. In this article I will explain how we tackled the problem and the final solution we ended up with.



Credits: Wesley Burke

```
article > figure {  
  grid-column: 1 / -1;  
  margin: 20px 0;  
}
```

KAO STO VIDIM, SADA SE SLIKA PROSTIRE CELOM SIRINOM STRANICE (I NJEN CAPTION, (credits: bla bla) KOJI JE DEFINISAN figurecaption TAGOM)

OVO JE RADIKALNO JEDNOSTAVNO, NEGO NACINIMA, KOJI SU BILI ZASTUPLJENI, KADA CSS GRID NIJE POSTOJAO

BEZ UPOTREBE GRID-A, MORAO BI ZA SVE OSTALE EMENTE DEFINISATI MARGINU, A ZA SLIKU NE U TOM SLUCAJU BI MORAO IMATI HTML KOJI BI BIO PREVISE CUSTOM

SADA CU DODATI NOVI HTML PARAGRAF ELEMENT, ZA KOJI CU DEFINISATI I class ATRIBUT SA VREDNOSCU "aside"

```
<p class="aside">  
  Yarn 1.3 was recently released and includes a bunch of new features!  
</p>
```

Dependency resolution

[Yarn](#) is a package manager for NPM which you can use a library. There is not a public API available at the moment, but as long as you depend on a fixed version everything works out nicely. Yarn has a `PackageResolver` which resolves all dependencies, and with the `PackageHoister` you can find the paths where packages needs to be installed. Yarn is decently decoupled and although it lacks documentation about the internal classes it's quite easy to figure things out.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Webpack

There are some online coding sandboxes ([Webpackbin](#) and [CodeSandbox](#)) which supports NPM packages through server-side Webpack. Their approach is essentially as follows:

ALI, POSTO JE TO ASIDE ELEMENT, JA ZELIM DA SE ON NALAZI SA STRANE I DA IMA NESTO DRUGACIJI STIL

```
article > .aside {  
  
  grid-column: 3;  
  color: #668;  
  font-size: 0.8em;  
  
}
```

POGLEDACU SADA Elements SEKCIJU, CHROME TOOLSA

Dependency resolution	
<p>Yarn is a package manager for NPM which you can use a library. There is not a public API available at the moment, but as long as you depend on a fixed version everything works out nicely. Yarn has a <code>PackageResolver</code> which resolves all dependencies, and with the <code>PackageHoister</code> you can find the paths where packages needs to be installed. Yarn is decently decoupled and although it lacks documentation about the internal classes it's quite easy to figure things out.</p>	Yarn 1.3 was recently released and includes a bunch of new features!
Module loader: Webpack	
<p>There are some online coding sandboxes (Webpackbin and CodeSandbox) which supports NPM packages through server-side Webpack. Their approach is essentially as follows:</p>	

```
article > .aside {  
    grid-column: 1;  
    grid-row: 1;  
    color: #668;  
    font-size: 0.8em;  
}
```

Dependency resolution

[Yarn](#) is a package manager for NPM which you can use a library. There is not a public API available at the moment, but as long as you depend on a fixed version everything works out nicely. Yarn has a `PackageResolver` which resolves all dependencies, and with the `PackageHoister` you can find the paths where packages needs to be installed. Yarn is decently decoupled and although it lacks documentation about the internal classes it's quite easy to figure things out.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Webpack

There are some online coding sandboxes ([Webpackbin](#) and [CodeSandbox](#)) which supports NPM packages through server-side Webpack. Their approach is essentially as follows:

(ONO STO JE VAZNO U OVOM SLUCAJU, ATO TREBA I DA UPAMTIS, JESTE DA JE PRILIKOM OVAKVOG POZICIONIRANJA BIO BITAN REDOSLED ELEMENATA U HTML-U)

MEDJUTIM, IPAK JE BOLJE DA BUDE U TRECOJ, KAKO SAM TO RANIJE PODESIO

```
article > .aside {  
    grid-column: 3;  
    color: #668;  
    font-size: 0.8em;  
}
```

ALI SADA CU `.aside` ELEMENT, PREMESTITI (MISLIM NA PREMESTANJE U HTML-U), KAKO BI SE POMENUTI NASAO U NEKOM DRUGOM REDU, ALI GLEDACU DA SUSEDNA, ODNOSN OSREDNJA CELIJA IMA TEKST, KOJEG NEMA, BAS MNOGO; TO RADIM JER ZELIM DA VIDIM, KOJA CE SADRZINA U TOM SLUCAJU DIKTIRATI VISINU REDA

```
<p>We created a prototype together with Christian Alfonni (the creator of Webpackbin), but had some pending work when we discovered a different approach.</p>  


Yarn 1.3 was recently released and includes a bunch of new features!


```

KAO STO VIDIM SA SLIKE GORE, NASAO SAM NOVO MESTO `.aside`-U

DAKLE, NAKON NEKOG DRUGOG PARAGRAFA

- Building the initial package can take a *long* time. This needs to be done on the server-side and its scalability is limited by how many servers you have.
- Packages can only be cached as a whole. This means that if you add one small dependency you must still create a whole new package.

We created a prototype together with Christian Alfoni (the creator of Webpackbin), but had some pending work when we discovered a different approach.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Unpkg + SystemJS

[StackBlitz](#) is an online coding sandbox which supports NPM without using Webpack at all. They briefly described their approach in a [comment at the unpkg-repo](#):

DAKLE, STVORILA SE MOZDA PROBLEMATICA SITUACIJA, JER JE U OVOM SLUCAJU TRECA CELIJA, ZATO STO IMA NAJVISE SADRZINE, DIKTIRALA VISINU REDA, PA SE NALAZI VECI NEPOPUNJENI PROSTOR, IZMEDJU PARAGRAFA U SREDNJOJ CELIJI I HEADER-A ISPOD NJEGA

ONO STO MOGU URADITI U TOM SLUCAJU JESTE PODESAVANJE VISINE, SAMOG .aside ELEMENTA, NA VREDNOST NULA

TADA BI ONAJ HTML ELEMENT, CIJA VISINA NIJE NULA, DIKTIRAO VISINU VISINU REDA U KOJEM JE SMESTEN

```
article > .aside {
    grid-column: 3;
    grid-row: ;
    color: #668;
    font-size: 0.8em;
    height: 0;
}
```

We created a prototype together with Christian Alfoni (the creator of Webpackbin), but had some pending work when we discovered a different approach.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Unpkg + SystemJS

[StackBlitz](#) is an online coding sandbox which supports NPM without using Webpack at all. They briefly described their approach in a [comment at the unpkg-repo](#):

KAO STO VIDIM SA SLIKE GORE, ELEMENT STILIZOVAN KLASOM .aside, NIJE MOGAO DA KOLAPSIRA U POTPUNOSTI (ZBOG NULA VISINE), VEC JE VISINA CELIJE REDA, KOJA JE DO NJEGA ODLUCILA I NJEGOVU VISINU, SAMO STO SE DOGODIO OVERFLOW IZ .aside-OVE CELIJE

ONO CIME SE JOS MOGU POZABAVITI JESTE blockquote ELEMENT, KOJI SE TAKODJE NALAZI U HTML-U, OVOG PRIMERA

```
<blockquote>
| <p>In short, SystemJS & Unpkg are an incredible duo for dev UX because they reflect what makes local dev environments great: the client is doing all of the downloading, installing, bundling, and even serving the application.</p>
</blockquote>
```

We created a prototype together with Christian Alfoni (the creator of Webpackbin), but had some pending work when we discovered a different approach.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Unpkg + SystemJS

[StackBlitz](#) is an online coding sandbox which supports NPM without using Webpack at all. They briefly described their approach in a [comment at the unpkg-repo](#):

In short, SystemJS & Unpkg are an incredible duo for dev UX because they reflect what makes local dev environments great: the client is doing all of the downloading, installing, bundling, and even serving the application.

They haven't open-sourced their solution yet, but from their description we can still go through how it works. Let's say we have an app.js which contains the following:

SADA CU STILIZOVATI, POMENUTI ELEMENT, KAKO BI SE PROSTIRAO CELOM SIRINOM GRID-A, KAKO BI IMAO LEVI BORDER, I JOS NEKE STILOVE

```
article > blockquote {
  grid-column: 1 / -1;
  border-left: #eee solid 10px;
  padding-left: 10px;
  margin: 10px 0;
  color: #668;
}
```

We created a prototype together with Christian Alfoni (the creator of Webpackbin), but had some pending work when we discovered a different approach.

Yarn 1.3 was recently released and includes a bunch of new features!

Module loader: Unpkg + SystemJS

[StackBlitz](#) is an online coding sandbox which supports NPM without using Webpack at all. They briefly described their approach in a [comment at the unpkg-repo](#):

In short, SystemJS & Unpkg are an incredible duo for dev UX because they reflect what makes local dev environments great: the client is doing all of the downloading, installing, bundling, and even serving the application.

