CSS proširenja kroz sustav LESS

Kukec, Emil

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:437513

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

University North Digital Repository

Sveučilište
Sjever

Završni rad br. 551/MM/2017

CSS proširenja kroz sustav LESS

Emil Kukec, 0303/336

Varaždin, rujan 2017. godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 551/MM/2017

CSS proširenja kroz sustav LESS

Student

Emil Kukec, 0303/336

Mentor

Vladimir Stanisavljević, mr. sc.

Varaždin, rujan 2017. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za multimediju, oblikovanje i primjenu

PRISTUPNIK: Emil Kukec | MATIČNI BROJ: 0303/336

DATUM: 1.09.2017. | KOLEGIJ: Programski alati 2

NASLOV RADA: CSS proširenja kroz sustav LESS

NASLOV RADA NA ENGL. JEZIKU: Extending CSS trough LESS preprocessor

MENTOR: mr.sc. Vladimir Stanisavljević | ZVANJE: viši predavač

ČLANOVI POVJERENSTVA:

1. doc.dr.sc. Dejan Valdec - predsjednik
2. izv.prof.dr. Mario Tomiša - član
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. dr.sc. Robert Logožar, v. pred. - zamjenski član
5.

VŽ KC

MMI

Zadatak završnog rada

BROJ: 551/MM/2017

OPIS

Tehnologija oblikovnih stilskih obrazaca (engl. Cascading Style Sheet ili skraćeno CSS) ima značajnu ulogu u oblikovanju Web stranica. Prije svega, CSS-om u odvojenoj datoteci može se ostvariti odvajanje oblikovanja od samog HTML koda. U stvarnom radu, pogotovo za veće projekte, pokazalo se da rad s CSSom ipak nije dovoljno fleksibilan. Podaci koji se koriste za parametriziranje npr. boja ili dimenzija su konstante vrijednosti i ukoliko se iste boje vrijednosti pojavljuju na više mjesta za promjenu oblikovanja treba ih ručno mijenjati na svim mjestima. Osim toga, nije moguća računska niti bilo kakva druga programska manipulacija s tim vrijednostima. Da bi se zaobišlo ta i takva ograničenja razvijeni su preprocesori CSS-a, programi koji iz datoteka napisanim korištenjm posebne sintakse stvaraju konačni CSS kod. LESS programski sustav je jedan od CSS preprocesora.

U ovom radu potrebno je:
* opisati osnove CSS-a, njegovu ulogu u oblikovanju Web stranica te analizirati njegove prednosti i ograničenja,
* opisati postupak instalacije i osnovnog korištenja LESS-a,
* detaljno obraditi mogućnosti i sintaksu LESS preprocesora i na primjerima pokazati njegovo korištenje,
* osmisliti i oblikovati demo web stranicu na kojoj će biti prikazane glavne mogućnosti, načini korištenja LESS-a i primjeri koda,
* ukratko usporediti LESS sa sličnim sustavima

Detaljno opisati sve korištene tehnologije i korake u radu potrebne da bi se ostvarilo traženo te detaljno opisati stečena iskustva i postignute rezultate.

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER
SVEUČILIŠNA KNJIŽNICA

ZADATAK URUČEN: 11.09.2017 | POTPIS MENTORA:

Predgovor

Ovaj završni rad nastao je sa željom da se proširi moje znanje u front-end developmentu gdje vidim svoju budućnost. Također se nadam da će pomoći svim studentima i programerima da steknu nova saznanja kojima će poboljšati svoj rad ili hobi.

Želim se zahvaliti svim profesorima koji su mi omogućili uspješan završetak studiranja na Sveučilištu Sjever. Posebna zahvala profesoru Vladimiru Stanisavljeviću koji me na svojim predavanjima zainteresirao za web development te mi pomogao kod pisanja završnog rada.

Sažetak

Svrha ovog završnog rada je obraditi i objasniti LESS.js te na kraju za praktični rad napraviti internetsku stranicu o LESS-u sa svrhom učenja tog CSS pred-procesora. LESS, odnosno CSS pred-procesori su nezaobilazni programski jezici u svijetu izrade internetskih stranica. Svatko bi morao početi koristiti neki od pred-procesora nakon što dobro savlada CSS. Vrlo ih je lako naučiti zbog sličnosti u sintaksi koja se ne razlikuje ili se razlikuje vrlo malo što je važno za početnike. Oni proširuju stilski jezik i daju mu razne mogućnosti poput varijabli, nestinga, operacija i funkcija koje je vrlo lako shvatiti i koristiti. Koriste ih većina front-end developera, odnosno ljudi koji izrađuju internetske stranice kako se ne bi ponavljali kod pisanja koda. LESS je poznatiji pred-procesor i vrlo zastupljeni među programerima. Praktični dio završnog rada napravljen je kako bi početnicima pomogao savladati LESS. Napravljena je internetska stranica, odnosno 6 .html datoteka koje uz definicije i primjere kodova objašnjavaju najvažnije značajke LESS pred-procesora.

Ključne riječi: LESS, CSS, pred-procesor, internetska stranica

Summary

The purpose of this final paper is to explain LESS.js and on the end of the paper make a LESS web site for the purpose of learning this CSS preprocessor. LESS, or CSS preprocessor are inevitable programming languages in the world of front-end development. Once we master CSS, everyone would have to start using some of the preprocessor. It is very easy to overcome them because of similarities with CSS in syntax that is not different or varies very little and that is very important for beginners. They extend the CSS style language and give it various features like variables, nesting, operations and functions that are easy to understand and use. They are used by most front-end developers because they don't want repeat code writing. The practical part of the final work is made to help beginners overcome the LESS. An web site or 6 .html files have been created that, along with definitions and code examples, explain the most important features of the LESS preprocessor.

