Analiza metoda i alata kod izrade responzivnih internetskih stranica

Golubićek, Marko

VELEUČILIŠTE VERN'

Zagreb

Poslovna informatika

ZAVRŠNI RAD

Analiza metoda i alata kod izrade responzivnih internetskih stranica

Marko Golubićek

Zagreb, 2019.

VELEUČILIŠTE VERN'

Preddiplomski stručni studij

Poslovna informatika

ZAVRŠNI RAD

Analiza metoda i alata kod izrade responzivnih internetskih stranica

Mentor: Tvrtko Begović, dipl. ing.

Student: Marko Golubićek

Zagreb, siječanj 2019.

SADRŽAJ

SAŽETAK

Cilj ovog rada je objasniti što su to responzivne internetske stranice i koje su sve metode i alati koji se upotrebljavaju u izradi istih.

Prvo je ukratko objašnjena povijest responzivnih stranica. Zatim su detaljno teorijski obrađene sve najčešće korištene metode i alati: HTML, CSS, *grid*, *flexbox*, *media query* upiti i responzivne slike. Na kraju je opisana izrada dvije jednostavne responzivne internetske stranice pomoću opisanih metoda i alata da bi se demonstriralo kako se oni međusobno nadopunjuju kako bi se postigla responzivnost, jedna kao primjer stranice za prijavu a druga kao primjer galerije slika. U zaključku su opisani rezultati teorijske i praktične obrade svih navedenih metoda i alata i dani su prijedlozi što i kako koristiti u budućnosti.

Ključne riječi: HTML, CSS, responzivni dizajn

SUMMARY

Title: Analysis of methods and tools used to develop responsive web pages

The purpose of this thesis is to explain what responsive web pages are and what methods and tools are used to make them.

A brief history of responsive web pages is explained first. Then, all the most common tools and methods are explained in theory: HTML, CSS, grid, flexbox, media queries and responsive images. Finally, two simple responsive web pages are made, one as an example of a login page and other as an example of a picture gallery, using described tools and methods to demonstrate how they interact with each other to accomplish responsiveness. In conclusion, all the results are presented and advice is given on how to use these tools and methods in the future.

Key words: HTML, CSS, responsive design

1. UVOD

Tijekom trogodišnjeg studiranja, autor se susreo s mnogim programskim jezicima, okruženjima, bazama i alatima. Programiralo se za mobitel, za desktop i za internet, ali grupa predmeta vezanih za internet je uvijek malo odstupala. Od učenja prvih koraka u HTML-u i CSS-u do izrade kompleksnih PHP stranica pomoću vanjskih biblioteka, dizajn internetskih stranica je uvijek bilo nešto gdje je autor htio znati više i bolje. Ideja za ovaj rad nastala je na praksi, na kojoj je autor dobio stranicu staru 15 godina i zadatak da ju napravi ispočetka, a uvjet je bio samo jedan: stranica mora biti responzivna. Ovim radom autor želi široj javnosti približiti responzivni internetski dizajn.

Rad je zamišljen u pet cjelina: nakon uvoda objašnjava se što je to responzivni dizajn, kada je i kako nastao i zašto se sve više dizajnera odlučuje upravo za njega. Kratko će se pokazati kako su se radile stranice prije responzivnog dizajna. Treće i najvažnije poglavlje teorijski pokriva sve neophodne i najčešće korištene metode i alate za izradu responzivnih stranica: objasnit će se HTML i CSS dokumenti i kako kombinirati razne atribute da bi se sadržaj i slike dinamički mijenjali s obzirom na veličinu ekrana.

U četvrtom poglavlju će se pokušati upotrijebiti sve do tad spomenuto da se izrade dvije responzivne stranice, kako bi se iz prve ruke vidjelo kako koji atribut utječe na elemente na stranici. U zadnjem poglavlju će se vidjeti je li izrada stranice bila uspješna, što je njome postignuto i može li se istim ili sličnim kodom napraviti stranica za neku potpuno drugu svrhu.

Ovim radom su pokrivena najnužnija znanja isključivo za samostalnu izradu responzivnih stranica i zato aplikacije koje daju gotova rješenja bez da korisnik mora imati predznanja o izradi stranica nisu obrađene.

2. POJAM I POVIJEST RESPONZIVNOG DIZAJNA

U ovom poglavlju se ukratko definira što je to responzivni dizajn i koji su njegovi ključni elementi. Ukratko će biti opisano što se koristilo prije responzivnog dizajna i kako je responzivni dizajn kroz vrijeme dobio na važnosti.

2.1. Definicija responzivnog dizajna

Termin "responzivni *web* dizajn" (eng. *responsive web design*) je 2010. godine iskovao web dizajner Ethan Marcotte. U svojoj knjizi Marcotte kaže da trebamo stvarati stranice koje su ne samo fleksibilne, nego se prilagođavaju mediju koji ih prikazuje (Marcotte, 2011., str. 8 i 9). Ukratko, responzivni web dizajn je izrada internetskih stranica čiji se izgled na ekranu prilagođava uređaju koji ih prikazuje, s ciljem da korisničko iskustvo bude ujednačeno, bio on na stolnom računalu, tabletu ili mobitelu.

Dalje u tekstu navodi koja su tri elementa na koja trebamo skrenuti pozornost kada se bavimo responzivnim dizajnom:

1. *grid layout*
2. responzivne slike
3. *media queries*

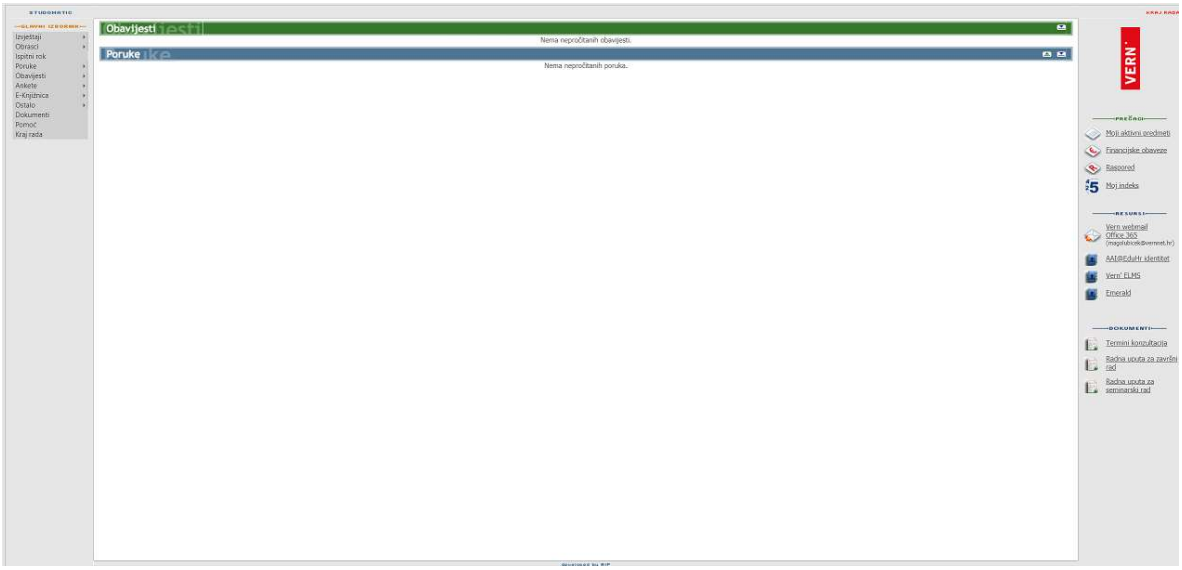
Ova tri elementa su jedne od metoda koje će se analizirati i koristiti u ovom radu da bi se prikazao način na koji ti elementi utječu na izgled prikazane stranice.

2.2. Povijest responzivnog dizajna

Prije responzivnog dizajna, između 2007. i 2010. godine, praksa je najčešće bila ovakva: ako je stranica postojala na desktopu prije pojave pametnih uređaja, naknadno se radila mobilna inačica stranice, koja je bila potpuno neovisna o desktop inačici. Vrlo dobar primjer za to je stranica Studomatic Veleučilišta VERN'. Na slikama

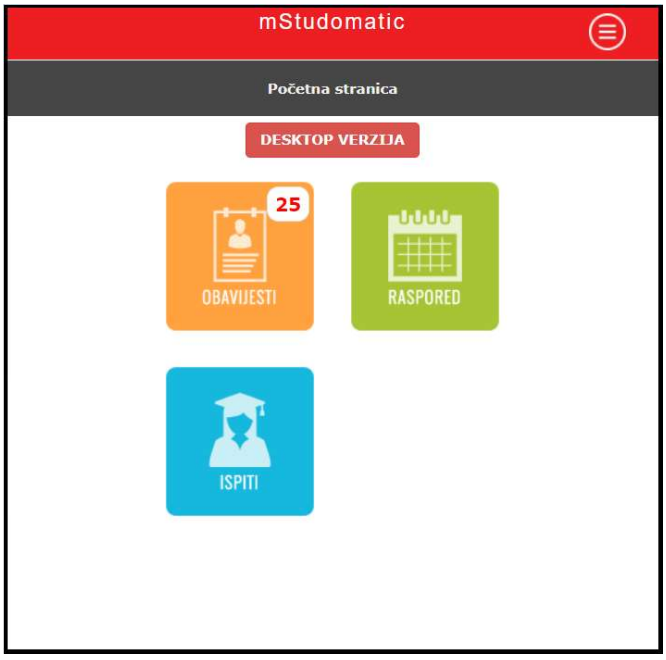
2.1 i 2.2 se prikazuje kako izgleda stranica Studomatic na stolnom računalu, a kako na mobitelu.