They haven't open-sourced their solution yet, but from their description we can still go through how it works. Let's say we have an app.js which contains the following:

I OVO BI BILO TESKO POSTICI, BEZ UPOTREBE GRID-A

GRID VS FLEXBOX

U OVOJ LEKCIJI, BAVICU SE RAZLIKAMA IZMEDJU FLEXBOX-A I GRID-A, A TAKODJE CU SAZNATI, KAKO MOGU DA IH KORISTIM ZAJEDNO

ZATO CU KREIRATI HEADER, KOJI CE KORISTITI FLEXBOX, I KREIRACU PAGE LAYOUT, POMOCU CSS GRID-A

OVAKO CE IZGLEDATI HTML MOG PRIMERA

```
<p>FLEXBOX HEADER</p>
<div class="flexbox-header">
  <div>HOME</div>
  <div>SEARCH</div>
  <div>LOGOUT</div>
</div>
<p>GRID PAGE</p>
<div class="grid-page">
  <div class="header">HEADER</div>
  <div class="menu">MENU</div>
  <div class="content">CONTENT</div>
  <div class="Footer">FOOTER</div>
</div>
```

U basic.css FAJLU, DO ODREDJENOG NIVOA CU STILIZOVATI HTML ELEMENTE, SA GORNJE SLIKE

```
html, body {
  background-color: #ffead;
  margin: 10px;
  font-size: 20px;
}

p {
  color: #96ceb4;
  text-align: center;
}

.flexbox-header {
  background-color: #96ceb4;
  color: #ffead;
}

.flexbox-header > div {
  display: flex;
  font-size: 1em;
  padding: 10px 20px;
}

.grid-page > div {
  color: #ffead;
  padding: 0px 20px;
  display: flex;
  align-items: center;
}

.header {
  background-color: #96ceb4;
}

.menu {
  background-color: #ff6f69;
}

.content {
  background-color: #ffcc5c;
}

.footer {
  background-color: #88d8b0;
}

.header :nth-child(2) {
  padding: 0 48px;
}
```

ZATIM CU DEFINISATI DA ELEMENT, KOJI IMA class ATRIBUT SA VREDNOSCU "flexbox-container", U PRAVO I POSTANE FLEX CONTAINER

A TAKODJE CU DEFINISATI DA ELEMENT, KOJI IMA class ATRIBUT SA VREDNOSCU "grid-page", POSTANE GRID CONTAINER

TO CU SVE DEFINISATI U index.css FAJLU

A TAKODJE CU DEFINISATI I PROSTIRANJE, POJEDINACNIH GRID ITEM-A, PO GRID-U

A STO SE TICE FLEX CONTAINER-A, DEFINISACU DA JEDAN OD NJEGOVIH FLEX ITEMA, IMA AUTOMATSku VREDNOST ZA LEVU MARGINU; TO CE NAIME BITI TRECI FLEX ITEM, CIJA CE LEVA MARGINA ZAUZETI SAV PRAZAN PROSTOR, CIME CE TAJ TRECI FLEX ITEM BITI SMESTEN UZ DESNU IVICU FLEX CONTAINER-A

```
.flexbox-header {  
    display: flex;  
}  
  
.flexbox-header > div:nth-child(3) {  
    margin-left: auto;  
}  
  
  
.grid-page {  
    display: grid;  
    grid-template-columns: repeat(12, 1fr);  
    grid-template-rows: 48px 206px 48px;  
}  
  
.header {  
    grid-column: 1 / -1;  
}  
  
.menu {  
    grid-column: 1 / 2;  
}  
  
.content {  
    grid-column: 2 / -1;  
}  
  
.footer {  
    grid-column: 1 / -1;  
}
```

A OVAKO SE SVE IZGLEDATI



localhost:3000

FLEXBOX HEADER

HOME

SEARCH

LOGOUT

GRID PAGE

HEADER

MENU

CONTENT

FOOTER

NA SLICI GORE, MOGU PRIMETITI HEADER, KOJI JE IZGRADJEN UZ POMOC FLEXBOX-A, I PAGE LAYOUT, IZGRADJEN UZ POMOC CSS GRID-A

POSMATRACU SADA POMENUTA DVA PRIMERA, JER MOZDA SE MOZE PRIMETITI RAZLIKA IZMEDJU FLEXBOX-A I GRIDA, POSMATRANJEM, DVA POMENUTA PRIMERA

NAIME, FLEXBOX SLUZI ZA IZGRADNNU JEDNODIMENZIONIH LAYOUT-A; KAO STO VIDIM FLEXBOX HEADER, IMA JEDAN PRAVAC, U SMERU OD LEVA NA DESNO, ODNOSNO U PITANJU JE SAMO JEDAN RED (SINGLE ROW)

DOK JE CSS GRID SLUZI ZA IZGRADNNU DVODIMENZIONALNIH LAYOUT-A; KAO STO JE PAGE LAYOUT SA SLIKE GORE, KOJI IMA TRI REDA, I IMA 12 KOLONA, S TIM STO SE VIZUELNO MOGU VIDETI DVE (TAMO GDE SE SPAN-UJU MNU I CONTENT), USLED SPECIFICNOG POLAGANJA (LAYING OUT) GRID ITEM-A

AKO ZELIM DA KREIRAM LAYOUT, KOJI IMA PRAVAC, ODNOSNO ILI KOLONU ILI RED, ONDA TREBAM KORISTITI FLEXBOX, KOJI JE SJAJAN ZA LINING (PORAVNAVANJA ILI REDANJA) CONTENT-A, U CONTAINER-U, JER PRUZA MNOGO FLEKSIBILNOSTI

ALI, AKO ZELIM DA KREIRAM, MORE OVERALL LAYOUT (VISE UOPSTEN LAYOUT), KOJI IMA REDOVE I KOLONE, ONDA BI TREBAO KORISTITI CSS GRID, JER CE MI ON PRUZATI, MNOGO VISE FLEKSIBILNOSTI, I JEDNOSTAVNIJI MARKUP (JEDNOSTAVNIJI HTML), NEGO KADA BI KREAIRA DVODIMENZIONALNI LAYOUT UZ POMOC FLEXBOX-A

ALI NAIME, POSTOJI JOS JEDNA OPSTA RAZLICA, IZMEDJU FLEXBOX-A I GRID-A

ONA SE OGLEDA U SLEDECDEM:

FLEXBOX TAKES BASIS IN THE CONTENT, ODNOSNO FLEXBOX UZIMA OSNOVU U SADRZINI (CONTENT-U) ; TO SE MOZE ZAPAMTITI, PREKO, TAKOZVANE ODREDNICE “CONTENT FIRST”

DOK GRID MOGU POSMATRATI KROZ SLEDECU ODREDNICU: “LAYOUT FIRST”, ODNOSNO GROD TAKES BASIS IN THE LAYOUT

POMENUTO MOGU SHVATITI GLEDAJUCI CSS MOG PRIMERA

FLEX CONTAINER SE KREIRA DODAVAJUCI, SLEDECI PROPERTI I SLEDECU VREDNOST display: flex, NEKOM ELEMENTU

A VIDECU DA SAM TRECEM FLEX ITEMU DODELIO LEVU MARGINU, SA VREDNOSCU auto, ODNOSNO margin: auto, CIME JE TRECI FLEX ITEM PRILEPLJEN ZA DESNU STRANU FLEX CONTAINER-A

DAKLE SADRZINA “JE ODLUCILA”, ODNOSNO FLEX ITEM

MEDJUTIM, GRID SE DEFINISE DRUGACIJE, PORED display: grid, PROPERTIJA I VREDNOSTI , JA MORAM DEFINISATI REDOVE I/ILI KOLONE, DAKLE “LAYOUT FIRST”

“CONTENT FIRST” I “LAYOUT FIRST” SU ABSTRAKCIJE, MEDJUTIM DOBRO JE POZNAVATI POMENUTE ODREDNICE

SADA CU VIDETI, KAKO MOGU KOMBINOVATI GRID I FLEXBOX; AND THEN GET BEST OF BOTH WORLDS

ONO STO CU URADITI JESTE KOPIRANJE I PROSLEDJIVANJE, ODNOSNO NEST-OVANJE, TRIJU FLEX ITEM-A, U HEADER GRID ITEM

```
<p>FLEXBOX HEADER</p>
<div class="flexbox-header">
  <div>HOME</div>
  <div>SEARCH</div>
  <div>LOGOUT</div>
</div>
<p>GRID PAGE</p>
<div class="grid-page">
  <div class="header">
    <div>HOME</div>
    <div>SEARCH</div>
    <div>LOGOUT</div>
  </div>
  <div class="menu">MENU</div>
  <div class="content">CONTENT</div>
  <div class="footer">FOOTER</div>
</div>
```

AKO SE PODSETIM, VEC SAM RANIJE KREIRAO DA JE HEADER, USTVARI FLEX CONTAINER (A TAKODJE SU SVI OSTALI GRID ITEM-I, TAKODJE I FLEX CONTAINERI)

```
.grid-page > div {
  color: #ffeedad;
  padding: 0px 20px;

  display: flex;
  align-items: center;
}
```