Keywords: LESS, CSS, preprocessor, web site

Popis korištenih kratica

HTML HyperText Markup Language
Sintaksa za obilježavanje hipertekstualnih dokumenata.

CSS Cascading Style Sheets
Sintaksa za oblikovanje hipertekstualnih dokumenta.

SASS Syntactically Awesome Style Sheets
CSS pred-procesor.

NPM Node Package Manager
Velika kolekcija gotovih JavaScript modula.

FTP File Transfer Protocol
Standardni mrežni protokol koji se koristi za premještanje datoteka s jednog servera na drugi putem mreže temeljene na TCP-u.

Sadržaj

1. Uvod....1
2. CSS....2
   2.1. Povezivanje CSS stilova....2
   2.2. Sintaksa....3
   2.3. Selektori....3
3. CSS pred-procesori....4
4. LESS....5
   4.1. Kompajliranje LESS-a u CSS....5
   4.2. Varijable....6
   4.3. Ugniježđeni stilovi (Nested rules)....7
   4.4. Operacije....8
   4.5. Mixin....8
5. Konkurencija....12
   5.1. SASS....12
   5.2. Stylus....13
6. Praktični rad....15
   6.1. Programi korišteni kod izrade....15
   6.2. Početak rada....17
   6.3. Početna stranica....18
   6.4. Oblikovanje preostalih stranica....23
   6.5. Izrada responzivnog web dizajna....29
   6.6. Prijenos praktičnog rada na server....31
7. Zaključak....32
8. Literatura....33

1. Uvod

Stranice koje se nalaze na internetu nezamislive su bez neke vrste oblikovanja. To oblikovanje danas više nije samo radi estetike, nego nam daje i informacije. Možemo pronaći stranice sa sve više animacija i tranzicija među elementima. Video i slika su odavno standard koji se pojavljuje na internetu, a tu su i igre dizajnera sa psihologijom boja i načinom kako će te stranice utjecati na nas.

Na početku je sva ta izrada internetskih stranica bila puno jednostavnija. Oblikovale su se HTML-om, bile su vrlo jednostavne te im je jedina svrha bila prijenos informacija. Stoga su bile većinom tekstualne, eventualno uz koju sliku. Tada se pojavio CSS te su front-end developeri, odnosno ljudi koji se bave kodiranjem internetskih stranica dobili puno više mogućnosti. S više mogućnosti oblikovanja dolazilo je i puno više posla, zahtijeva klijenata, načina izrade i slično. Programeri uvijek žele za čim manje vremena učiniti čim više posla. Uvijek se traže neki prečci kako se kod ne bi morao pisati vremenski dugo, a uz vrlo malo dobivenog rezultata. Zbog toga su nastali CSS-a pred-procesori, produžeci CSS-a. Jedan od njih je i LESS.

LESS je jedan od prvih CSS pred-procesora koji su se pojavili u svijetu izrade internetskih stranica. Vrlo brzo je postao popularan jer je donio značajke poput varijabli, nestinga, operacija pa čak i funkcija. Danas se traži njegovo znanje na bilo kojem ozbiljnom razgovoru za posao front-end developera.

U ovom završnom radu objasniti ću što je to CSS pošto nam je on vrlo bitan za LESS pred-procesor. Tada ću objasniti što su točno CSS pred-procesori, zašto su nam toliko važni i koja im je svrha. Dalje ću objašnjavati što je LESS, njegove značajke i konkurenciju. Obratit ću pažnju na sintaksu, varijable i operacije. Objasnit ću što su to nesting i mixing. Pokušat ću odgovoriti na pitanje zar je LESS uistinu tolika prednost u odnosu na CSS i koliko ga je kompliciranije koristiti. Nakon toga ću napraviti praktični rad o LESS-u. To će biti internetska stranica kojoj će svrha biti učenje LESS-a za korisnike kojima CSS više nije dovoljno dobar. Na stranici će biti pojašnjene najvažnije značajke tog pred-procesora pomoću definicija, primjera i kodova. Cijeli praktični dio završnog rada biti će oblikovan pomoću LESS-a.

2. CSS

CSS (Cascading Style Sheets) je stilski jezik za oblikovanje internetskih stranica. Prva verzija je izašla 1996. godine i promijenila način izrade stranica. Korisnicima se više nisu davale samo informacije u obliku nepreglednog teksta, nego se htjelo postići da to bude što preglednije i jednostavnije. To je bilo moguće pošto je CSS omogućavao pozicioniranje elemenata, oblikovanje teksta i slika, uređivanje tablica i slično. Time su stranice postale ugodnije za korištenje pa je korisnik lakše i brže dolazio do željenih informacija. Isto tako je olakšao posao programerima. Pošto se CSS mogao nalaziti u zasebnoj datoteci, HTML dokumenti su zadržavali puno manje koda. Time smo dobili mogućnost lakšeg oblikovanja, ali i mijenjanja samog sadržaja. [1]