Slika 2.1 Inačica stranice Studomatic za stolna računala



Izvor: https://eduneta.vern.hr/vern-student/Default.aspx (25.11.2018.)

Slika 2.2 Mobilna inačica stranice Studomatic

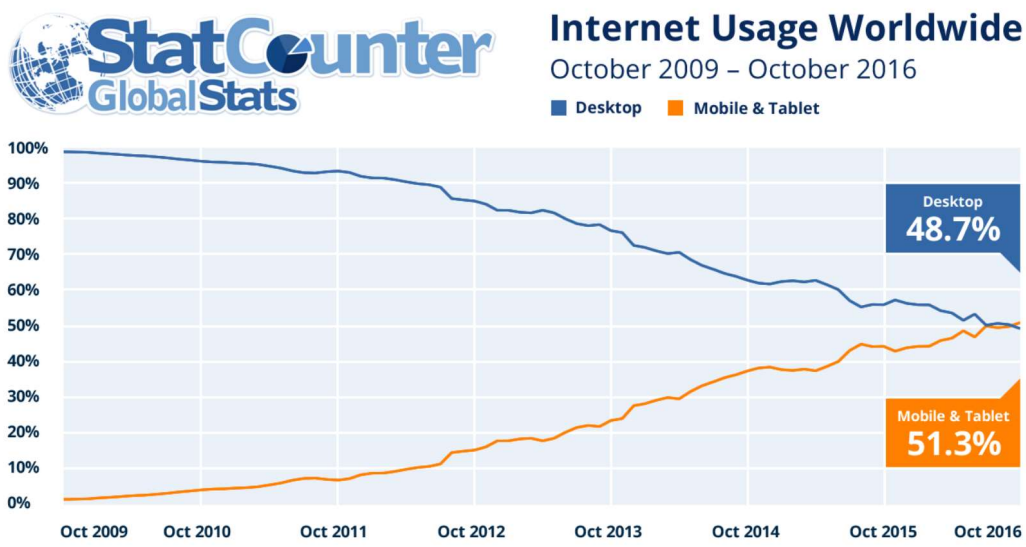


Izvor: https://eduneta.vern.hr/vern-mstudent/home.aspx (25.11.2018.)

2010. godine Ethan Marcotte nije izumio nešto što do tada nije postojalo. Dapače, *media queries* u nekom obliku spominju se već 1994. godine, u prvom nacrtu CSS-a (Hakom W Lei, 1994.), a Audi je 2002. godine imao stranicu koja se prilagođavala veličini ekrana korisnika. Međutim, Marcotte je bio taj koji je prvi uočio da su upravo tri spomenuta elementa, među stotinama drugih, najvažnija za najbolje moguće korisničko iskustvo u vremenu koje dolazi. To je vrijeme u kojem broj mobilnih korisnika sustiže i polako prestiže broj desktop korisnika na internetu.

Broj mobilnih korisnika interneta je toliko brzo rastao da je Google već 2015. godine objavio da će responzivne stranice dobivati bolji rejting i prikazivati se među prvima u rezultatima pretrage[1]. 2016. godine je broj korisnika koji pristupaju internetu preko mobitela prestigao broj korisnika koji internet posjećuju preko stolnih računala što se vidi na slici 2.3. Danas tvrtke koje su među prvima prihvatile responzivan dizajn ostvaruju veću posjećenost i prodaju od onih koje to još nisu napravile.

Slika 2.3 Broj korisnika koji pristupaju stranicama preko različitih uređaja



Izvor: http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide (25.11.2018.)

[1] Preuzeto s https://www.forbes.com/sites/thesba/2015/03/26/now-is-the-time-for-responsive-design/#9af943321f92 (25.11.2018.)

3. METODE I ALATI ZA IZRADU RESPONZIVNIH STRANICA

Slijedi najvažnije poglavlje ovog rada. U njemu će biti objašnjeni svi pojmovi usko vezani za izradu responzivnih stranica - HTML, CSS, *media queries*, *grid layout* i responzivne slike. Opisat će se i *flexbox*, najaktualniji modul za raspored elemenata koji usko surađuje s *grid layout* modulom kako bi se sadržaj rasporedio po ekranu.

3.1. HTML

HTML (eng. *Hypertext Markup Language*) je jezik koji služi za pisanje hipertekstualnih dokumenata koje prikazuju internetski preglednici. Hipertekstualni dokumenti specifični su po tome što sadrže hiperveze koje međusobno povezuju te dokumente, stvarajući mrežu po kojoj korisnici mogu "šetati" od dokumenta do dokumenta. Zato se i informacijski sustav na internetu zove *World Wide Web*[2].

HTML dokument se sastoji od skupa HTML oznaka (eng. *HTML tags*) koje preglednik čita, interpretira, a zatim prikazuje na ekranu na način koji je korisniku jednostavan za razumijevanje. <html>, <head> i <body> su tri oznake koju su najnužniji dio svakog HTML dokumenta. Svaka otvorena oznaka mora biti i zatvorena, na primjer oznaka mora biti zatvorena sa .

Iako u nazivu HTML ima riječ *jezik*, to nije programski jezik u pravom smislu te riječi jer ne služi za nikakvo programiranje, nema nikakve varijable i funkcije a o nekim složenijim komponentama da i ne govorimo. HTML služi isključivo tome da bi sadržaju koji se mora prikazati na stranici dali neke oznake kako bi ga preglednik znao ispravno prikazati.

Hipertekstualni dokumenti su temelj svake internetske stranice. Svi ostali alati, poput CSS-a, dodaju se HTML-u kako bi promijenili njegov izgled i ponašanje, ali nisu nužni. Postoje stranice koje se sastoje isključivo od HTML oznaka i sadržaja u njima. Iako

[2] eng. *web* = mreža

nisu lijepe, ako su oznake pravilno korištene, korisničko iskustvo ne mora biti nužno loše. HTML samo po sebi nije responzivan. Slijedi primjer jednostavnog HTML koda:

```
<html>
   <head>
      <title>Naslov u head oznaci</title>
   </head>
   <body>
      <h1>Naslov u body oznaci</h1>
   </body>
</html>
```

Tih jednostavnih 8 linija daje rezultat prikazan na slici 3.1.

Slika 3.1 Rezultat pokretanja HTML dokumenta

Izvor: vlastiti rad autora

3.2. CSS

CSS (eng. *Cascading Style Sheets*) najčešće dolazi u obliku dodatnog dokumenta koji se pridružuje HTML dokumentima kako bi se oznakama dodijelili posebni atributi poput veličine, boje, poravnavanja i još stotine drugih.

Pravila pisanja CSS-a su vrlo jednostavna: uvijek se prvo navodi selektor a zatim se unutar šiljatih zagrada deklarira niz pravila koja se odnose na taj selektor. Selektori određuju koji HTML element se mijenja i ima ih nekoliko vrsta, a ovo su najvažniji:

- selektor oznake - cilja čiste HTML oznake. Primjer: p{ } cilja sve paragrafe
- id selektor - id bi trebao biti jedinstven na stranici tako da ovaj selektor određuje pravilo za jedan jedini element na stranici. Za odabiranje id elementa rabi se znak # i naziv id-ja. Primjer: #lozinka { }
- selektor klase - ovaj selektor ima ista svojstva kao i id selektor osim što ga se može upotrijebiti više puta na stranici a poziva ga se točkom pa nazivom klase. Primjer: .plavo { }
- grupni selektor - koristi se kada se želi postići da više različitih elemenata ima ista svojstva. Primjer: p, h1, div { }

Selektori mogu ciljati i neka promjenjiva stanja na stranici. Na primjer, ako se selektoru *a:hover* dodijeli atribut {color: red;}, on će promijeniti boju teksta linka u crvenu kada se preko njega prelazi mišem. Atributima se nekim elementima mogu dodijeliti čak i jednostavne animacije.

CSS je mjesto gdje se stavljaju *media query* upiti te dodaju atributi *flexbox* i *grid* elementima kako bi se postigla responzivnost stranice.

Slijedi primjer dodavanja CSS atributa HTML-u kako bi poprimio određena svojstva.