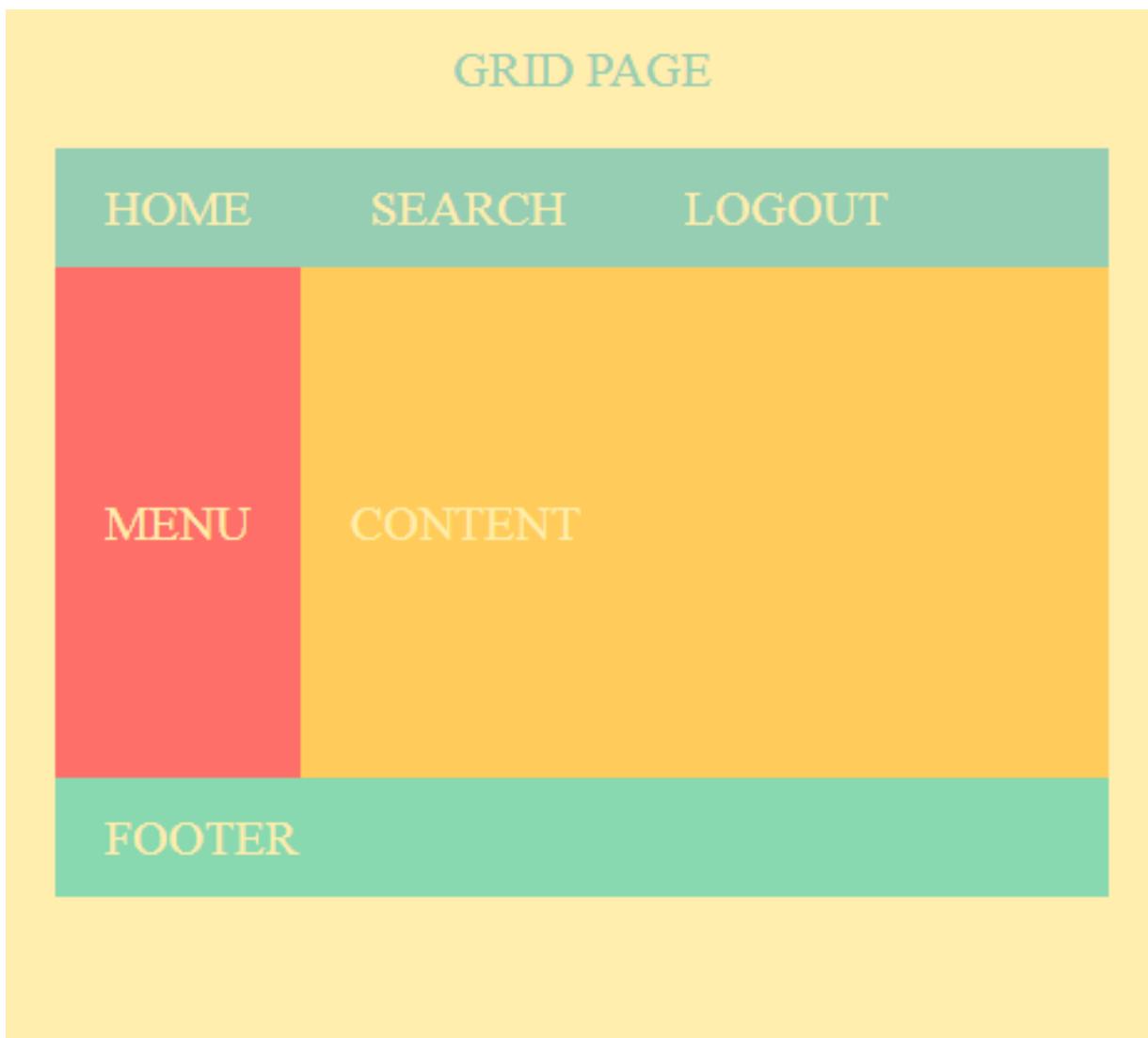
A TAKODJE SAM, KONKRETNO HEADEROV-OM DRUGOM CHILD ELEMENTU, DODAO PADDING RANIJE, IAKO TAJ ELEMENT NIJE JOS POSTOJAO

```
.header :nth-child(2) {  
    padding: 0 48px;  
}
```

(****NE ZNAM ZASTO SU OVO U RADILI U TUTORIJALIMA, JER MI IZGLEDA NEBITNO U ODНОСУ NA TEMU OVE LEKCIJE, CAK MOZDA I BESPOTREBNO ****)

KAO STO VIDIM PADDING, POMENUTOG FLEX ITEM-A, SA LEVE I DESNE JESTE 48 PIKSELA, DOK JE GORE I DOLE, USTVARI NULA

ZA SADA HEADER ELEMENT GRID-A, IZGLEDA OVAKO



JA SAD MOGU DEFINISATI JUSTIFICATION I ALIGNMENT FLEX ITEMA, UZ POMOC VEC POMINJANIH PROPERTIJA, ALI JA CU, IPAK DEFINISATI, DA LEVA MARGINA TRECEG FLEX ITEMA, BUDE auto

```
.header > div:nth-child(3) {  
    margin-left: auto;  
}
```



← → C

localhost:3000

FLEXBOX HEADER

HOME

SEARCH

LOGOUT

GRID PAGE

HOME

SEARCH

LOGOUT

MENU

CONTENT

FOOTER

```
*****  
*****  
*****  
*****
```

---- CSS VARIABLES ----

PRE NEGO STO SE POCNEM BAVITI CSS VARIJABLAMA, KREIRACU PRIMER KOJIC U KORISTITI, TOKOM OVIH TUTORIJALA, VEZANIH ZA CSS VARIJABLE

```
<header>
  <nav id="navbar">
    <ul>
      <li><a href="#">home</a></li>
      <li><a href="#">about</a></li>
      <li><a href="#">contacts</a></li>
    </ul>
  </nav>
</header>
<main>
  <h1>my awesome grid portfolio</h1>
  <div class="grid">
    <div class="item">
      <h1>project a</h1>
      <button>learn more</button>
    </div>
    <div class="item">
      <h1>project b</h1>
      <button>learn more</button>
    </div>
    <div class="item">
      <h1>project c</h1>
      <button>learn more</button>
    </div>
    <div class="item done">
      <h1>project d</h1>
      <button>learn more</button>
    </div>
  </div>
</main>
<footer><p>Made with I in Norway</p></footer>
```

(NARAVNO header, main I footer SU NESTED U body-JU)

index.html

```
html, body {
  background: #ffeedd;
  color: #ff6f69;
}

#navbar a {
  color: #ff6f69;
}

.item {
  background: #ffcc5c;
}

button {
  background: #ff6f69;
  color: #ffcc5c;
}
```

index.css

```
body, html {  
    margin: 0;  
    padding: 0;  
}  
  
a {  
    text-decoration: none;  
}  
  
h1 {  
    text-align: center;  
    font-size: 36px;  
}  
  
#navbar {  
    padding: 20px 0;  
    font-size: 24px;  
}  
  
  
.grid {  
    display: grid;  
    grid-template-columns: 200px 200px;  
    grid-auto-rows: 140px;  
    grid-gap: 20px;  
    justify-content: center;  
}  
  
a {  
    font-size: 28px;  
}  
  
#navbar > ul {  
    display: flex;  
    justify-content: space-around;  
    margin: 0 auto;  
    padding: 0;  
    list-style-type: none;  
}
```

```
p {  
    text-align: center;  
}  
  
h1 {  
    font-size: 36px;  
}  
  
p {  
    font-size: 18px;  
}  
  
  
footer {  
    position: absolute;  
    bottom: 0;  
    width: 100%;  
}  
  
footer > p {  
    font-size: 12px;  
}  
  
button {  
    margin: 0px auto;  
    display: block;  
    padding: 5px 10px;  
    border: none;  
}
```

basic.css

POSTO SAM NAPISAO HTML I CSS CODE SA GORNJIH SLIKA, I NAKO NSTO SAM GA MALO PROUCIO,
POZBAVICU SE SLEDECIM PITANJEM:

ZASTO NAUCITI CSS VARIJABLE?

ZATO STO SE BEZ KORISCENJA CSS VARIJABLI, MOGU NACI U OVAKVOJ SITUACIJI

```
title {  
    color: #ff6f69;  
}  
  
/*GLAVNA CRVENA BOJA, MOJA APLIKACIJE*/  
  
.quote {  
    color: #ff6f69;  
}
```

NAIME, KADA JE POTREBNO KORISTITI, NPR. BOJU NA VISE MESTA U APLIKACIJI, STARIM NACINOM
SE MORALA REUSE-OVATI, HEKSADECIMALNA VREDNOST, KAO NA SLICI GORE

A NOVIM NACINOM, ODNOSNO SA CSS VARIJABLAMA, U MOGUCNOSTI SAM DA, PRVO DEKLARISEM VARIJABLU, PA DA ONDA TU VARIJABLU, KORISTIM, TAMO GDE ZELIM

```
:root {  
    --crvena: #ff6f69; /*DEKLARISANJE VARIJABLE*/  
}  
  
title {  
    color: var(--crvena);  
}  
.quote {  
    color: var(--crvena); /*REFERENCIRANJE, POMENUTE VARIJABLE*/  
}
```

OCIGLEDNA KORIST JESTE, DA JEDNOSTAVNO MOGU UPDATE-OVATI HEKSADECIMALNU VREDNOST, NA SLEDECEM MESTU

```
:root {  
    --crvena: #ff6f69;  
}  
  
title {  
    color: var(--crvena);  
}  
.quote {  
    color: var(--crvena);  
}
```

I COLOR STILOVI ZA TITLE I .QUOTE CE BITI AUTOMATSKI UPDATE-OVANI

I SEMANTICKI, KORISCENJE VARIJABLI, IMA VISE SMISLA, SASVIM LAKO JE ZAKLJUCITI, DA JE TITLE ZAISTA CRVEN, NEKO KADA BIH POSMATRAO HEKSADECIMALNU VREDNOST BOJE, SEM AKO NISAM NEKO KO JE U STANJU DA PARSE-UJE HEKSADECIMALNE VREDNOSTI, BRZO U SVOJOJ GLAVI