2.1. Povezivanje CSS stilova

CSS stilove povezujemo s HTML datotekom na tri načina. Nabrojat ću ih po prioritetu izvršavanja u web pregledniku. Prvi način je linijski. Njega primjenjujemo direktno na element u HTML datoteci pomoću atributa *style* kojem dodajemo bilo koje CSS svojstvo. Taj se način najmanje koristi i možemo ga pronaći većinom na starijim internetskim stranicama. [2]

```
<h1 style="color : blue;">Naslov</h1>
```

Drugi način je unutarnji. CSS stilove koristimo unutar sekcije *head* u HTML datoteci. S tim načinom možemo opteretiti HTML datoteku s previše koda pa je teže raditi promijene i održavanje zbog nepreglednosti. [2]

```
<html>
<head>
     <title> </title>
     <style>
          h1 {color : blue};
     </style>
</head>
</html>
```

Vanjski način je najkorišteniji način povezivanja CSS stilova. Sav CSS se piše u odvojenoj datoteci od HTML-a. Unutar CSS datoteke upisujemo selektore te im dodajemo stilove koje želimo primijeniti na njih. Važno je da su te datoteke povezane pomoću poveznice u zaglavlju HTML datoteke. [2]

```
<link rel="stylesheet" type="text/css" href="style.css">
```

2.2. Sintaksa

CSS nije osjetljiv na velika i mala slova. Njegova sintaksa se sastoji od dva osnovna dijela: selektora i deklaracijskog bloka. Selektor odabire koji će dio, odnosno element u HTML dokumentu biti oblikovan. Selektoru pridružujemo deklaracijski blok koji u vitičastim zagradama sadrži jednu ili više deklaracija. Deklaracije moraju uvijek završavati s točka-zarezom. Sastoje se od svojstva i vrijednosti koje odvajamo dvotočkom. Svojstvo definira koji dio stranice će opisati vrijednost koju mu dodamo. Pa tako pod svojstvo spada font, pozadina, slika i slično, dok mu s vrijednošću dodajemo veličinu, boju, okvir i ostalo. Ukoliko želimo ista svojstva i vrijednosti primijeniti na više različitih selektora, dovoljno je nabrojiti selektore koje odvajamo sa zarezom te im dodajemo zajednički deklaracijski blok. [1]

2.3. Selektori

Kao što sam već spomenuo, CSS selektori odabiru koji element će biti stiliziran. Postoji više vrsta selektora. Najčešće korišteni je selektor klase. Njega definiramo pomoću točke koja se navodi ispred imena. Prednost ovog selektora je da više HTML elemenata može imati istu vrijednost. Zbog toga možemo stilizirati više elemenata sa samo jednim selektorom i deklaracijskim blokom. Ukoliko želimo stilizirati određeni element s određenim nazivom klase, prije točke napišemo ime elementa (npr. *h1*), zatim točku pa naziv klase. ID selektor je sličan selektoru klase samo što se on definira za samo jedan HTML element. Više HTML elemenata ne smiju imati istu ID vrijednost. Druga razlika je što ga definiramo pomoću znaka ljestve (#). Sljedeći je selektor HTML oznaka. To je ustvari naziv elementa poput *body*, *h1*, *p* ili slično. Time svi elementi istog tipa imaju zajednički stil. Zadnji su pseudo selektori. Oni se kombiniraju s ostalim selektorima na način da ih odvojimo s dvotočkom. Najčešće se koriste za linkove, ali može ih se dodati i ostalim HTML elementima. Pseudo selektori stiliziraju element kada se strelica miša nalazi na njima (:*hover*), kada kliknemo na element (:*active*), kada je element posjećen (:*visited*) i slično. Isto tako možemo dodavati sadržaj prije ili poslije elemenata (:*before*, :*after*). Na taj način možemo dodati sliku prije ili poslije nekog teksta pomoću CSS-a. Postoji i univerzalni selektor prikazan u obliku zvjezdice (*). Njime svim elementima u HTML-u dodajemo ista svojstva i vrijednosti. [1]

3. CSS pred-procesori

CSS nam je omogućio nove značajke u oblikovanju internetskih stranica koje su prije bile nezamislive. Svejedno nekim programerima to nije bilo dovoljno. Svaki programer želi pisati što manje koda, a dobiti što veći rezultat. CSS zahtijeva puno pisanja istog koda te su se samim time programeri ponavljali kod pisanja. Stoga su izmislili CSS pred-procesore.

"CSS pred-procesor je prošireni jezik koji se kompajlira u CSS." [3] Jednostavnije rečeno, CSS pred-procesori su programski jezici koji se moraju kompajlirati, odnosno prevoditi u CSS sintaksu kako bi dobili željeni rezultat na našoj internetskoj stranici. Oni uvelike proširuju mogućnosti CSS-a. Dozvoljavaju nam varijable, ugnježđivanje, operacije, funkcije, mixin i ostalo. Kod manjih projekata CSS pred-procesori ne doprinose nekom vidljivom poboljšanju u usporedbi s CSS-om, ali kod većih projekata gdje se koristi puno linija koda on uvelike doprinosi kvaliteti sadržaja. Time programeri zbog prijašnje navedenih mogućnosti štede vrijeme kod pisanja i mijenjanja koda za oblikovanje internetske stranice. [3]

Postoji nekoliko CSS pred-procesora, ali tri najveća su SASS, LESS i Stylus. Oni osim što su sa svojom sintaksom i mogućnostima najbolji, također su baš zbog tog razloga najkorišteniji, a samim time i najpoznatiji. [3]