HTML:

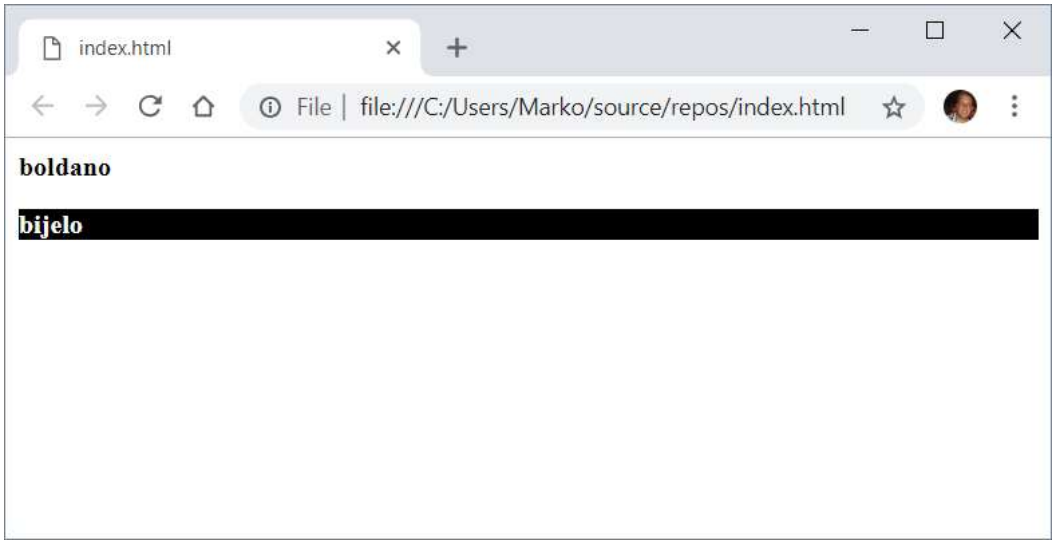
```
<div>boldano</div>

<p class="bijelo">bijelo</p>
```

CSS:

```
div {
    font-weight: bold;
}
.bijelo {
    color: white;
    background-color: black;
}
```

Tekst u svim *divovima* bi trebao biti podebljan, a u klasi ".bijelo" bijel sa crnom pozadinom. Rezultat je prikazan u slici 3.2.

Slika 3.2 Primjer korištenja CSS-a



Izvor: vlastiti rad autora

3.3. *Grid layout*

Tablice su korištene za raspored elemenata stranice dugi niz godina. Pojavom pametnih uređaja ispostavilo se da su nespretne za prikaz na manjim ekranima, a redizajn je zadavao prave glavobolje (Meloni, 2015.). Korištenje tablica za raspored

izbačeno je iz upotrebe, a umjesto njih su se počeli koristiti *float*[3] i *positioning*[4]. To nije njihova prava namjena i njihova masovna upotreba gotovo da je stvorila više problema nego ih je riješila. Na velikim uređajima *float* i *positioning* su izgledali i djelovali sasvim solidno, ali na mobilnim uređajima su elementi s tim atributima redovito kvarili korisničko iskustvo jer su se prikazivali jedan preko drugoga ili su se gurali međusobno s ekrana.

Grid layout (u nastavku: grid) iz temelja je promijenio način izrade stranica. Grid se koristi za izradu redova i stupaca u koje se potom stavlja sadržaj stranice. Ti redovi i stupci čine kostur stranice. Jednostavnost i prilagodljivost u izradi korisničkog sučelja su mu glavne vrline. Iako zvuči dosta rigidno, ako se za veličine redova i stupaca ne koriste fiksne znamenke, zadane pikselima[5] ili točkama[6], nego se koriste postotci koji se prilagođavaju veličini ekrana, grid postaje vrlo fluidan i fleksibilan element stranice. Grid je puno noviji koncept od svega do sad spomenutog ali od ožujka 2017. godine većina internetskih preglednika ima podršku za CSS grid (slika 3.3).

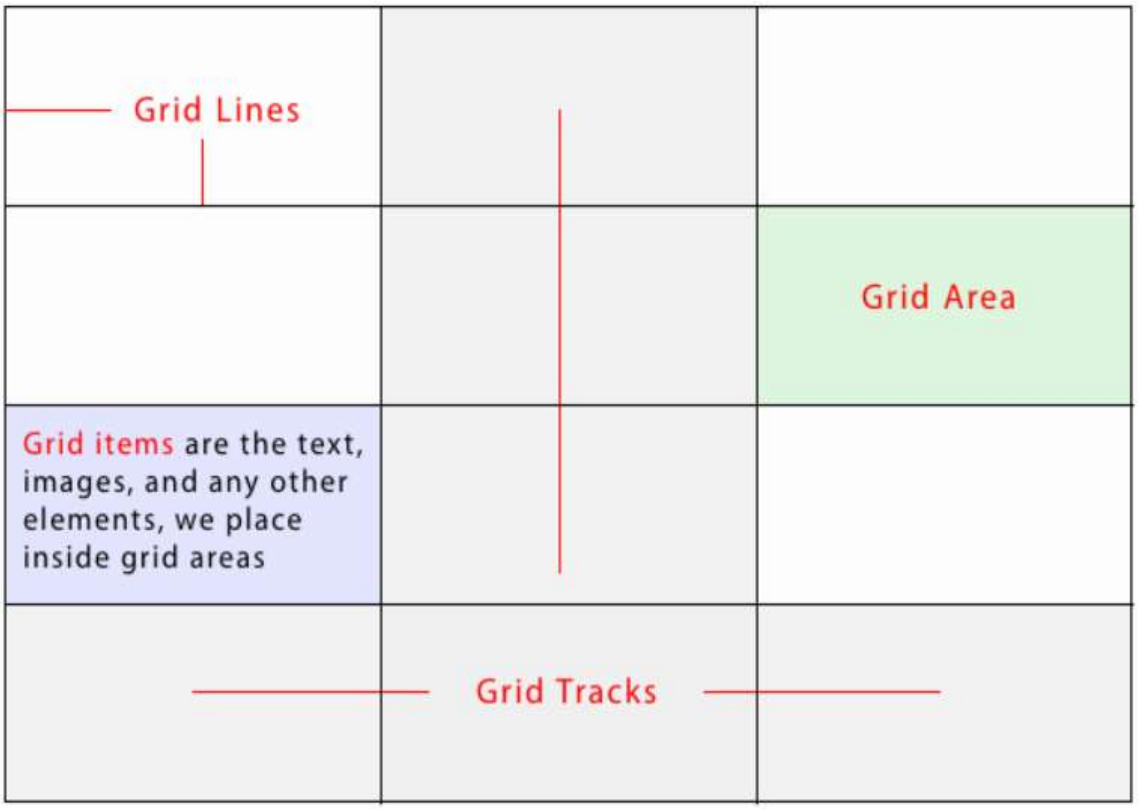
Grid je jednostavan za korištenje. Za početak, treba odabrati neki element i u CSS-u mu dodijeliti atribut {display: grid}. Time taj element postaje spremnik (eng. *grid container*) i sve što se radi s gridom, odvija se unutar njega. Spremnik ima osnovne elemente:

- *grid items* - "djeca" (eng. *child elements*) spremnika. Jedan *item* početno odgovara jednoj ćeliji grida ali mu se može zadati da zauzima više ćelija
- *grid cell* - najmanja jedinica grida
- *grid lines* - linije koje horizontalno odnosno vertikalno razdvajaju redove i stupce grida
- *grid track* - prostor između dvije linije grida. Može ga se gledati kao red/stupac grida
- *grid area* - prostor omeđen s četiri linije grida

---

[3] eng. *float* = atribut CSS-a koji služi za smještanje elementa u prostoru stranice
[4] eng. *positioning* = atribut CSS-a koji služi za poravnavanje elemenata na stranici
[5] eng. *pixel* ili *px* = najmanja točka ekrana
[6] točka (eng. *point* ili *pt*) je tipografska veličina za mjeru i iznosi 1/72 inča

Slika 3.3 Prikaz grid elemenata



Izvor: https://vanseodesign.com/css/grid-layout-module/ (26.11.2018.)

3.4. *Flexbox layout*

Nedvojbeno je da bi Marcotte uvrstio i *Flexbox* na svoj popis ključnih elemenata responzivnog dizajna, međutim prva verzija specifikacije je napisana tek 2012. godine.

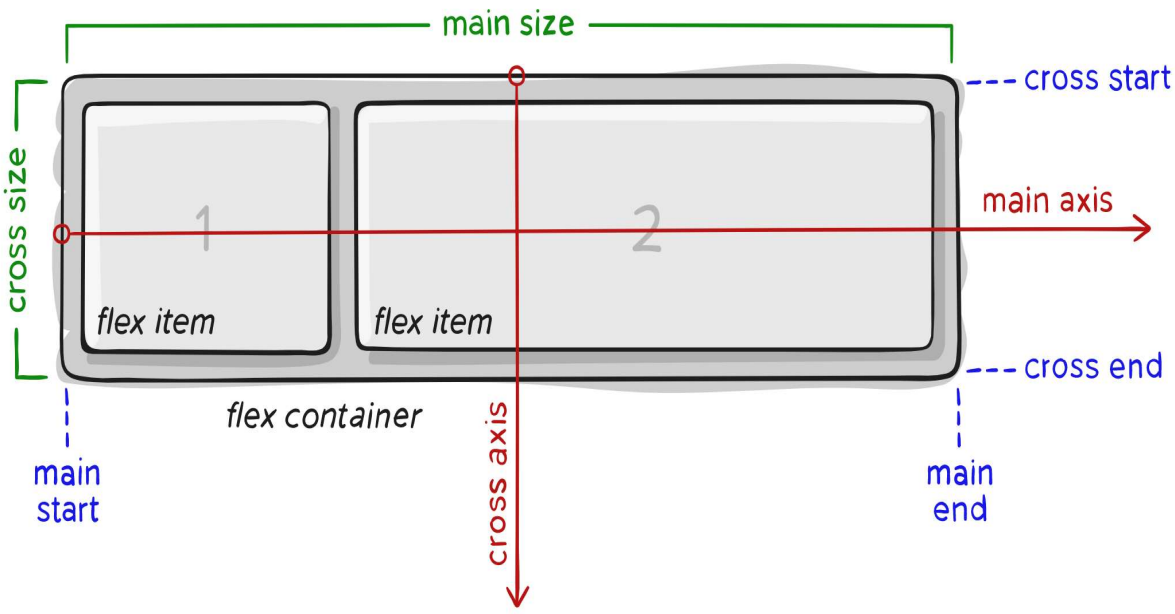
Flexbox layout (u nastavku: flexbox) se, kao i grid, koristi za izradu rasporeda elemenata stranice. Koje su razlike među njima i kako se odlučiti koji koristiti? Poanta je upravo u tome da dizajner stranice ne mora odlučivati za jedan od njih jer se, u pravilu, koriste oba. Naime, iako i grid i flexbox služe istoj svrsi, njihovo korištenje i namjena su različiti. Kombinacija najčešće izgleda ovako: prvo se koristi grid kako bi se izradio kostur - redovi i stupci stranice koji bi bili zaglavlje, tijelo i podnožje stranice,

a tada bi se pojedinim *grid item* elementima davali flexbox atributi za bolji i praktičniji raspored unutar tog elementa.