IAKO SE JA NISAM SUSRETAO SA SASS-OM I LESS-OM, KOJI IMAJU TAKODJE VARIJABLE, MORAM SPOMENUTI DA POSTOJE PREDNOSTI U KORISCENJU CSS VARIJABLI, UMESTO SASS-A I LESS-A

-LAKSE JE POČETAK SA CSS VARIJABLAMA, ZATO STO SU ONE, NATIVE TO THE BROWSER, DAKLE NIJE POTREBAN TRANSPILING

-CSS VARIJABLE IMAJU PRISTUP DOM ELEMENTIMA-A, STO ZNACI SLEDECE:

MOGUĆE JE KREIRANJE LOKALNIH OBIMA (LOCAL SCOPES), POD TIM MISLIM NA CSS VARIJABLE, KOJE RADE SAMO U ODREDJENIM SEKCIJAMA, MOJE APLIKACIJE

VARIJABLE, TAKODJE MOGU MENJATI, UZ POMOC JAVASCRIPT-A, STO JE VEOMA KORISNO, AKO ZELIM DA OMOGUCIM DA KORISNIK MOZE, NA PRIMER MENJATI VELICINU FONT-A,

IDEALNE SU ZA RESPONSIVENESS, JER JE MOGUĆE MENJATI VARIJABLE, U ODNOSU NA MEDIA QUEERIES (@media, VEC SAM POMINJAO POMENUTO U OVOM DOKUMENTU)

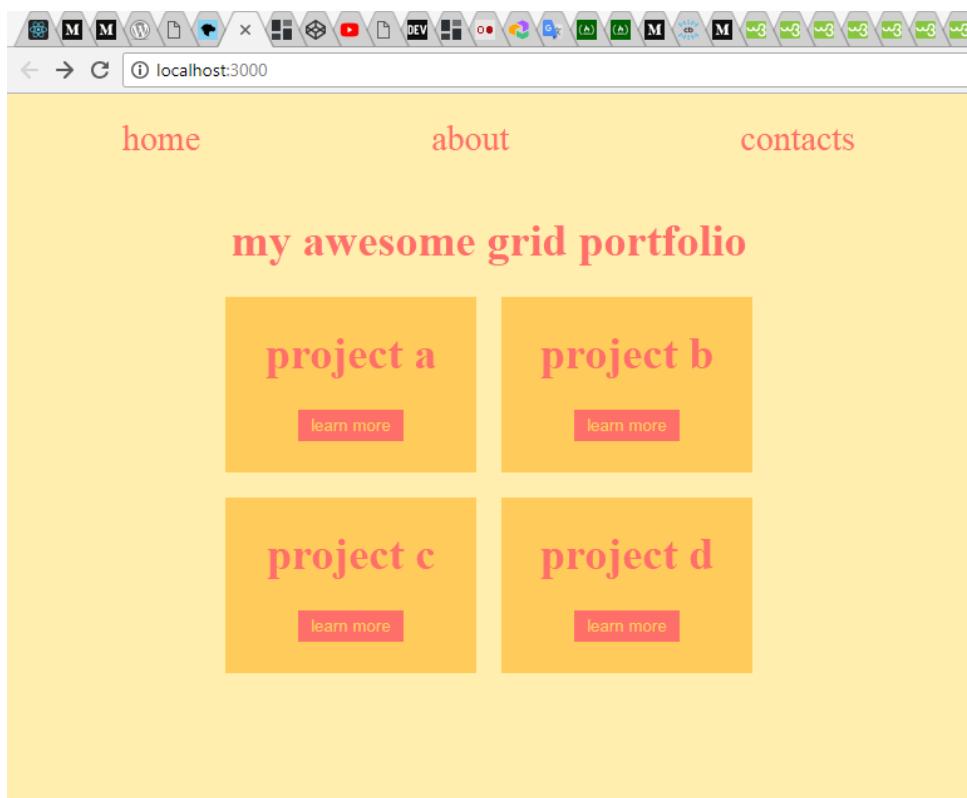
I SVI OVI, POMENUTI SLUCAJEVI, NE BI BILI MOGUCI, KORISCENJEM SASS-A I LESS-A; STO CE BITI, I PRIKAZANO U NEKOJ OD NAREDNIH LEKCIJA

-SAVRSENE SU ZA TEME (THEMES), POD TIME MISLIM NA COMPONENT SPECIFIC THEMES ITEM-A, KOJE ZELIM MALO DA PROMENIM, NAIME NE MISLIM KONKRETNO DA POSETILAC WEBSAJTA BUDE U MOGUCNOSTI DA MENJA LAYOUT

NAIME, TO MOZE DA BUDE PRODUCT, KOJI JE DODAT U SHOPPING CART, ILI ARTICLE, KOJI ZELIM DA FEATURE-UJEM

MOJA PRVA CSS VARIJABLA

U PROSLOJ LEKCIJI SAM KREIRAO PRIMER PORTFOLIO WEBSITE-A, KOJI SE SASTOJI OD NAV BAR-A, MAIN SEKCIJE (KOJA SE SASTOJI OD NEKOLIKO ITEM-A), I FOOTER-A



KAO STO VIDIM SA SLIKE GORE, PRISUTNE SU TRI BOJE

RED (DEFAULT TEXT COLOR)

BEIGE (BEZ) BOJA BACKGROUND-A

YELLOW COLOR ZA ITEM-E

```
html, body {  
    background: #ffead;  
    color: #ff6f69;  
}  
  
#navbar a {  
    color: #ff6f69;  
}  
  
.item {  
    background: #ffcc5c;  
}  
  
button {  
    background: #ff6f69;  
    color: #ffcc5c;  
}  
  
h1, p {  
    color: #ff6f69;  
}
```

I KAO STO SE VIDI, USTVARI SE REUSE-UJU HEXADECIMAL VALUES ZA BOJE, NA BAS MNOGO MESTA

MAIN RED COLOR JE PODESENA ZA BOJU TEKST-A html I body ELEMENATA

ISTU BOJA JE PODESENA I ZA h1 I p TAGOVE, ALI ZASTO JE TO URADJENO GOVORICU U NEKOJ DRUGOJ LEKCIJI

ISTA BOJA SE JOS UPOTREBLJAVA ZA TEKST ANCHOR ELEMENATA, I ZA BACKGROUND BUTTON ELEMENATA

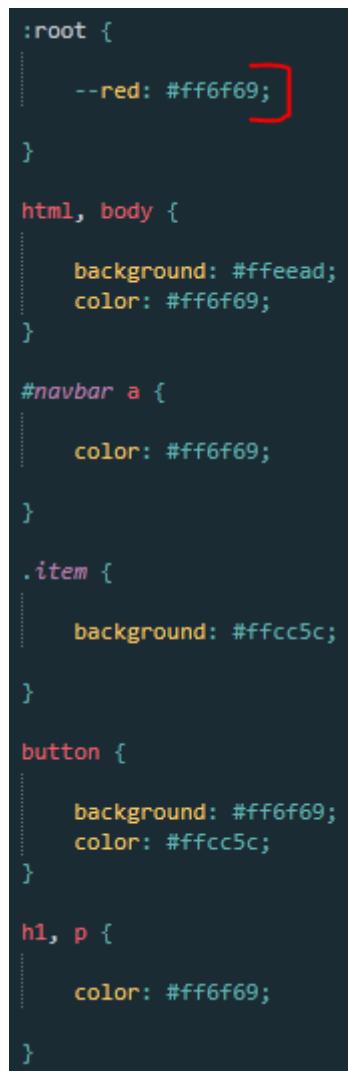
KADA GOD VIDIM DA JE CSS VREDNOST OVAKO REUSED, TO MOZE BITI SVRSENA PRILIKA ZA UVODJENJE CSS VARIJABLI, TAKO DA CU TO I DA URADIM

POCECU TAKO STO CU DEFINISATI VARIJABLJU, ALI MORAM PRVO ODLUCITI U KOJEM OBIMU ZELIM DA ZIVI, POMENUTA VARIJABLA

IZABRACU :root

NAIME POMENUTO :root JESTE PSEUDO KLASA KOJA UPIRE, ODNOSENJE UPUCUJE NA ROOT ELEMENT U DOKUMENTU, A U OVOM SLUCAJU TO JE SAM html ELEMENT

KADA DEFINISEM TU PSEUDO KLASU, KAO SADA, ZA TOP LEVEL ELEMENT, SVE VARIJABLE KOJE BUDU BILE DEKLARISANE U POMENUTOM OBIMU, BICE DOSTUPNE CELOJ APLIKACIJI; TE VARIJABLE CE BITI TAKOZVANE GLOBALNE VARIJABLE



```
:root {  
    --red: #ff6f69;  
}  
  
html, body {  
    background: #ffeedd;  
    color: #ff6f69;  
}  
  
#navbar a {  
    color: #ff6f69;  
}  
  
.item {  