4. LESS

LESS je dinamički stilski jezik. Podržava identičnu sintaksu kao i CSS, ali s mogućnošću dodavanja različitih elemenata uobičajenih za dinamičke programske jezike. Stoga nam LESS dopušta pisanje sasvim identičnog koda kao i u CSS datoteci, ali nam također dopušta dodavanje varijabli i funkcija koje nam olakšavaju i krate vrijeme oblikovanja internetskih stranica. Često se susrećemo s pojmom CSS pred-procesori. Postoji više CSS pred-procesora, a LESS je upravo jedan od najpoznatijih i najkorištenijih. Pred-procesor je zapravo skriptni jezik koji proširuje CSS i prevodi se u običajnu CSS sintaksu. [4]

LESS se prvi puta pojavio 2009. godine i bio je predviđen kao konkurencija SASS-u. Prva verzija je napisana za programski jezik Ruby. [5] Zbog dobre prihvaćenosti pojavile su se implementacije za PHP i .NET. Programeri su htjeli još više pojednostaviti LESS pa su ga prebacili na JavaScript. Time je LESS postao i do 40 puta brži, ali i jednostavniji za implementaciju. Za korištenje više nije bio potreban serverski orijentirani jezik, nego bilo koji internetski preglednik. [6]

4.1. Kompajliranje LESS-a u CSS

Kod pisanja LESS-a kod se mora kompajlirati, odnosno prevesti u uobičajeni CSS kod. To je potrebno jer internetski preglednici ne podržavaju LESS. Kompajliranje se provodi preko servera ili klijenta. [5]

LESS ima opciju kompajliranja koja je vrlo jednostavna. Možemo skinuti paket datoteka s njihove stranice, povezati ih s *.html* datotekom i naš će se kod preko preglednika kompajlirati u CSS. Ta opcija je dobra za početnik koji tek uče LESS. Ukoliko radimo neke velike projekte, to može usporiti učitavanje internetske stranice. JavaScript datoteke mogu biti većih veličina ukoliko radimo na zahtjevnijim projektima. Posebice sporo će se kompajliranje vršiti na mobilnim telefonima. Time se gubi dobro korisničko iskustvo. Također nije moguće koristiti sve internetske preglednike za taj način kompajliranja. To podržavaju samo najnovije verzije najpoznatijih preglednika poput Chroma, Mozille Firefox, Safaria i Internet Explorera. Isto tako ne vrše svi preglednici kompajliranje na isti način te nam stranica vjerojatno neće izgledati jednako. [5]

Postoji drugi način koji je također dobar i jednostavan za korištenje, a ne gubimo na brzini učitavanja stranica jer se kod kompajlira direktno na našem računalu. To su programi poput WinLessa, Chruncha, LessEnginea i slično. Za primjer možemo uzeti WinLess kompajler. On ima grafičko sučelje te je prelagan za korištenje. Nakon instalacije otvorimo program, preko njega odaberemo mapu gdje nam se nalazi projekt, odnosno *.less* datoteka i kod se kompajlira.

Odlična stvar je u tome što se kompajliranje vrši u realnom vremenu. Nakon toga prebacimo *.less* i *.css* datoteku na server. Internetski preglednici će čitati *.css* datoteku i riješili smo problem sporog učitavanja stranice.

Još jedan od načina za kojeg službena stranica (http://lesscss.org/#using-less) kaže da je najlakši je putem npm-a ("Node Package Manager"). Moramo skinuti JavaScript platformu, odnosno Node.js na naše računalo kako bi mogli vršiti kompajliranje na taj način. Kada je Node.js instaliran, otvorimo Node.js Command Prompt. Unutar njega napišemo liniju koda "*$ npm install -g less*" kojim instaliramo LESS. Zatim pomoću komandne linije nađemo mapu u kojoj nam se nalazi *.less* datoteka. Kada uđemo u tu mapu napišemo *lessc* što znači *less compiler* i odaberemo *.less* datoteku, a zatim *.css* datoteku u koju će se kompajlirati. Pritisnemo enter i ukoliko nam se ne pojavljuje nikakva greška, kod je kompajliran. [5]

4.2. Varijable

Definicija varijable govori da je "varijabla memorijska lokacija čiji se sadržaj može mijenjati tokom izvođenja programa". [7] Kada stiliziramo internetsku stranicu pomoću CSS-a i želimo da nam sav tekst bude u nekoj boji heksadekadskog zapisa, taj zapis moramo pisati i do nekoliko puta unutar datoteke. Pošto se taj zapis u većini slučajeva sastoji od 6 znamenaka, vrlo je teško pamtljiv te se mora kopirati. Zato je LESS uveo varijable koje deklariramo najčešće na početku dokumenta. U LESS-u je varijablu moguće i prvo pozvati, a tek onda deklarirati. To se u informatici naziva *lazy loading*. Pa tako možemo dodati varijablu za neku boju i pozivati ju kroz cijeli dokument. Time dobijemo pregledni i organizirani kod. Varijablu definiramo na način da ispred imena dodamo znak @, a zatim ime varijable po želji. Moram naglasiti da je i kod LESS-a deklariranje varijable osjetljivo na velika i mala slova. Pa ako uzmemo prošli primjer i želimo da nam cijeli tekst na internetskoj stranici ima plavu boju, potrebno je na početku dokumenta deklarirati varijablu poput *@tekst* te joj dodati vrijednost. Tako bi cijela varijabla izgledala ovako: "*@tekst: #1d50a3;* ". Kada bi tekstu trebali dodati boju, samo bi se pozvali na varijablu *@tekst* i on bi postao plav. Varijable osim što su pojednostavile pisanje koda, omogućile su i puno jednostavnije mijenjanje. Pa tako ukoliko zbog nekog razloga više ne želimo da nam tekst bude plave boje, samo je potrebno promijeniti vrijednost varijable, odnosno heksadekadski zapis u bilo koju drugu boju koja nam odgovara. To je naravno moguće raditi s bilo kojim svojstvima, a ne samo s bojama. [5] Varijable možemo deklarirati za selektore, poveznice i svojstva. Pa tako umjesto da poveznicu slike pišemo svaki puta, jednostavno napravimo varijablu. S time opet kratimo vrijeme pisanja i pojednostavljujemo kod. [8]