Kako grid ima spremnik i u njemu *grid item* elemente, tako i flexbox ima spremnik i u njemu *flex item* elemente (slika 3.4). Ključna razlika između grida i flexboxa jest to da je grid dvodimenzionalan (ima i redove i stupce) a flexbox je jednodimenzionalan, jer sve elemente smješta ili horizontalno u red ili vertikalno u stupac. Flexbox će popuniti cijeli red/stupac sa svim *flex item* elementima, a ako mu zadamo atribut *flex wrap*, preći će u novi red/stupac i nastaviti popunjavati sve dok ne smjesti sve elemente. Stavljaju li se elementi u red ili stupac te u kojem smjeru, određuje atribut *flex direction*, a kako bi se odredio redoslijed elemenata, pridodaje im se atribut *flex order*.

Ovo su bili najosnovniji atributi flexboxa, koji se gotovo uvijek koriste bez obzira na veličinu ili kompleksnost spremnika. Kako raste složenost spremnika i elemenata u njemu, tako je potrebno uvoditi dodatne atribute, koji će određivati veličinu elemenata, razmak između njih, je li njihovo poravnavanje ulijevo, udesno ili centralno i još nekolicinu drugih.

Slika 3.4 Prikaz flexbox elemenata

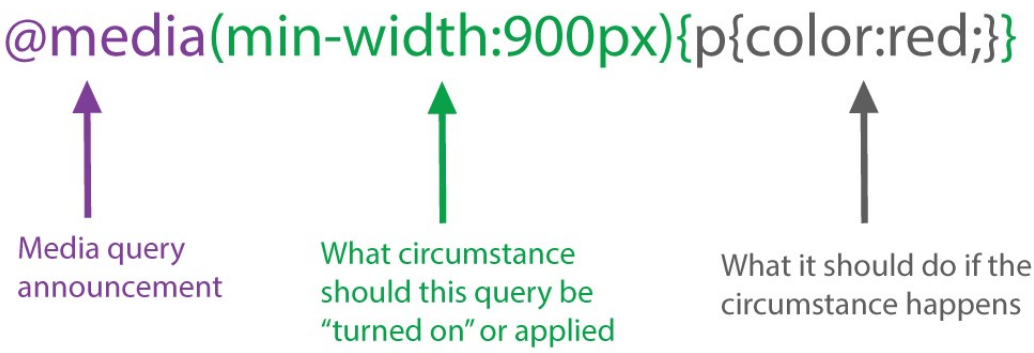


Izvor: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (9.12.2018.)

3.5. *Media queries*

Media queries su, jednostavno rečeno, upiti o nekom svojstvu medija na kojem se prikazuje internetska stranica. Kako bi zadali neko pravilo, koristi se sintaksa *@media*, nakon kojeg se u obične zagrade stavlja pravilo, a potom u šiljate zagrade niz atributa koji će biti pridodani elementu ukoliko je to pravilo ispunjeno (slika 3.5). Iako se mogu pisati i u HTML-u, *media queries* se gotovo uvijek pišu u CSS datoteku, radi lakšeg održavanja i naknadnih promjena pravila. Kompleksnije stranice, upravo radi lakšeg održavanja, imaju nekoliko CSS datoteka koje se pozivaju u HTML-u, a svaka datoteka sadrži pravila za određenu veličinu ekrana, pa se u HTML-u prilikom pozivanja CSS-a koristi *@media* pravilo da bi se znalo koji CSS ispunjava to pravilo i koji atributi se trebaju pridodati elementima stranice.

Slika 3.5 Primjer media query upita



Izvor: https://varvy.com/mobile/media-queries.html (22.12.2018.)

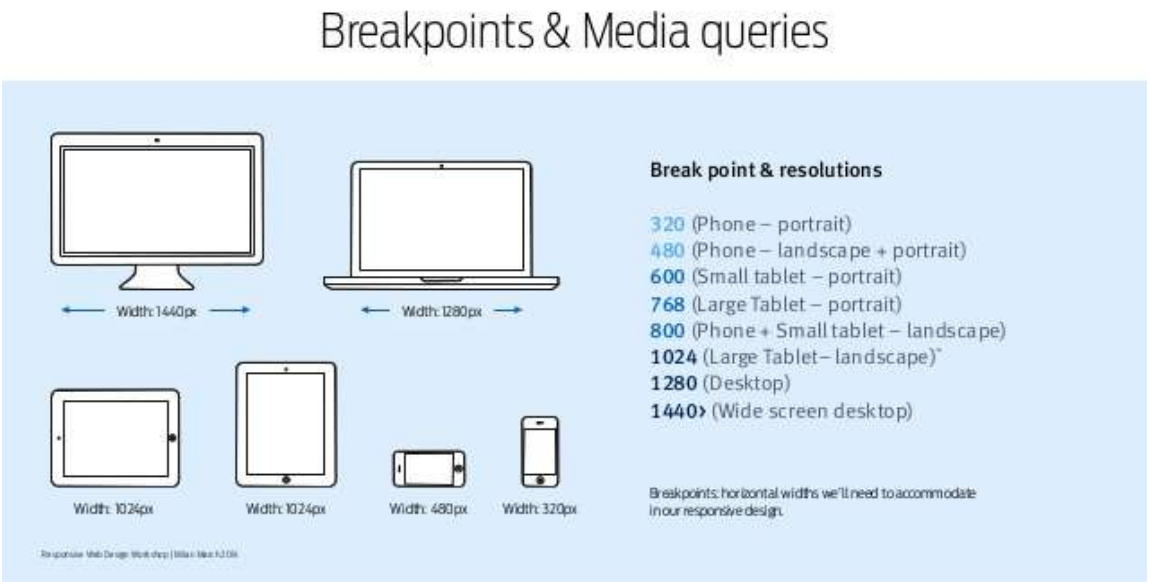
Postoje 2 podjele pravila: prema namjeni i prema svojstvu. Prema namjeni pravila se mogu odnositi na fizičke ekrane uređaja (*screen*), na pregled za printanje sadržaja (*print*) ili na specijalne uređaje koji čitaju sadržaj internetskih stranica slijepim i slabovidnim osobama (*speech*). Ako se ne odabere jedno od ovih posebnih pravila, tada će se odabrani atributi primjenjivati na sve.

Širina ekrana je samo jedno od brojnih svojstava koja se koriste za zadavanje pravila. Moguće ih je zadati preko dvadeset različitih, međutim u praksi se najviše koristi ovih pet: širina, visina, omjer stranica, rezolucija te orijentacija ekrana.

Media queries se koriste kako bi se identičan sadržaj prikazivao drugačije na različitim uređajima (Slika 3.6), ovisno o pravilu: ako stranica u desktop veličini ima stupac koji zauzima širinu pola stranice, a htjeli bismo da u prikazu na mobitelu zauzima cijelu širinu ekrana, to se vrlo jednostavno postiže zadajući atribut da stupac bude širine 100 % ako je zadovoljeno pravilo da veličina ekrana odgovara onoj na mobitelu.

Flexbox je CSS modul za koji, u načelu, nije potrebno koristiti *media queries* jer je sam po sebi već responzivan i svi elementi koji imaju *flex* atribute ponašati će se prema očekivanjima neovisno o širini, visini ili rezoluciji ekrana na kojem se prikazuju. Promjena veličine *flex item* elementima je jedan od mogućih razloga za kombiniranje flexboxa i *media* upita, ali samo u rijetkim situacijama, na primjer kada je početna veličina *flex item* elementa toliko velika da čak i kada bi u jednom redu bio samo jedan element, njegova širina bi svejedno prelazila širinu ekrana i moralo bi se pomicati stranicu lijevo-desno kako bi se vidio kompletan sadržaj tog elementa.

Slika 3.6 Primjer najčešće korištenih rezolucija u media query upitima



Izvor: https://www.onlinedesignteacher.com/2015/01/css3-media-queries-for-responsive_81.html (10.12.2018.)

3.6. Responzivne slike

Kako bi se postiglo da sve slike na nekoj stranici budu responzivne, dovoljno je u CSS-u napisati sljedeće:

```
img{
    width:100%;
    height:auto;
}
```

Ovakvi atributi na *img*[7] elementu za rezultat imaju to da svaka slika zauzme 100 % dostupne širine, a zatim da namjesti visinu tako da se ne mijenja omjer slike. Preostaje još samo staviti sliku u neki element koji ima zadane dimenzije, na primjer *flex* ili *grid item*, i slika će ispuniti koliko može taj *item* element i kako se mijenja njegova širina, tako će se i slika prilagođavati da ispuni 100 % dostupne joj širine. Naravno, ovo je najjednostavnija implementacija za postizanje responzivnosti slika na internetskoj stranici.