```

SADA MOGU KORISTI, VARIJABLJU OZNACENU CRVENOM NA SLICI GORE, KROZ JEDNU MALO CUDNU SITAKSU

MORAM ZAPAMTITI DA MORAK KORISTITI POMENUTE DVE CRTICE (DASHES), A PRILIKOM REFERENCIRANJA POMENUTE VARIJABLE MORAM KORISTITI, I :

var()

```
:root {  
    --red: #ff6f69;  
}  
  
html, body {  
    background: #ffeedad;  
    color: var(--red);  
}  
  
#navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: #ffcc5c;  
}  
  
button {  
    background: var(--red);  
    color: #ffcc5c;  
}  
  
h1, p {  
    color: var(--red);  
}
```

SADA CU URADITI ISTO I ZA BEIGE BACKGROUND, IAKO SE KORISTI NA SAMO JEDNOM MESTU, ALI IPAK MISLIM DA JE POTREBNO STAVITI POMENUTU BOJU U VARIJABLU; JER JE TO POBOLJSAVA CILJIVOST

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedad;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: #ffcc5c;  
}  
  
button {  
    background: var(--red);  
    color: #ffcc5c;  
}  
  
h1, p {  
    color: var(--red);  
}
```

A TAKODJE CU DEFINISATI I CSS VARIJABLU, KOJA CE SKLADISTITI ZUTU BOJU, KOJU ZELIM DA IMAJU, GRID ITEMI, KAO POZADINU, ALI DA TO BUDE BOJA TEKST I BUTTON ELEMENATA

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeead;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```



VEC SE VIDE BENEFITI, POMENUTOG KORISCENJA VARIJABLI:

NE MORAM DA SE PONAVLJAM (IMAM DRY CODE, JER NE VRSIM DIREKTAN REUSE HEKSADECIMALNE VREDNOSTI)

CODE JE MORE UNDERSTANDABLE (RAZUMLJIVIJI), JER SU IMAENA VARIJABLI DESKRIPTIVNA

I OCITA KORISNOST SE OGLEDAU TOME, DA AKO ZELIM DA PROMENIM, NEKU VREDNOST; NA PRIMER DA MALO IZMENIM CRVENU BOJU, MOGU JEDNOSTAVNO PROMENITI VREDNOST --red VARIABLE

```
:root {  
    --red: #ff0f69;  
    --beige: #ffeade;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
.navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

→→→→

```
:root {  
    --red: #ff8e69;  
    --beige: #ffeedd;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
.navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

SADA CE SVI ELEMENTI, CIJA JE CRVENA BOJA DEFINISANA, POMOCU --red VARIJABLE, IMATI DRUGACIJU NIJANSU CRVENE



MEDJUTIM, VRATICU PREDHODNU NIJANSU CRVENE (NE MORAM, SADA TO POKAZIVATI OVDE)

OVERRIDING VARIABLES

U OVOJ LEKCIJI CU NAUCITI, KAKO DA OVERRIDE-UJEM VARIJABLE

U PROSLOJ LEKCIJI SAM NAUCIO KAKO DA KRIRAM VARIJABLE NA ROOT ELEMENTU, KOJI POINTS TO ROOT ELEMENT MOG document-A, A TO JE html TAG

CSS VARIJABLE FUNKCIONISU NA TAKAV NACIN DA SVA DECA (CHILDREN) ELEMENTA, ZA KOJI SAM DEKLARISAO VARIJABLE, IMAJU PRISTUP, POMENUTIM VARIJABLAMA

A CEO MARKUP, MOJE STRANICE, JESTE DESCENDANT, POMENUTOG html TAGA, STO ZNACI DA SU MOJE CSS VARIJABLE DOSTUPNE, KROZ CELOKUPNOST APLIKACIJE

MEDJUTIM, POSTOJE SLUCAJE GDE ZELIM DA VARIJABLE PROMENE SVOJE VREDNOSTI, U ODREĐENIM SEKCIJAMA APLIKACIJE

TU NA SCENU STUPA OVERRIDING

POSTO TO NIKAD NISAM URADIO RANIJE, KONKRETNO PRILIKOM BAVLJENJA SA JAVASCRIPTO, RECI CU KOJA JE RAZLIKA IZMEDJU OVERWRITING-A I OVERRIDING-A:

AKO ZAMENIS JEDNU IMPLEMENTACIJU U POTPUNOSTI SA DRUGOM, TO JE "OVERWRITING" ILI ČESTO ZVANO "ZAMENA" ILI "REPLACEMENT". A AKO ZAMENIM IMPLEMENTACIJU SA DRUGOM, SAMO U NEKIM KONKRETNIM SLUČAJEVIMA, TO JE OVERRIDING

DA "OVERWRITE-UJEM" NEŠTO, ZNAC STAVITI NEŠTO DRUGO NA NJEGOVO MJESTO, UNIŠTAVAJUCI GA. DA "override-UJEM" NEŠTO JE DA PROUZROKUJEM DA NESTO DRUGO RADI DRUGACIJE, BEZ ŠTETE ILI PROMENE OVERRIDE-OVANE STVARI

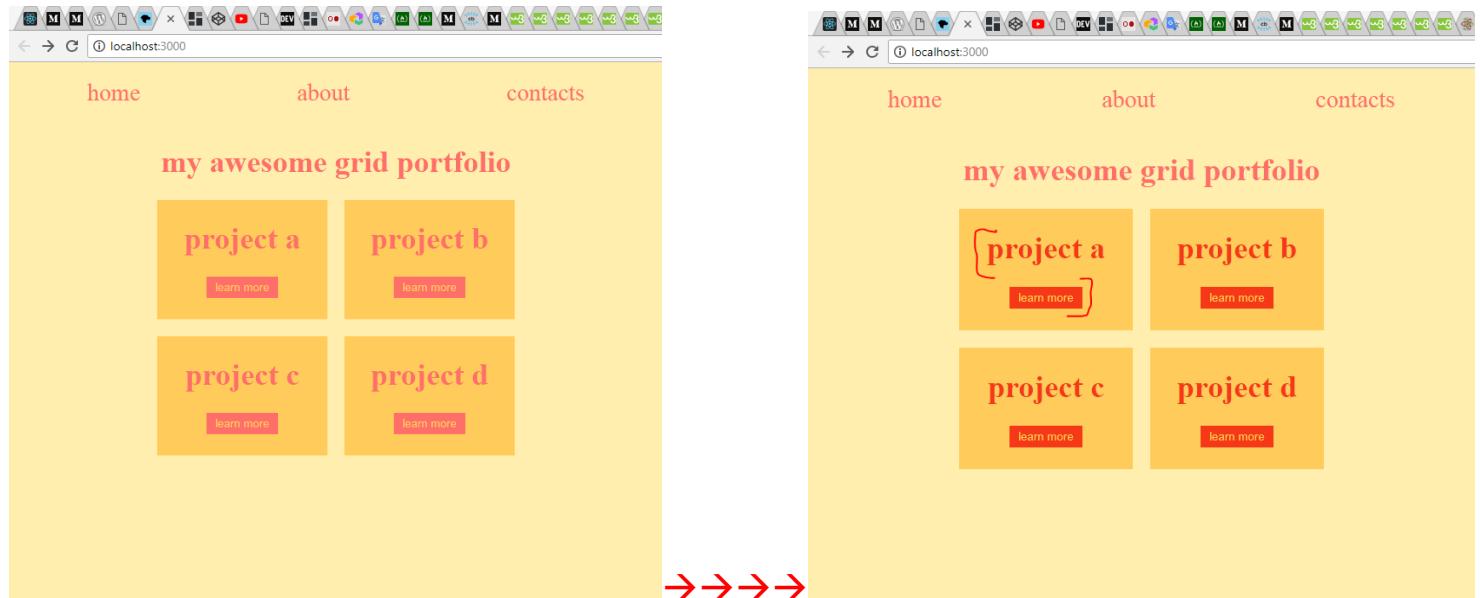
OVERRIDING NIJE NESTO STO MOGU URADITI SA SASS I LESS VARIJABLAMA, JER ONE NE ZNAJU NISTA O DOM-U, DOK CSS VARIJABLE ZNAJU

TAKO DA MOGU RECI CSS VARIJABLI, DA ZELIM DA SE VREDNOST PROMENI, NA PRIMER U DATOM ELEMENTU, I TO CU URADITI, U GRID ITEM-IMA (ELEMENTIMA, KOJI IMAJU CSS class ATRIBUT SA VREDNOSCЮ .item)