Ovdje je primjer deklariranja i pozivanja varijabli:

```
//Varijabla
@tekst: #1d50a3;  //plava boja
@slika: "img/slika1.png"; // URL slike

h2 {
     color: @tekst;
}

p {
     font-size: 16px;
     color: @tekst;
}

.wrapper {
     background-image: @slika;
     font-family: "Arial", sans-serif;
     font-style: italic;
     color: @tekst;
}
```

Važno je da pravilno biramo imena varijabli pa da nam kasnije ne stvaraju probleme. Imena moraju biti smislena kako bi znali o čemu se radi. Ukoliko koristimo plavu boju, ne smijemo staviti ime varijable *@plava* jer kasnije može doći to potrebe za mijenjanje boje u zelenu. Tada ćemo imati varijablu *@plava* s vrijednošću heksadekadskog zapisa zelene boje. To osim što nema smisla, može i stvarati probleme programerima kod održavanja i mijenjanja internetske stranice. [5]

4.3. Ugniježđeni stilovi (Nested rules)

Ugniježđeni stilovi su još jedna velika prednost LESS-a. Zbog njih možemo u jednom selektoru pisati druge selektore koji su povezani s njime. Druge selektore dodajemo pomoću selektora klase, ali možemo staviti i samo HTML oznaku. Na primjeru to možemo objasniti pomoću sekcije i naslova. U HTML-u imamo element *section* koji sadrži naslov (*h1*). Pomoću CSS-a normalno oblikujemo tu sekciju, ali želimo da naslov ima drugačije oblikovanje od drugih naslova na stranici. Umjesto da elementu *h1* dodajemo klasu, u LESS-u je potrebno napisati normalni CSS za oblikovanja sekcije i u njezinom deklaracijskom bloku navesti *h1* element s njegovim oblikovanjem. Evo primjera:

```
section {
     witdh: 100%;
     height: auto;
     background-color: blue;
          h1 {
               font-weight: bold;
               color: white;
          }
}
```

U CSS-u bi morali elementu *h1* dati ime klase te ga stilizirati pomoću selektora klase ili ga napisati na slijedeći način:

```
section {
     witdh: 100%;
     height: auto;
     background-color: blue;
}

section h1 {
     font-weight: bold;
     color: white;
}
```

4.4. Operacije

"LESS podržava matematičke operacije zbrajanja (+), oduzimanja (-), množenja (*) i dijeljenja (/)". [5] Operacije je moguće izvršavati na brojkama, varijablama i bojama. Preporuča se da se operacije stavljaju unutar zagrada. LESS ima mogućnost konvertiranja brojeva. Pa tako ukoliko zbrajamo centimetre i milimetre, LESS će konvertirati milimetre u centimetre i rezultat ćemo dobiti u centimetrima. To je moguće samo kod uspoređivanja, zbrajanja i oduzimanja. Množenje i dijeljenje ne podržava konvertiranje jer u većini slučajeva nije smisleno. Isto tako je moguće konvertirati samo jedinice istog tipa. LESS neće konvertirati piksele u centimetre ili obrnuto. [9]

```
@operacija: (10cm + 50mm); // rezultat je 15cm
@operacija: (10px + 10cm); // rezultat je 20px
```

4.5. Mixin

LESS.js podržava značajku mixin. Ona nam dozvoljava da klasi dodamo neka svojstva, zatim pozovemo novu klasu kojoj kao svojstvo dodamo ime prethodne klase. S time nova klasa ima jednaka svojstva kao i prethodna klasa. Mixin je zbog toga jako korisna stvar jer ne moramo više puta pisati ista svojstva. Opet naglašavam da je kod programiranja bitno što više smanjiti kod te ga pojednostaviti kako bi se lakše snalazili u njemu i mijenjali ga ukoliko je to potrebno. Primjer mixina je ukoliko želimo staviti sliku određene dužine i širine te određenih margina. Napišemo ime klase i damo joj svojstva koja želimo. Tada umjesto da pišemo sva ta svojstva ponovno, dodamo ime te klase kao atribut kod svih ostalih selektora za slike koje želimo oblikovati tim stilom. Evo primjera:

```
.oblikovanje-slike {
     width: 100px;
     height: 100px;
     margin: 10px 20px;
```

```
}

.slika1 {
     .oblikovanje-slike ();
     border: 1px solid black;
}
```

Ovo u CSS-u izgleda ovako:

```
.oblikovanje-slike {
     width: 100px;
     height: 100px;
     margin: 10px 20px;
}

.slika1 {
     width: 100px;
     height: 100px;
     margin: 10px 20px;
     border: 1px solid black;

}
```

Možemo vidjeti da slika ima ista svojstva kao i klasa imena *oblikovanje-slike*. To smo postigli tako što smo tu klasu dodali kao svojstvo te smo klasi imena *slika1* još dodali i obrub. Isto tako možemo vidjeti da smo dodali zagrade kod uključivanja imena klase kao svojstvo. Zagrade možemo dodati zbog preglednosti, ali nije nužno. Naredba će se izvršiti u oba slučaja.