Slike su danas jako velike, čak i po nekoliko desetaka megabajta[8]. Ako se izrađuje stranica kojoj će većina sadržaja biti slike, poput galerije, tada se može umjesto jedne slike pripremiti nekoliko verzija iste slike i, ovisno o veličini medija, preglednik prikazuje samo onu sliku koja mu odgovara. Tako jedna slika koja inače ima 20 MB i služi prikazu na stolnom računalu, može imati verziju od 5 MB namijenjenu prikazu na mobitelu. To smanjuje količinu internetskog prometa koji mora potrošiti korisnik i istovremeno se ubrzava učitavanje stranice. Ovakav proces postizanja responzivnosti je dugotrajan i težak za održavati, tako da ga većina dizajnera koristi samo ako je to apsolutno nužno. Kada je god moguće, koristiti će se jednostavnija, prva spomenuta metoda.

---

[7] *img* je HTML oznaka za sliku (skraćeno od *image)*
[8] megabajt (skraćeno MB) je mjera količine informacija u memoriji računala i iznosi milijun bajtova

4. IZRADA RESPONZIVNE INTERNETSKE STRANICE

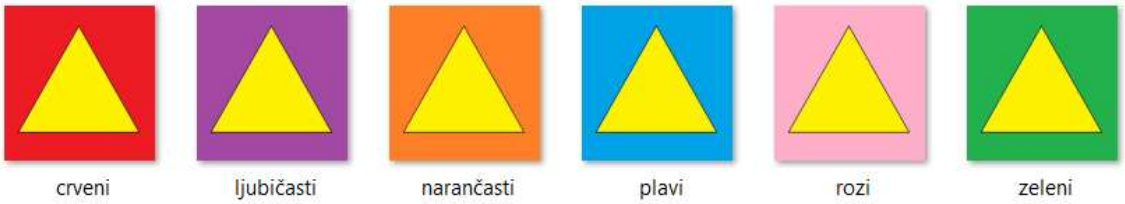
U ovom poglavlju će se demonstrirati primjena do sad spomenutih metoda i alata. Autor rada će korak po korak izrađivati stranice i opisivati što se i kako radi. Biti će napravljene dvije povezane stranice, jedna će biti primjer stranice za prijavu (eng. *login*), a druga, će biti primjer galerije slika koje će biti responzivne. Stranice će služiti isključivo za demonstraciju responzivnog dizajna i ni na koji drugi način neće biti funkcionalne.

Sav napisani kod je vlastiti rad autora. Kompletan kod nalazi se u prilozima na kraju rada, a u ovom poglavlju biti će prikazani najvažniji isječci. Radi jednostavnosti demonstracije, svaka od dvije HTML stranice će imati svoju zasebnu CSS datoteku i njihova nomenklatura izgleda ovako:

- login.html
- login.css
- galerija.html
- galerija.css

Za potrebe prikaza responzivnih slika, autor je izradio 6 jednostavnih slika u programu *MS Paint*. Sve slike su veličine 1000 x 1000 piksela i prikazuju žuti trokut. Nalaze se u mapi "img". Svaka slika oko trokuta ima drugačiju boju kako bi se razlikovale, kao što je prikazano na slici 4.1.

Slika 4.1 Slike koje se koriste za izradu galerije



Izvor: vlastiti rad autora

4.1. Stranica za prijavu

Na stranici za prijavu pokazat će se kako je flexbox vrlo jednostavan i praktičan alat koji i bez *media query* upita može postići responzivnost sadržaja na stranici.

Prvo će se u *body* elementu definirati *div* kojem će se dodati klasa "flex-container". Taj *div* postaje flexbox spremnik. Zatim se dodaje novi *div* koji će biti *flex item*, i to će biti jedini *flex item* u spremniku. Nazivi klasa su sasvim proizvoljni, ali je poželjno nazivati ih po nekoj svrsi koju obavljaju kako bi bilo lakše održavati kod. *Flex itemu* dodaju se dva *diva*, jedan u kojem je proizvoljni tekst, a u drugom je forma za prijavu. Forma se sastoji od tri elementa: polja za korisničko ime, polja za lozinku te gumba za prijavu. Polja za korisničko ime i lozinku imaju *placeholdere*[9] koji upućuju korisnika što treba unijeti. Klik na gumb za prijavu vodi na stranicu galerije. Elementima unutar *flex item diva* dodana je klasa "margin" samo kako bi u CSS-u dodali marginu, da elementi ne bi bili zalijepljeni jedan za drugoga. Opisani HTML izgleda ovako:

```
<body>
   <div class="flex-container">
      <div class="flex-item">
         <div class="margin"> Dobrodošli na moje stranice </div>
         <div>
            <input class="margin" type="text" placeholder="Korisničko ime"><br />
            <input class="margin" type="password" placeholder="Lozinka"><br />
            <button class="margin" type="submit"
onclick="window.location.href='galerija.html'">Prijava</button>
…
```

[9] *Placeholder* je tekst u polju za unos koji upućuje korisnika što treba unijeti

CSS započinje tako da se prvo dodaju stilovi za *body* i *html* oznake:

```
body, html {
    height: 100vh;
    margin: 0;
}
```