```
.item {  
    background: var(--yellow);  
}  
→→→→ .item {  
    --red: #F53817;  
    background: var(--yellow);  
}
```

KAO STO VIDIM, SA SLIKE GORE, SAMO SAM DEFINISAO NOVU VREDNOST, POMENUTE VARIJABLE, U SLUCAJU .item ELEMENATA; STO ZNACI DA JE SVAKI, .item ELEMENT, ALI I NJEGOV DESCENDANT, KOJI KORISTI --red VARIJABLJU, SADA DOBIO NOVU VREDNOST, ZA POMENUTU VARIJABLJU

ONO STO ZNAM DA JESTE DA BUTTON ELEMENTI, KOJI SU NESTED U .item ELEMENTIMA, I h1 ELEMENTI, UPRAVO KORISTI, POMENUTU --red VARIJABLU (JER SAM JA DEFINISAO RANIJE, DA POMENUTI ELEMENTI KORISTE POMENUTU VARIJABLU)



A VIDIM DA h1 ELEMENT, KOJI IMA TEKST "my awesome grid portfolio", IMA ORIGINALNU CRVENU BOJU, JER ON NIJE DESCENDANT .item ELEMENTA, I ON KORISTI --red VARIJABLU, SA ORIGINALNOM HEKSA VREDNOSCU CRVENE BOJE

LOKALNE VARIJABLE

U OVOJ LEKCIJI CU URADITI, SKORO POTPUNO ISTO KAO ONO STO SAM URADIO U PROSLOJ, GDE SAM OVERRIDE-OVAO VARIJABLU U LOKALNOM OBIMU, SAMO STO CU OVOG PUTA, KREIRATI, POTPUNO NOVU VARIJABLU, U LOKALNOM OBIMU

I OVO JE RELEVANTNO U SLUCAJEVIMA, GDE ZNAM DA CE VARIJABLA, JEDINO BITI KORISCENA U TOJ SEKCIJI APLIKACIJE

ZATO NEMA POTREBE, STAVLJATI JE U GLOBALNI OBIM, ODNOSNO OBIM :root-A

SADA NA PRIMER ZELIM DA CRVENA BOJA U NAV BAR-U, IMA DRUGACIJU NIJANSU, NEGO STO IMA KROZ OSTATAK APLIKACIJE

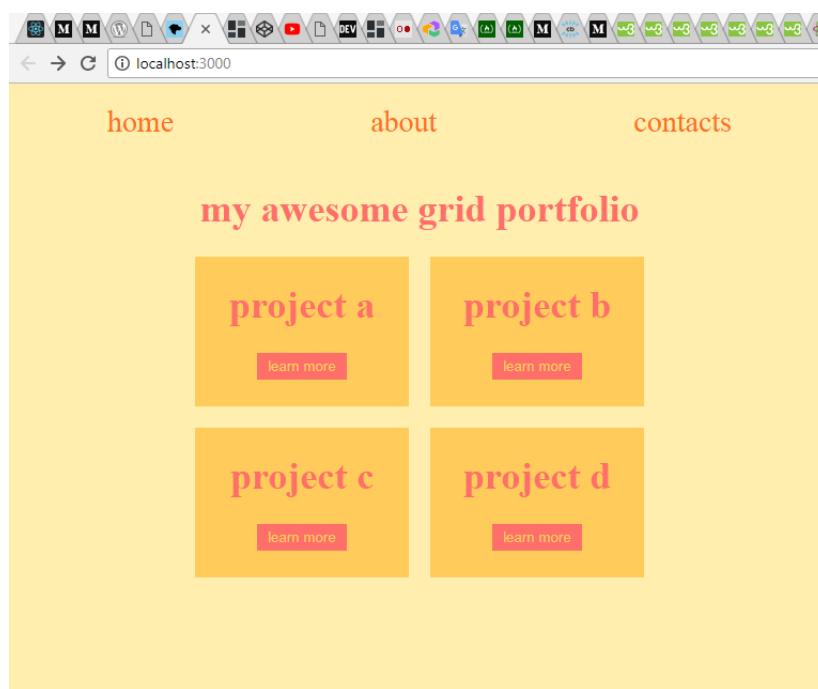
ZATO MOGU U OBIMU .navbar KREIRATI NOVU VARIJABLU, KAO STO SAM TO URADIO U OBIMU :root-A

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedad;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
.navbar {  
    --nav-red: #ff6f19;  
}  
  
.navbar a {  
    color: var(--red);  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

A SADA CU UMESTO PREDHODNE CRVENE (ODNOSNO PREDHODNE REFERENCE VARIJABLE), KOJA JE KORISCENA, KAO VREDNOST BOJE TEKSTA, ZA VREDNOST DEFINISATI NOVU CRVENU, ODNOSNO REFERENCU NOVE VARIJABLE, KOJU SAMO MOZE KORISTITI `#navbar` ELEMENT I NJEGOVI DESCENDANTI

```
:root {  
    --red: #ff6f69;  
    --beige: #ffead;  
    --yellow: #ffcc5c;  
  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar {  
    --nav-red: #ff6f19;  
  
}  
  
#navbar a {  
    color: var(--nav-red);  
  
}  
  
.item {  
    background: var(--yellow);  
  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

SADA JE BOJA ANCHOR ELEMENATA, ORANGISH



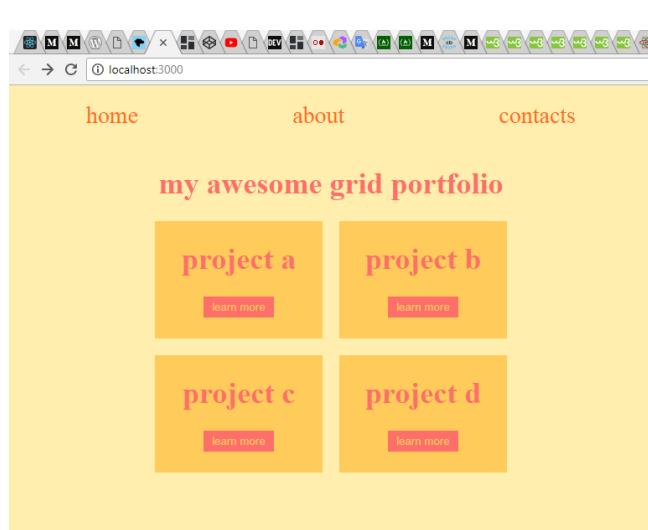
DA SAM KORISTIO, POMENUTI --nav-red VARIJABLU, U NEKOM DRUGOM ELEMENTU KOJI JE NA PRIMER SIBLING #navbar ELEMENTU, NE BIH NISTA POSTIGAO

KREIRACU SADA I NOVU LOKALNU VARIJABLU, U SLUCAJU .item ELEMENTA

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedad;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
.navbar {  
    --nav-red: #ff6f19;  
}  
  
.navbar a {  
    color: var(--nav-red);  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```



```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedad;  
    --yellow: #ffcc5c;  
}  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
.navbar {  
    --nav-red: #ff6f19;  
}  
  
.navbar a {  
    color: var(--nav-red);  
}  
  
.item {  
    --it-yellow: #ffcc8c;  
    background: var(--it-yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```



ALI KAO STO VIDIM, POSTO SE U BUTTONIM-A REFERENCIRA GLOBALNA YELLOW BOJA, ODNOSNO BOJA KOJU SKLADISTI --yellow VARIJABLA, ONA NECE BITI PROMENJENA

UMESTO --yellow VARIJABLE, MOGAO SAM I ZA BUTTON ELEMENT REFERENCIRATI --it-yellow VARIJABLU

A TO DA SAM URADIO ZA NEKI DRUGI TAG KOJI NIJE DESCENDANT .item ELEMENT, NE BI IMALO ZELJENI EFEKAT

A MOGU ODVOJITI DEKLARISANJE VARIJABLE I NJENO REFERENCIRANJE NA SLEDECI NACIN; STO JE USTVARI STVAR LICNOG IZBORA

```
.item {  
    --it-yellow: #ffcc8c;  
}  
  
.item {  
    background: var(--it-yellow);  
}
```

THEMING WITH CSS VARIABLES

OVDE CE SE PRICATI O TEMAMA, JER JE TO OBLAST GDE CSS VARIJABLE, JESU VEOMA KORISNE MOZDA MISLIM SLEDECE:

NECU DA DOZVOLIM KORISNICIMA DA MENJAJU TEME NA MOM WEBSAJTU, PA ZASTO BI ONDA UCIO OVO?

NAIME, OVDE SE NE GOVORI SAMO O USER SPECIFIC THEMES, VEC I O COMPONENT SPECIFIC THEMES, KOJI JE NORMALNIJI SLUCAJ KORISCENJA

NA PRIMER, MOZDA ZELIMA DA OZNACIM (MARK) ITEM U ECOMERCE PRODAVNICI, KAO KUPIJEN ILI DA GA DODAM U CART (KORPU), A MOZE BITI DA IMAM I ADMI NSEKCIJU SAJTA, KOJA TREBA DA IMA DRUGACIJU TEMU, MOZDA TAMNIJU NEGO WEBSITE ITSELF, ILI NESTO STO CU JA URADITI U OVOJ LEKCIJI, A TO JE SLEDECE

FEATURING NECEGA NA MOM SAJTU

DAVAJUCI MU FEATURE CLASS, KAKO BI SE ISTICALO OD OSTATKA WEBSAJTA

DODACU SADA FEATURE KLASU, JEDNOM OD ITEMA MOG PRIMERA, KAKO BI SE ON ISTICAO
ONO STO BI PRVO MORAO DA URADIM JESTE DODAVANJE FEATURE KLASE U HTML, OVAKO

```
<header>
  <nav id="navbar">
    <ul>
      <li><a href="#">home</a></li>
      <li><a href="#">about</a></li>
      <li><a href="#">contacts</a></li>
    </ul>
  </nav>
</header>
<main>
  <h1>my awesome grid portfolio</h1>
  <div class="grid">
    <div class="item">
      <h1>project a</h1>
      <button>learn more</button>
    </div>
    <div class="item">
      <h1>project b</h1>
      <button>learn more</button>
    </div>
    <div class="item">
      <h1>project c</h1>
      <button>learn more</button>
    </div>
    <div class="item featured">[redacted]
      <h1>project d</h1>
      <button>learn more</button>
    </div>
  </div>
</main>
<footer><p>Made with 🇳🇴 in Norway</p></footer>
```

ZATIM DEFINISANJE STILA NA SLEDECI NACIN