Mixin je vrlo sličan funkcijama pa mu stoga možemo dodavati parametre. Recimo da imamo obrub od 5 piksela na prvoj slici, na drugoj slici nam treba obrub od 10 piksela, dok na trećoj od 15 piksela. To možemo napraviti pomoću mixin značajke tako što joj dodajemo parametar. Nakon što napišemo klasu koja će nam služiti za oblikovanje obruba slika, odmah pored u zagradama deklariramo varijablu i damo joj vrijednost ovisno kakav obrub želimo. Tako svaka slika ima isti stil, ali je obrub različite debljine jer smo mijenjali parametre. Evo primjera:

```
.obrub-slike (@obrub: 5px solid black) {
     width: 20%;
     height: 50px;
     border: @obrub;
}

.slika1 {
     .obrub-slike;
}

.slika2 {
     .obrub-slike (10px solid black);
}

.slika3 {
     .obrub-slike (15px solid black);
}
```

LESS je drugačiji od svoje konkurencije jer se trudi ostati što sličniji CSS-u. Tako on podržava varijable i funkcije kao i JavaScript, ali s nešto drugačijom sintaksom. U LESS-u koristimo sintaksu sličnu medijskim upitima umjesto if/else funkcija. Koriste se relacijski operatori (<, >, =, >=, =<), logički operatori i određene ključne riječi. Evo primjera sličnog koda kao i na stranici LESS.js [10]. Ovo je jednostavan primjer pomoću kojega možemo objasniti sintaksu i funkcioniranje funkcija u LESS-u.

```
.mixin (@a) when (lightness(@a) >= 50%) {
     color: @a;
     background-color: black;
}
.mixin (@a) when (lightness(@a) < 50%) {
     color: @a;
     background-color: white;
}

.mixin (@a) {
     width: 100px;
     height: 100px;
}

.container {
     .mixin (darkblue);
}
```

U kodu imamo dva uvjeta. Prvi uvjet nam govori ukoliko je tekst koji se nalazi u *div* elementu klase imena *container* svjetline manje ili jednake od 50%, tada će pozadina biti crne boje. Drugi uvjet je postavljen tako da kod teksta svjetline veće od 50% pozadina bude bijele boje. U ovom primjeru koda imamo tekst tamno plave boje pa u pregledniku dobivamo pozadinu bijele boje. Ukoliko želimo da nam pozadina bude crne boje, moramo staviti svjetliji tekst. Koristimo *when* kao ključnu riječ i relacijske operatore više, manje i jednako. Ovaj kod bi u JavaScriptu izgledao ovako: [11]

```
function mixin(a){
  if (lightness(a) >= 50%){
   element.style.backgroundColor = "white";
 } else if (lightness(a) < 50%){
   element.style.backgroundColor = "black";
  }
}
```

Pomoću varijabla, operatora i svojstva mixina možemo raditi petlje. Ponovno je sintaksa prilagođena da sliči što više CSS-u. Jednostavna petlja u LESS-u izgleda ovako: [11]

```
.petlja (@i:1) when (@i <=3) {
     .div-@{1} {
          width: 10px;
     }
```

```
    .petlja(@1 + 1);
}

.petlja();
```

Kod nakon kompajliranja izgleda ovako:

```
.div-1 {
    width: 10px;
}

.div-2 {
    width: 20px;
}

.div-3 {
    width: 30px;
}
```

5. Konkurencija

LESS ima dva velika konkurenta, SASS i Stylus. Postoje još nekoliko CSS pred-procesora, ali nisu ni blizu kvaliteti ovoj trojici. Stoga ćemo spomenuti samo njih. Na prvi pogled su vrlo slični, ali postoje male razlike koje programerima puno znače za jednostavnije i stabilnije programiranje. Teško je odrediti koji pred-procesor je bolji jer svaki ima svoje prednosti i nedostatke.

Za početak ću ukratko nabrojati prednosti i nedostatke LESS-a. LESS je najjednostavniji što se tiče prilagođavanja i učenja. Preporuča se novim korisnicima zbog sintakse najsličnije CSS-u. Isto tako je jednostavan za korištenje jer se koristi programskim jezikom JavaScript. Nije potrebna instalacija dodatnih programa, nego ga možemo jednostavno pozvati preko poveznice. Iako to nije najbolje rješenje, opet imamo mogućnost korištenja programa za kompajliranje koji imaju grafičko sučelje. Povezivanja preko poveznice je dobro ukoliko se koristi za učenje ili neke jednostavne i male projekte. Novim korisnicima ili korisnicima koji nisu upoznati sa serverskom stranom programiranja to mnogo olakšava korištenje. Nedostatak LESS-a je što su SASS i Stylus kompleksniji. Kod LESS-a je puno teže ili nemoguće napisati neke zahtjevnije funkcije. Razlog tome je što on koristi medijske upite za pisanje funkcija. Također je u SASS-u kod ugnježđivanja i mixina više toga dozvoljeno. [12]