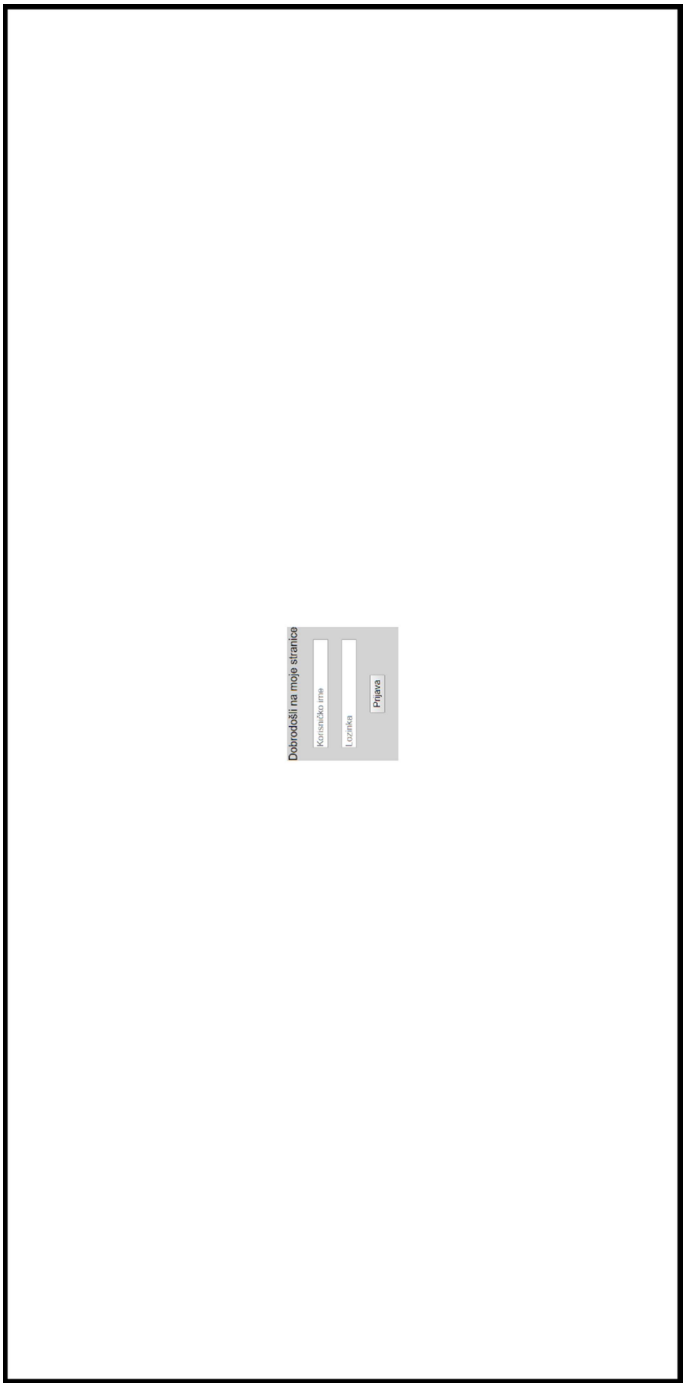
Vh (eng. *viewport height*) je mjerna jedinica za visinu i 1 vh zauzima 1 % visine ekrana kojeg korisnik vidi. Za visinu je postavljeno 100 vh, što znači da *body* mora zauzeti cijelu visinu ekrana. Margina na 0 miče bilo kakvu predefiniranu marginu na *body* elementu. Slijedi dodavanje atributa klasi "flex-container", koja je u *divu* neposredno nakon *bodyja*:

```
.flex-container {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

Visina na 100 % definira da će ovaj *div* zauzeti sav mogući prostor koji mu je omogućio element iznad njega, a pošto je iznad njega *body* koji zauzima cijeli ekran, i on će zauzeti cijeli ekran. Ovo je vrlo zgodan trik da se *div* raširi na cijeli ekran, a da se ne dodaju atributi izravno na *body* element, kojeg se uvijek treba truditi držati što čišćim. Atribut "display: flex" proglašava *div* flexbox spremnikom, a zadnja dva atributa poravnavaju *flex item* elemente unutar spremnika: prvi će ih poravnati na sredinu spremnika vodoravno, a drugi okomito. Ova 4 atributa u potpunosti uklanjaju potrebu za *media query* upitom o rezoluciji ekrana jer će forma za prijavu uvijek biti točno na sredini ekrana, bez obzira na njegovu rezoluciju ili orijentaciju.

Sama forma za prijavu ima "flex-item" klasu kojoj je dodana pozadinska boja zato da se vizualno demonstrira koju površinu će zauzeti ako joj nisu zadane širina i visina. Cijeli spremnik ima samo jedan *flex item*, tako da nije potrebno dodavati atribute kako bi se *flex itemi* poravnavali u redove ili stupce te prebacivali u novi red ako ne stanu svi u jedan. Izgled stranice na stolnom računalu prikazan je na slici 4.2.

Slika 4.2 Izgled stranice za prijavu na ekranu stolnog računala (okrenuti vodoravno)



Izvor: vlastiti rad autora

Izgled stranice na mobitelu prikazan je na slici 4.3.

Slika 4.3 Izgled stranice za prijavu na ekranu mobitela

Izvor: vlastiti rad autora

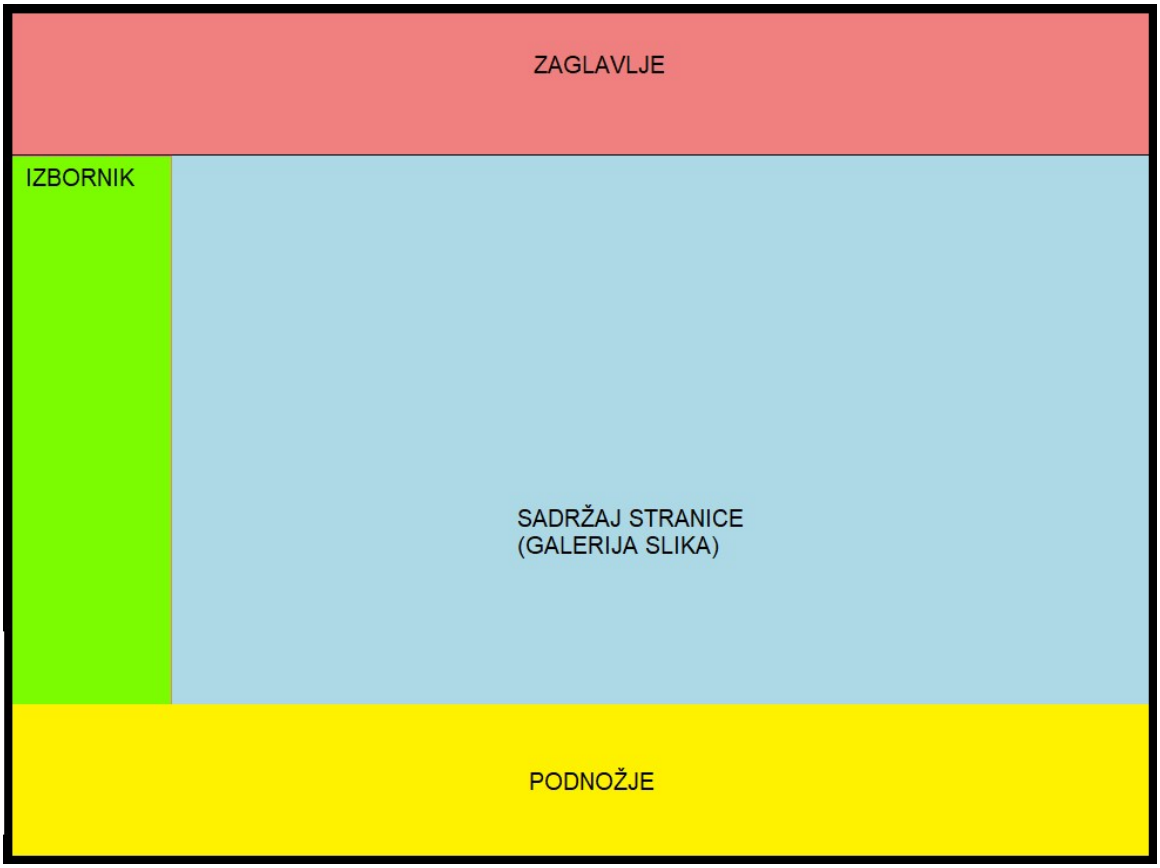
U manje od 20 redaka CSS-a postignut je cilj: u potpunosti responzivna stranica za prijavu, neovisno o uređaju na kojem se prikazuje. Prikazana forma za prijavu je uvijek i vodoravno i okomito centrirana na stranici, a pošto joj nisu zadane širina i visina, pomoću pozadinske sive boje može se vidjeti da zauzima samo onoliko prostora koliko joj je potrebno.

4.2. Stranica s galerijom slika

Stranica s galerijom slika će biti dosta složenija od stranice za prijavu. Za izradu će se koristiti svi opisani alati i metode. Prvo će pomoću grida biti postavljen kostur stranice, a zatim će se koristiti flexbox za samu galeriju slika. Sve slike će biti responzivne, a pomoću *media query* upita će se mijenjati ponašanje nekih elemenata s obzirom na rezoluciju ekrana. Pristup će biti "prvo-desktop" (eng. *desktop-first*) a zatim će se dodati atributi za optimizirani prikaz na mobitelu. Kao prijelomnu točku za *media query* upit autor je odabrao 800 piksela jer će time osigurati da svi mobiteli i većina tableta, bez obzira na orijentaciju ekrana, imaju sadržaj prikazan na jedan način, dok će korisnici na stolnim računalima imati malo drugačije, bogatije iskustvo.

Kod kompleksnijih stranica poput ove, poželjno je prvo nacrtati izgled stranice i smještaj elemenata na stranici, radi lakše izrade. Iako danas postoje programi isključivo za tu namjenu, crtanje izgleda stranice rukom još je uvijek najpopularniji izbor. Što je kompleksnija stranica, to je crtanje izgleda stranice potrebnije, a posebno je potrebno kod izrade velikog niza stranica koje imaju isti generalni uzorak, na primjer isto zaglavlje i podnožje, a mijenja se samo sadržaj u glavnom dijelu stranice. Za crtanje izgleda stranice s galerijom slika korišten je *MS Paint*. Izgled stranice na stolnom računalu prikazan je slici 4.4.

Slika 4.4 Izgled stranice na stolnom računalu



Izvor: vlastiti rad autora

Na mobitelu, stranica bi trebala izgledati kao na slici 4.5.

Slika 4.5 Izgled stranice na mobitelu

Izvor: vlastiti rad autora

Na ovim slikama se jasno mogu vidjeti elementi grida - redovi i stupci koji će biti kostur ove stranice. Na mobitelu stranica neće imati bočni izbornik, a to skrivanje ćemo postići *media query* upitom. Sada kada je definiran izgled stranice, može se krenuti s njezinom izradom. Prvo se *bodyju* dodaje jedan *div* s klasom "grid-container", a zatim se unutar tog *diva* definiraju svi najvažniji elementi stranice:

- zaglavlje (eng. *header*)
- bočna traka (eng. *aside*)
- sadržaj (eng. *main*)
- podnožje (eng. *footer*)

Kod u *body* elementu HTML-a izgleda ovako:

```
<body>
    <div class="grid-container">
        <header>
            <div>ZAGLAVLJE</div>
        </header>
        <aside>
            <div>IZBORNIK</div>
        </aside>
        <main>
            <div>SADRŽAJ - galerija slika</div>
        </main>
        <footer>PODNOŽJE</footer>
    </div>
</body>
```

Ovo nije konačan HTML, ali je dovoljan da se u potpunosti definira grid. Sada se u CSS-u dodaje klasa "grid-container" s atributima:

```
.grid-container {
    display: grid;
    grid-template-columns: repeat(12, 1fr);
    grid-template-rows: 100px 100% 100px;
}
```

Atributom "display: grid" *div* je proglašen grid spremnikom. Atribut za broj stupaca i njihovu širinu ima vrijednost "repeat(12, 1fr)" - grid ima 12 stupaca i svaki je širok 1 fr. 1 fr (eng. fractional unit) je 1 dio slobodnog prostora. To znači da će 12 stupaca ravnomjerno rasporediti po slobodnom prostoru. Ako je slobodan prostor širok 120 piksela, svaki stupac će širok 10 piksela, a ako je slobodno 1200 piksela, stupci će biti po 120 piksela široki. To čini mjernu jedinicu fr izrazito fleksibilnom i responzivnom. Staviti baš 12 stupaca je nepisano pravilo. S 12 je lagano računati a opet je dovoljno velik broj stupaca da pokrije puno *grid itema*.

Atribut za broj i visinu reda ima vrijednost "100px 100% 100px". To znači da grid ima tri reda, od kojih prvi i zadnji imaju točno po 100 piksela, a srednji je responzivan i zauzeti će onoliko prostora koliko mu treba. Prvi i zadnji red su zaglavlje i podnožje stranice, a srednji red su bočna traka i sadržaj. Oni moraju biti responzivni kako bi bilo dovoljno mjesta za sve slike. Tek kada je postavljen grid, može se elementima definirati koliko stupaca da zauzmu:

```
header {grid-column: span 12;}
aside {grid-column: span 2;}
main {grid-column: span 10;}
footer { -column: span 12;}
```

Zaglavlje i podnožje stranice zauzeti će svih 12 stupaca, bočna traka 2, a sadržaj ostalih 10. Ovime je u završen grid, barem što se tiče prikaza na stolnom računalu. Stranica s galerijom slika u zaglavlju, podnožju i bočnoj traci neće imati nikakav sadržaj jer bi to izlazilo iz okvira ovog rada. Preostaje samo dodati galeriju slika i učiniti i galeriju i slike responzivnim.

Da bi se dodala galerija slika, potrebno je vratiti se u HTML i pronaći ovaj dio koda:

```
<main>
        <div>SADRŽAJ - galerija slika</div>
</main>
```

Taj element je do sada isključivo bio *grid item*, a sada će biti pretvoren u flexbox u kojem će *flex item* elementi biti responzivne slike. U *main* element se dodaje šest *divova* za šest slika koje su pokazane na slici 4.1. Svaki *div* na sebi treba imati klasu "flex-item", div u kojem je naziv slike i samu sliku. *Main* sada izgleda ovako:

```
<main>
        <div class="flex-item">
                <div>Crvena slika</div>
                <img src="img/crveni.jpg" />
        </div>
```

```
        <div class="flex-item">
            <div>Plava slika</div>
            <img src="img/plavi.jpg" />
        </div>
        <div class="flex-item">
            <div>Zelena slika</div>
            <img src="img/zeleni.jpg" />
        </div>
        <div class="flex-item">
            <div>Narančasta slika</div>
            <img src="img/narančasti.jpg" />
        </div>
        <div class="flex-item">
            <div>Roza slika</div>
            <img src="img/rozi.jpg" />
        </div>
        <div class="flex-item">
            <div>Ljubičasta slika</div>
            <img src="img/ljubičasti.jpg" />
        </div>
</main>
```

HTML kod je završen u potpunosti i više ga se neće mijenjati. U CSS se mogu dodati atributi za responzivne slike:

```
img{
    width: 100%;
    height: auto;
}
```

Već je objašnjeno da se ovime slika uvijek prilagodi veličini nadređenog joj elementa. U galeriji na ovoj stranici nadređen joj je *div* element s klasom "flex-item" tako da će svaka slika preuzimati vrijednost za veličinu iz atributa na toj klasi.

Tamo gdje je elementu *main* definirano da zauzme 10 stupaca treba dopisati atribute za flexbox:

```
display: flex;
flex-flow: row wrap;
justify-content: space-evenly;
align-items: center;
```

Atribut "flex-flow: row wrap" će složiti sve *flex item* elemente u red, a ako ne stanu svi u jedan red, prijeći će u novi, dok će atribut "justify-content: space-evenly" osigurati jednake razmake lijevo i desno među *flex item* elementima.

Autor je želi da galerija na stolnom računalu uvijek ima tri slike u jednom redu. To se postiže na ovaj način:

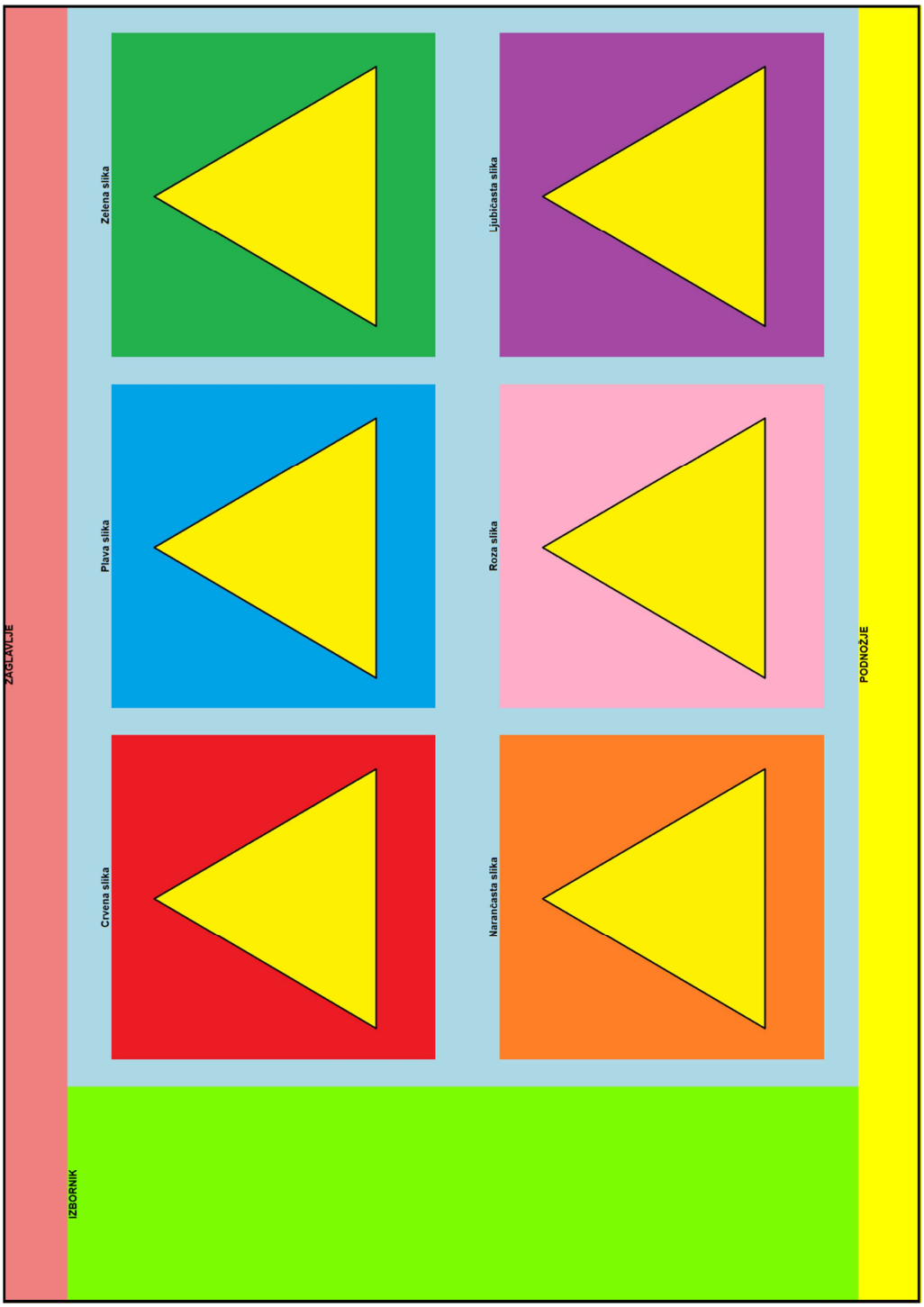
```
.flex-item{
    width: 30%;
}
```

Kako to funkcionira? Širina od maksimalnih 100 % podijeli se na tri jednaka dijela. Rezultat je 33.33 %. Ako bi se to uzelo kao vrijednost, sve slike bi se međusobno doticale jer ne bi bilo razmaka među njima. Zato je praktičnije taj rezultat zaokružiti na 30 % a preostalih 10 % pustiti da se potroši na razmake između slikama, koji će se radi atributa "space-evenly" automatski podjednako rasporediti. Kada dođe četvrti *flex item* na red za prikazivanje, preglednik će vidjeti da mu 10 % nije dovoljno mjesta pa će ga prebaciti u novi red. Na ovaj način će svaki red spremnika imati uvijek točno tri *flex item* elementa, neovisno o veličini ekrana, i njihova veličina će se mijenjati promjenom rezolucije ekrana tako da uvijek zauzima 30 %. Tako će galerija široka 1000 piksela imati tri slike širine po 300 piksela i 100 piksela će se rasporediti na razmake među njima i mijenjanje širine galerije će automatski mijenjati i širine slika.

Da bi stranica bila u potpunosti spremna za prikaz na stolnom računalu, potrebno je još samo kao i na stranici za prijavu dodati atribut "margin: 0" na *body* element, kako bi poništili unaprijed zadanu marginu od 8 piksela. Atribut za visinu nema smisla dodavati jer si svaki element ionako sam zauzima onoliko prostora koliko mu treba, pa

bi vrijednost za visinu tog elementa „pregazila“ vrijednost visine *bodyja*. Još su samo dodane boje *grid item* elementima da ih se može vizualno razlikovati i verzija stranice za stolna računala je spremna. Rezultat je prikazan na slici 4.6.

Slika 4.6 Izgled stranice s galerijom slika na stolnom računalu (okrenuti vodoravno)



Izvor: vlastiti rad autora

Preostalo je još samo dodati *media query* upit kako bi se stranica prilagodila prikazu na mobilnom uređaju. Pomoću upita treba sakriti bočnu traku i postaviti galeriju da zauzima 12 umjesto 10 stupaca jer će skrivanje bočne trake osloboditi 2 stupca. Osim toga, postavit će se širina *flex item* elemenata na 200 piksela. Cijeli *media query* upit izgleda ovako:

```
@media (max-width:800px) {
    aside {
        display:none;
    }
    main {
        grid-column: span 12;
    }
    .flex-item{
        width: 200px;
    }
}
```

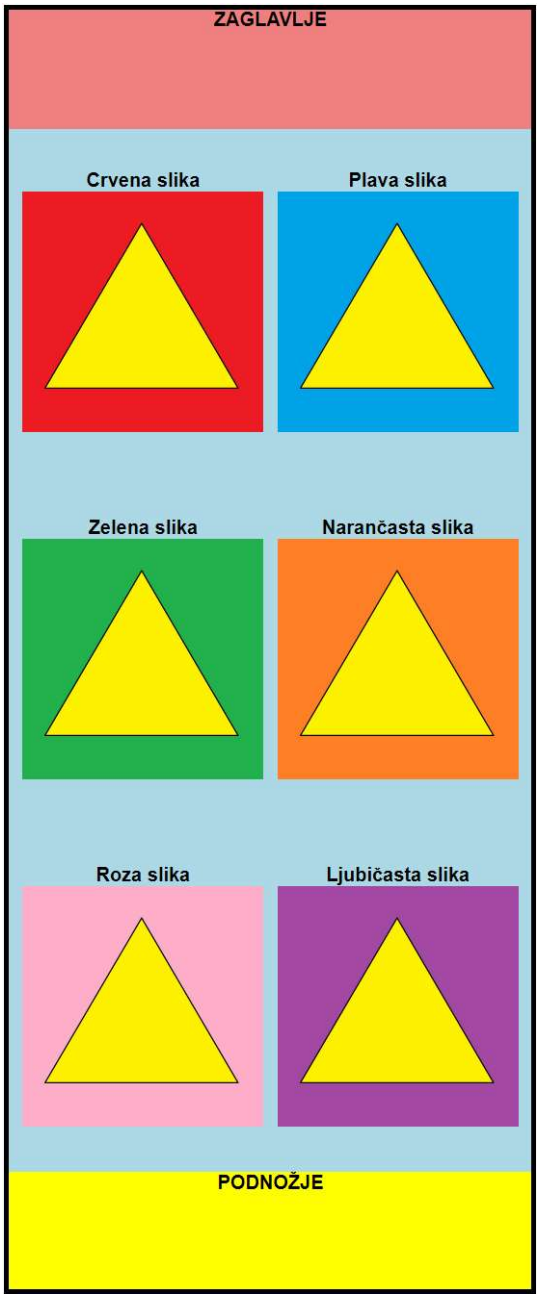
Flex item element se postavlja na fiksnih 200 piksela zato što bi širina od 30 % ekrana bila premala na manjim uređajima. Na primjer, iPhone 4[10] ima ekran širok 5 centimetara pa bi slika koja zauzima 30 % ekrana imala samo 1.5 centimetara, što je jako sitno. Postavljanje širine na 200 piksela omogućuje korisniku da uvijek vidi sliku u jednakoj veličini na manjim uređajima, a flexbox će se pobrinuti za broj slika u jednom redu: ako je ekran jako mali, biti će samo jedna, a ako se radi o tabletu, bit će ih najviše tri.

Ovaj jednostavan *media query* upit se pobrinuo za to da se ne mora izrađivati još jedna stranica samo za tablete i mobitele. Ušteda je već vidljiva u vremenu izrade, a dugoročne uštede vidjele bi se u troškovima i vremenu održavanja stranice. Stranica

[10] iPhone je popularan mobilni uređaj tvrtke Apple

je sada u potpunosti završena, a izgled stranice na mobilnom uređaju prosječne širine od otprilike 450 piksela vidi se na slici 4.7.

Slika 4.7 Izgled stranice s galerijom slika na mobilnom uređaju



Izvor: vlastiti rad autora

5. ZAKLJUČAK

U ovom radu su kroz teorijski i praktični dio objašnjene sve najbitnije metode i alati za samostalnu izradu responzivnih internetskih stranica. Kroz teorijski dio detaljno su opisani HTML, CSS, grid, flexbox, *media query* upiti te responzivne slike, a u praktičnom dijelu je demonstrirana upotreba istih za izradu dvije responzivne stanice potpuno različitih zahtjeva, parametara i kompleksnosti.

U dvjema izrađenim stranicama pokazano je kako svaki od tih alata i metoda sam za sebe ne može postići da stranica bude responzivna, odnosno da dinamički mijenja izgled ovisno o tome prikazuje li se na mobitelu ili na velikom monitoru. Željeni rezultat se može postići samo pažljivom i ispravnom kombinacijom svih elemenata. Ispravna kombinacija ne znači da postoji samo jedna i kod napisan za potrebe izrade ove dvije stranice nije jedini koji bi dao isti ovakav rezultat. Na primjer, moglo se raditi prvo za prikaz na mobitelu pa kasnije prilagoditi stolnom računalu. CSS bi bio u potpunosti drugačiji, ali rezultat bi bio isti. Zato su za samostalnu izradu ovakvih stranica potrebna određena predznanja i dosta pokušaja i pogreški.

Svaki od alata ima svoje prednosti i mane, pa se ponekad mora pisati dodatni kod da se te mane isprave. Neke stvari su sigurne: *media queries* odlično rade svoj posao a jednostavni su za implementaciju a grid i flexbox imaju najviše iskoristivih atributa i zato treba biti oprezan jer će i najmanja greška sve pokvariti. Što se tiče responzivnih slika, sigurno je da ni u jednom trenutku ni na jednom ekranu nije bila slika u nativnoj rezoluciji od 1000 x 1000 piksela, nego su se prilagodile spremniku u kojem su bile, tako ona dva jednostavna atributa zapamtiti i koristiti kada kod je to moguće.

Kod napisan za izradu ovih stranica je iskoristiv i za druge svrhe osim ovih pokazanih. Koristeći dijelove tog koda i teorijska znanja napisana u ovom radu svatko uz malo truda i mašte može napraviti svoju vlastitu responzivnu stranicu, prema vlastitim željama i potrebama.

LITERATURA

Knjige

Marcotte, E. (2011.) *Responsive Web Design*. Preuzeto s https://www.reprosol.be/sites/reprosol.beta.the-aim.be/files/responsive-webdesign(ethan-marcotte).pdf (22.12.2018.)

Meloni, Julie C. (2015.) *Sams Teach Yourself HTML, CSS and JavaScript All in One, Second Edition.* Hoboken, NJ: Pearson Education, Informit

Internetski izvori

HTML: preuzeto s https://www.w3schools.com/html/ (16.12.2018.)

CSS: preuzeto s https://www.w3schools.com/css/ (17.12.2018.)

Grid: preuzeto s https://css-tricks.com/snippets/css/complete-guide-grid/ (22.12.2018.)

Flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (22.12.2018.)

Media queries: https://www.w3schools.com/css/css_rwd_mediaqueries.asp (18.12.2018.)

POPIS SLIKA

PRILOZI

PRILOG 1: Kod stranice *login.html*

PRILOG 2: Kod stranice *login.css*

PRILOG 3: Kod stranice *galerija.html*

PRILOG 4: Kod stranice *galerija.css*

PRILOG 1: Kod stranice *login.html*

```
 <!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <link rel="stylesheet" type="text/css" href="login.css" />
   <title>Login</title>
</head>
<body>
   <div class="flex-container">
      <div class="flex-item">
         <div class="margin">
            Dobrodošli na moje stranice
         </div>
         <div>
            <input class="margin" type="text" placeholder="Korisničko ime"><br />
            <input class="margin" type="password" placeholder="Lozinka"><br />
            <button class="margin" type="submit"
onclick="window.location.href='galerija.html'">Prijava</button>
         </div>
      </div>
   </div>
</body>
</html>
```

PRILOG 2: Kod stranice *login.css*

```
body, html {
    height: 100vh;
    margin: 0;
    font-family: Arial;
}
.flex-container {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}
.flex-item{
    text-align:center;
    background-color:lightgray;
}
.margin{
    margin-bottom:20px;
}
```

PRILOG 3: Kod stranice *galerija.html*

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <link rel="stylesheet" type="text/css" href="galerija.css" />
   <title>Galerija</title>
</head>
<body>
   <div class="grid-container">
      <header>
         <div>ZAGLAVLJE</div>
      </header>
      <aside>
         <div>IZBORNIK</div>
      </aside>
      <main>
         <div class="flex-item">
            <div>Crvena slika</div>
            <img src="img/crveni.jpg" />
         </div>
         <div class="flex-item">
            <div>Plava slika</div>
            <img src="img/plavi.jpg" />
         </div>
         <div class="flex-item">
            <div>Zelena slika</div>
            <img src="img/zeleni.jpg" />
         </div>
         <div class="flex-item">
            <div>Narančasta slika</div>
            <img src="img/narančasti.jpg" />
         </div>
         <div class="flex-item">
            <div>Roza slika</div>
            <img src="img/rozi.jpg" />
         </div>
         <div class="flex-item">
            <div>Ljubičasta slika</div>
            <img src="img/ljubičasti.jpg" />
         </div>
      </main>
      <footer>PODNOŽJE</footer>
   </div>
</body>
</html>
```

PRILOG 4: Kod stranice *galerija.css*

```
body {
    margin: 0;
    font-family:Arial;
    font-weight:bold;
    text-align:center;
}

/* desktop prikaz */
.grid-container {
    display: grid;
    grid-template-columns: repeat(12, 1fr);
    grid-template-rows: 100px 100% 100px;
}

header {
    grid-column: span 12;
    background-color: lightcoral;
}

aside {
    grid-column: span 2;
    background-color:lawngreen;
}

main {
    grid-column: span 10;
    background-color: lightblue;

    /*flexbox atributi*/
    display: flex;
    flex-flow: row wrap;
    justify-content: space-evenly;
    align-items: center;
}

footer {
    grid-column: span 12;
    background-color: yellow;
}

img{
    width:100%;
    height:auto;
}
```

```
.flex-item{
    width:30%;
}


/* mobilni prikaz */

@media (max-width:800px) {
    aside {
        display:none;
    }
    main {
        grid-column: span 12;
    }
    .flex-item{
        width: 200px;
    }
}
```