```
.featured {  
}  
.  
.featured > h1 {  
}  
}  
.featured button {  
}  
}
```

TO BI BILO MNOGO EXTRA CSS-A, KAKO BI POSTIGAO POMENUTO

LEPOTA KORISCENJA CSS VARIJABLI SE OGLEDA U TOME DA MI .feature > h1 I .feature > button, NE TREBAJU, JER MOGU SVE DA DEFINISEM NA SLEDECI NACIN

```
.featured {  
    --yellow: #F4E60C;  
    --red: #FC500F;  
}
```



JASNO MI JE KOJI SE SADA ITEM ISTICE OD OSTALIH ITEM-A

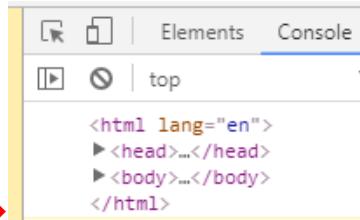
MENJANJE VARIJABLJI SA JAVASCRIPT-OM

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeade;  
    --yellow: #ffcc5c;  
}
```

POSTO CSS VARIJABLE "ZIVE" U DOM-U, TAKO DA IH MOGU, UPDATE-OVATI, POMOCU JAVASCRIPT-A, STO JE SUPER COOL, I TO NIJE NESTO STO MOGU URADITI SA SASS ILI LESS VARIJABLAMA, JER SU ONE TRANSPILED DO REGULARNOG CSS-A

ONO STO CU PRVO URADITI JESTE PRISTUPANJE, POMENUTOM :root ELEMENTU

```
import React from 'react';  
import ReactDOM from 'react-dom';  
import './basic.css';  
import './index.css';  
  
const root = document.querySelector(':root');  
  
console.log(root);
```



ZATIM CU PRISTUPITI STILOVIMA, POMENUTOG ELEMENTA, UZ POMOC METODE, ODNOSNO FUNKCIJE:

getComputedStyle (REFERENCA DOM ELEMENTA, JOJ JE ARGUMENT)

```
const root = document.querySelector(':root');  
  
console.log(root);  
  
const rootStyles = getComputedStyle(root);  
  
console.log(rootStyles)
```



KAO STO VIDIM SA SLIKE GORE, POV RATNA VREDNOST, POMENUTE FUNKCIJE JESTE CSSStyleDeclaration INSTANCA, MEDJUTIM BITNO JE RECI DA JE OVAJ OBJEKAT, USTVARI READ-ONLY

KAKO BI PRISTUPIO VREDNOSTI, ODREDJENOG PROPERTIJA, KORISTIM SLEDECU METODU, CSSStyleDeclaration-OVOG PROTOTIPA

getPropertyValue

```
const root = document.querySelector(':root');  
  
console.log(root);  
  
const rootStyles = getComputedStyle(root);  
  
console.log(rootStyles)  
  
const red = rootStyles.getPropertyValue("--red");  
  
console.log(red);
```



A PRIMENOM setStyle METODE, NAD CSSStyleDeclaration, ODNOSNO SETTER METODE, MOGU PROMENITI VREDNOST ODREDJENOG STILA

BITNO JE RECI DA PRVO MORAM PRISTUPITI STILOVIMA, POMOCU style PROPERTIJA
style PROPERTIJEM SE PRISTUPA ZELJENOJ CSSStyleDeclaration INSTANCI

```
root.style.setProperty('--red', '#F60547');
```



ZAPAMTI SLEDECU STVAR

```
console.log(rootStyles === root.style) →→→ false
```

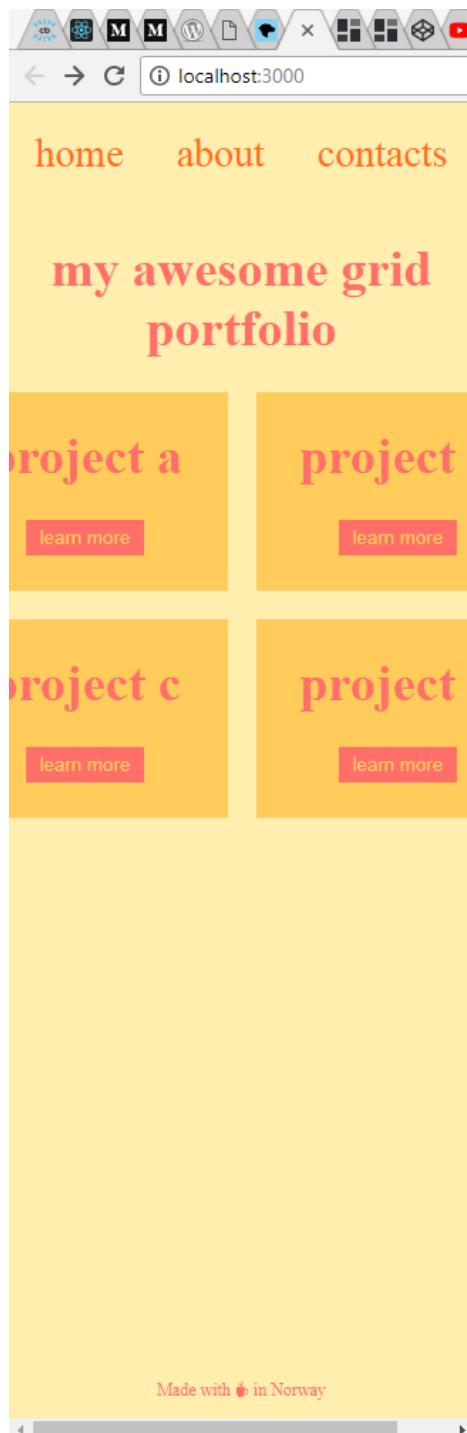
ZAPAMTI DA FUNKCIJU getComputedStyle POZIVAS (DODAJUCI JOJ DOM OBJEKAT KAO ARGUMENT), KAKO BI SAMO IMAO UVID U VREDNOST, ODREDJENOG PROPERTIJA, ODNOSNO CSS VARIJABLE U OVOM SLUCAJU; A KONKRETNOJ VREDNOSTI PRISTUPAM UZ POMOC getPropertyValue MEDODE, KOJA JE METODA CSSStyleDeclaration INSTANCE, KOJA JE POV RATNA VREDNOST getComputedStyle FUNKCIJE, A POMENUTA INSTANCA JE READ-ONLY

AKO ZELIM DA PROMENIM ZELJENU VREDNOST, ONDA PRISTUPAM CSSStyleDeclaration INSTANCI, KOJ JE VREDNOST style PROPERTIJA, NEKOG DOM ELEMENTA, I NAD TOJ INSTANCI PRIMENUJUJEM setProperty METODU, KOJOM DODELUJEM NOVU VREDNOST ZA ZELJENI CSS PROPERTI, ILI U OVOM SLUCAJU VREDNOST ZA ZELJENU CSS VARIJABLU

RESPONSIVENESS AND CSS VARIABLES

U OVOJ LEKCIJU CU SE BAVITI TIME, KAKO DA DEFINISEM PROMENU VARIJABLE, U ODNOSU NA SIRINU EKRANA

ONO STO ZELIM DA DEFINISEM JESTE DA SE GRID ITEMI STACK-UJU JEDAN NA DRUGI, U SLUCAJU MANJEG EKRANA, JER SADA JE MOJ GRID TOO WIDE FOR THE SCREEN



ONO STO NEKI DEVELOPERI VOLE JESTE DEKLARISANJE SVIH VARIJABLJI NA VRHU

PORED TOGA MOGU DODATI I KOMENTAR, DA JE REC O VARIJABLAMA

A MOGU DODATI I KOMENTAR, TAMO GDE POCINJE DEFINISANJE CSS SELEKTORA, ODNOSNO STILOVA, KAKO BI IMAO STILOVE I VARIJABLE RAZDVOJENE

```

/*Variable declarations*/
.grid {
    --columns: 200px 200px;
}

/* styles */
body, html {
    margin: 0;
    padding: 0;
}

a {
    text-decoration: none;
}

h1 {
    text-align: center;
    font-size: 36px;
}

#navbar {
    padding: 20px 0;
    font-size: 24px;
}

.grid {
    display: grid;
    grid-template-columns: var(--columns);
    grid-auto-rows: 140px;
    grid-gap: 20px;
    justify-content: center;
}

```

```

a {
    font-size: 28px;
}

#navbar > ul {
    display: flex;
    justify-content: space-around;
    margin: 0 auto;
    padding: 0;
    list-style-type: none;
}

p {
    text-align: center;
}

h1 {
    font-size: 36px;
}

p {
    font-size: 18px;
}

footer {
    position: absolute;
    bottom: 0;
    width: 100%;
}

footer > p {
    font-size: 12px;
}

button {
    margin: 0px auto;
    display: block;
    padding: 5px 10px;
    border: none;
}

```

basic.css

SADA CU PROMENITI VREDNOST, POMENUTE VARIJABLE, ZA SLUCAJ, KADA SIRINA EKRANA BUDE
MANJA OD 450PX

```

/*Variable declarations*/
.grid {
    --columns: 200px 200px;
}

/* Styles */
body, html {
    margin: 0;
    padding: 0;
}

a {
    text-decoration: none;
}

h1 {
    text-align: center;
    font-size: 36px;
}

#navbar {
    padding: 20px 0;
    font-size: 24px;
}

.grid {
    display: grid;
    grid-template-columns: var(--columns);
    grid-auto-rows: 140px;
    grid-gap: 20px;
    justify-content: center;
}

@media all and (max-width: 450px) {
    .grid {
        --columns: 200px;
    }
}