5.1. SASS

SASS se prvi puta pojavio već 2006. godine, čak 3 godine prije LESS-a. Iako je razlika u pojavljivanja za informatički svijet vrlo velika, ideja iza oba pred-procesora je ista; pojednostaviti pisanje CSS-a. [13] SASS je odmah u početku zamišljen više kao programski jezik pa mu je tako i sintaksa bila nešto drugačija od CSS-a. S vremenom se sintaksa mijenjala te najnovija verzija podržava identičnu sintaksu CSS-a. Međutim postoji razlika u sintaksi između SASS-a i LESS-a. Pa tako kod SASS-a možemo izostaviti vitičaste zagrade i točku-zarez. Iako je to moguće, nije preporučljivo te se koristilo većinom na starijim verzijama. Izostavljanje vitičaste zagrade i točke-zarez može biti pozitivna stvar, ali i ne mora. S time možemo ubrzati pisanje koda i učiniti ga jednostavnijim, ali možda i prejednostavnim. Kod može izgledati nepovezano pa se teško snalaziti u njemu. Teže je shvatiti gdje je početak i kraj klasa, funkcija i slično. Što se tiče brzine pisanja, svi noviji editori pomažu kod pisanja te nije potrebno pisati doslovno svaku zagradu, točku pa čak niti svojstva i vrijednosti. Editori to sve već čine umjesto nas ukoliko ih znamo koristiti. [14]

Deklariranje varijabli u SASS-u je vrlo slično kao i kod LESS-a. Možemo deklarirati brojke, tekst i boje. Umjesto znaka at (@) kao kod LESS-a, koristimo znak dolara ($) ispred imena

varijable. Isto tako SASS podržava *lazy loading*, odnosno varijablu možemo prvo pozvati pa ju tek onda deklarirati bilo gdje u dokumentu. Potrebno je koristiti dvotočku za odvajanje imena i vrijednosti varijable, ali ne moramo koristiti točku-zarez na kraju kao kod LESS-a. [14]

SASS ima nekoliko prednosti pred LESS-om. Jedna od njih je njegova kompleksnost kod pisanja funkcija. SASS više sliči programskom jeziku, nego stilskom. Prema tome je jednostavnije pisati kompleksnije, odnosno kompliciranije funkcije. Njegova sintaksa je logičnija, posebice programerima. Sljedeća pozitivna stvar, možda i najveća, je zastupljenost i korištenje. SASS ima najveću uporabu među programerima i dizajnerima, puno veću nego LESS i Stylus. Ima više razloga zašto je to tako. Pa osim svoje kompleksnosti, Compass koristi SASS. Compass je jedan od CSS frameworka koji dizajneri vole koristiti na svojim projektima. Još jedan od frameworka koji koristi SASS je najnoviji Bootstrap. To je jedan od najpopularnijih frameworka za izradu internetskih stranica. Bootstrap je u početku koristio LESS kao CSS-ov pred-procesor, ali baš su zbog velike korištenosti SASS-a među programerima odlučili najnoviju platformu prilagoditi njemu. SASS sa svojom velikom zajednicom dobiva veliku prednost među konkurentima. Osim što ima odličnu podršku, svi noviji framworki su pisani SASS-om jer žele biti što zastupljeniji. Isto tako se najbrže mijenja, poboljšava i prilagođava korisnicima. [12]

Svi programeri moraju znati CSS ukoliko žele naučiti koristiti neki pred procesor. SASS nije najbolji za početnike kao što je recimo LESS. LESS ima sličniju sintaksu CSS-u pa je novim korisnicima lakše početi s njime. SASS je kasnije također prilagodio sintaksu CSS-u, ali je svejedno potrebno više vremena za učenje i prilagodbu. Još jedna mana SASS-a je što mu je potreban Ruby. To nije toliki problem programerima koji su jako dobro upoznati s back-end developmentom, ali zna davati poteškoće dizajnerima i novim programerima koji su tek počeli ulaziti u svijet izrade internetskih stranica. [12]

5.2. Stylus

Stylus je nastao nešto kasnije nego LESS, točnije kojih godinu dana nakon. Nije popularan kao LESS ili SASS, ali svejedno ima svoju grupu korisnika. U sintaksi nema prevelike razlike između njega i SASS-a. Kod Stylusa osim što možemo izostaviti vitičastu zagradu i točku-zarez, ne moramo pisati niti dvotočku. Korisnik ima najveću slobodu u pisanju koda te mu može pojednostaviti izgled. Time se Stylus podosta udaljio od CSS sintakse. Kao što sam i napomenuo kod SASS-a, jednostavan kod ne mora uvijek biti i najbolja opcija. Deklariranje varijabli podržava i Stylus. Deklariramo brojke, tekst i boje, ali na malo drugačiji način. Kod deklariranja varijabli ne moramo stavljati nikakav znak na početku, iako je moguće staviti znak dolara ($). Isto tako nije potrebno stavljati točku-zarez na kraj varijable. Jedino što moramo je staviti znak jednako između imena i vrijednosti. [15]

Stylus je najsličniji programskom jeziku. Korisniku dozvoljava pisanje kompleksnih funkcija na jednostavniji način kao što to dopuštaju LESS i SASS. Mnogi mu kao prednost uzimaju i jednostavnu sintaksu. Pisanje koda je brže, a on izgleda čišće. Stylus se pokreće preko Node.js platforme. Nije mu potreban Ruby kao SASS-u. Nedostatak Stylusa je vrijeme koje je potrebno novom korisniku za prilagodbu. U njega treba uložiti najviše vremena da ga se nauči koristiti u njegovom punom potencijalu. SASS i LESS su također puno više zastupljeni te se puno više i koriste. To je veliki problem zbog podrške i razvijanja Stylusa. Ukoliko je broj korisnika mali te se k tome i smanjuje zbog sve većeg jačanja SASS-a, Stylus će se razvijati sve sporije te će korisnika biti sve manje. [12]