```

KAO STO VIDIM, MORAO SAM OVERRIDING, USTVARI OBMOTATI (WRAPP IT) U ISTI OBIM, U KOJEM SAM DEFINISAO VARIJABLJU, A TO JE .grid OBIM



Made with in Norway

NI OVO NIJE NESTO STO BI MOGLO BITI URADJENO SA LESS ILI SASS VARIJABLAMA, JER ONE NEMAJU INFORMACIJU O NICEMU STO SE DOGADJA U DOM-U, JER SU ONE TRANSPILED DO REGULARNOG CSS-A

A POSTO, POMENUTE CSS VARIABLE, ZIVE U DOM-U, MOGU IH MENJATI, PO NA PRIMER, RESIZE-U SCREEN-A, KAO STO JE TO OVDE BIO SLUCAJ, ILI KAD JE U PITANJU SCREEN SIZE IN GENERAL

DVA PRIMERA INHERITANCE-A

ONO O CEMU CE SE GOVORITI U OVOJ LEKCIJI JESTE, NESTO O NASLEDJIVANJU

KAO STO SAM VIDEO TOKOM OVOG UPOZNAVANJA SA CSS VARIJABLAMA, ONE SE PONASAJU PRILICNO KAO I DRUGI CSS PROPERTIJI

NAIME NJIHOVE VREDNOSTI SU NASLEDIVE I CASCADING

MEDJUTIM POSTOJE NEKE STVARI ZA KOJE SE MOZE PREDPOSTAVITI DA BI FUNKCIONISALE SA CSS VARIJABLAMA, A ZAPRAVO TE STVARI NE FUNKCIONISU

ONO STO JESTE MOGUCE, JE REFERENCIRANJE JEDNE VARIABLE, KAO VREDNOSTI DRUGE

TO MOGU URADITI NA SLEDECİ NACIN

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedd;  
    --yellow: #ffcc5c;  
  
    --main-color: var(--red);  
}  
  
/* Styles */  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar a {  
    color: var(--main-color);  
}  
  
.item {  
    --it-yellow: #ffcc8c;  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

ALI AKO SADA URADIM SLEDECE (OZNACENO CRVENOM):

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeead;  
    --yellow: #ffcc5c;  
  
    --main-color: var(--red);  
}  
  
/* Styles */  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar {  
    --red: orange;  
}  
  
#navbar a {  
    color: var(--main-color);  
}  
  
.item {  
    --it-yellow: #ffcc8c;  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

TO NECE PROMENITI, ONO STO SAM OZNACIO ZELENOM NA GORNJOJ SLICI

UPRAVO ZATO STO JE VREDNOST BOJE VARIJABLE --red, DODELJENA NOVOJ VARIJABLI, BOLJE RECENO, VREDNOST --main-color VARIJABLE JE, VEC RESOLVED SA VREDNOSCU --red VARIJABLE --main-color NEMA REFERENCU --red VARIJABLE, VEC SAMO VREDNOST BOJE, KOJA JOJ JE DATA

ONO STO U OVOM SLUCAJU MOGU URADITI, JESTE OVERRIDING VARIABLE --main-color

```
/*Variable declarations*/
:root {
    --red: #ff6f69;
    --beige: #ffeead;
    --yellow: #ffcc5c;

    --main-color: var(--red);
}

/* Styles */
html, body {
    background: var(--beige);
    color: var(--red);
}

#navbar {
    --main-color: orange;
}

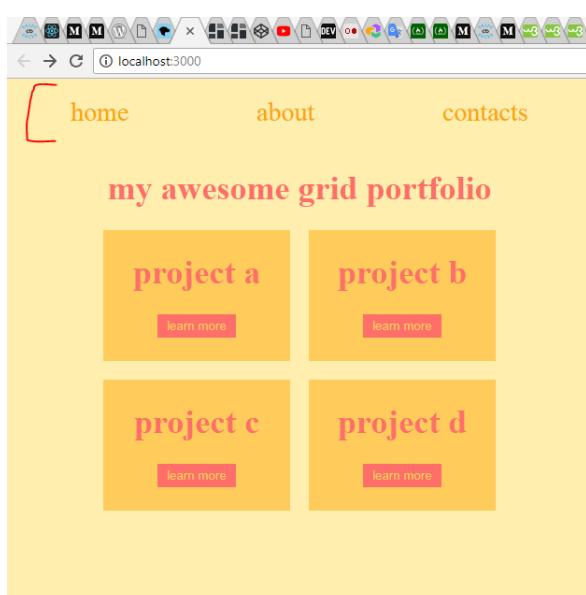
#navbar a {
    color: var(--main-color);
}

.item {
    --it-yellow: #ffcc8c;
}

.item {
    background: var(--yellow);
}

button {
    background: var(--red);
    color: var(--yellow);
}

h1, p {
    color: var(--red);
}
```



ONO STO SAM JOS RANIJE DEFINISAO U MOJIM STILOVIMA JESTE SELEKTOR ZA h1 I ZA p

```
:root {  
    --red: #ff6f69;  
    --beige: #ffeedd;  
    --yellow: #ffcc5c;  
  
    --main-color: var(--red);  
}  
  
/* Styles */  
  
html, body {  
    background: var(--beige);  
    color: var(--red);  
}  
  
#navbar {  
    --main-color: orange;  
}  
  
#navbar a {  
    color: var(--main-color);  
}  
  
.item {  
    --it-yellow: #ffcc8c;  
}  
  
.item {  
    background: var(--yellow);  
}  
  
button {  
    background: var(--red);  
    color: var(--yellow);  
}  
  
h1, p {  
    color: var(--red);  
}
```

NAIME, OVO JE BILO NEPOTREBNO, JER OVI ELEMENTI, NASLEDJUJU VREDNOST color PROPERTIJA, KOJU SAM DEFINISAO ZA htm I body TAGOVE, I AKO UKLONIM, POMENUTI SELEKTOR, NISTA SE NECE PROMENITI, JER body I html IMAJU var(--red) VREDNOST color PROPERTIJA

ALI AKO UKLONIM POMENUTI SELEKTOR, SLEDECE (OZNACENO CRVENOM) NECE IMATI NIKAVOG EFEKTA NA h1 i p ELEMENTE

```

:root {
  --red: #ff6f69;
  --beige: #ffeedd;
  --yellow: #ffcc5c;
  --main-color: var(--red);
}

/* Styles */

html, body {
  background: var(--beige);
  color: var(--red);
}

#navbar {
  --main-color: orange;
}

#navbar a {
  color: var(--main-color);
}

.item {
  --it-yellow: #ffcc8c;
}

.item {
  background: var(--yellow);
  --red: #F918BB;
}

button {
  background: var(--red);
  color: var(--yellow);
}

```

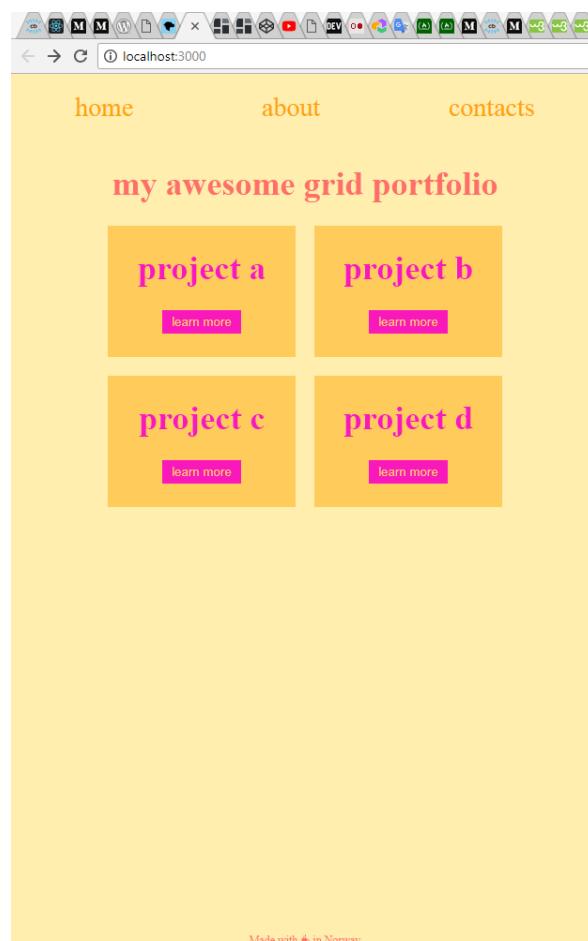


ZATO CU VRATITI ONO PREDHODNO DA h1, p ELEMENTI IMAJU color, SA VREDOSCU var(--red) (p ME USTVARI I NEZANIM, JE NEMA NI JEDAN p NESTED U GRID ITEM-IMA)

```

h1, p {
  color: var(--red);
}

```



PRIMETI ISTO DA PARAGRAF IZ FOOTERA NIJE PROMENIO BOJU, JER VREDNOST --red VARIJABLE NIJE OVERRIDE-OVAN ZA TAJ ELEMENT