6. Praktični rad

Za praktični dio završnog rada osmislio sam internetsku stranicu o LESS-u. Svrha stranice je učenje i pružanje informacije početnicima kojima CSS više nije dovoljno dobar te su spremni na nešto kompleksnije. Na jednom mjestu sakupio sam sve najvažnije dijelove potrebne za savladavanje tog pred-procesora. Pa tako praktični dio sadrži 6 *.html* datoteka uključujući i početnu stranicu na kojima se sadržaj prikazuje korisnicima. To su *index.html, about.html, varijable.html, nesting.html, operacije.html* i *mixin.html*. Na stranicama se uz definicije nalaze i primjeri kodova LESS-a. Da bi te kodove i sintaksu LESS-a bilo što jednostavnije shvatiti, stavio sam i primjere tih kodova kompajliranih u CSS. Cilj je korisnicima objasniti što jednostavnije definiciju i kod.

U početku izrade praktičnog rada u planu je bila izrada "*one pagera*", odnosno jedne *.html* datoteke s cjelokupnim sadržajem koju bi se pregledavalo klizanjem miša. Takva stranica je vizualno ljepša i informacije se mogu pružati na zanimljiviji način. Isto tako je jednostavnija za izradu i prilagodbu za mobilnu verziju. Odustao sam od te ideje jer smatram kako se svaki dio teorije o LESS-u mora vidljivo odvojiti da bi učenje bilo što jednostavnije. Isto tako korisnici moraju znati o kojem dijelu teorije čitaju u određenom trenutku. To sam postigao izradom 6 *.html* datoteka nabrojanih u prijašnjem odlomku. Dizajn stranica je vrlo jednostavan. Nema nepotrebnih elementa koji bi korisnicima odvraćali pozornost od sadržaja. Koristio sam tri boje: plavu, sivu i bijelu. Njima sam razdvojio cjeline zbog lakšeg snalaženja po stranicama.

Programski jezici korišteni kod izrade praktičnog dijela su HTML, LESS, CSS, JavaScript i jQuery. Svi nazivi klasa, varijabli i funkcija su na engleskom jeziku kao i kod profesionalne izrade internetskih stranica. Gdje i kako sam koji od tih programskih jezika koristio objasnit ću detaljno u daljnjem dijelu rada.

6.1. Programi korišteni kod izrade

Najvažniji program korišten u završnom radu je Sublime Text 3 uređivač teksta. To je vrlo brz program koji podržava mnogo dodataka te je jako prilagodljiv korisnicima. Izgledom je vrlo jednostavan pa tako s lijeve strane imamo izbornik s našim mapama i datotekama, na vrhu se nalazi mali izbornik s opcijama programa, a najveći dio zaslona uzima dio gdje se upisuje kod. Pošto je program dosta prilagodljiv korisnicima, možemo ga promijeniti na način kako nama odgovara za rad. Pa tako možemo podijeliti dio koji služi pisanju koda na dva ili više dijela. Kao primjer možemo uzeti dvije datoteke, *.html* i *.less* datoteku. Njih možemo otvoriti istovremeno te će jedna biti prikazana na gornjem dijelu zaslona programa, a druga na donjem. Tako se možemo bolje snalaziti s imenima selektora, poveznicama i slično. Isto tako taj način pomaže kod

mijenjanja koda, ali i kad imamo više zaslona na kojima radimo. Program možemo staviti i u "*Distraction Free Mode*". To je opcija koja stavlja dio za pisanje koda na puni zaslon i vidimo samo kod kako nas ništa ne bi ometalo. Brzina Sublimea je njegova sljedeća velika vrlina. Ukoliko imamo otvoreno više datoteka u kojim pišemo kod te mijenjamo pogled na njih, one se otvaraju u istom trenutku. Nema zastajkivanja ili bespotrebnog čekanja na učitavanje. Sublime ima i prenosivu verziju. Ukoliko pišemo kod na tuđem računalu ne moramo instalirati program, nego ga je dovoljno imati na USB-u i pokrenuti ga preko njega. Tako koristimo naš, već prilagođeni Sublime. Za Sublime Text postoje razni dodaci koji se mogu instalirati. Pa tako jedan od prvih koji je potreban svakom programeru je Emmet. To je dodatak koji olakšava pisanje koda. Umjesto da pišemo cijele HTML tagove i klase, dovoljno je napisati unaprijed određene kratice, pritisnuti enter i ostatak koda se pojavljuj sam. Time si ubrzamo pisanje koda i uštedimo vrijeme za pisanje važnijih naredbi. Program podržava po zadanim postavkama jako puno programskih jezika. Nažalost LESS nije jedan od njih. Stoga sam prije početka izrade stranice za praktični dio završnog rada morao instalirati dodatak za LESS. Sublime radi na svim operacijskim sustavima (Linux, Windows, OS X). Program se može skinuti i koristit besplatno u probnoj verziji koja se ne razlikuje mnogo od licencirane.

WinLess je program koji služi za kompajliranje koda. To je vrlo jednostavan program s grafičkim sučeljem koji nam omogućuje kompajliranje LESS-a na najjednostavniji način. Dovoljno je otvoriti WinLess i odabrati mapu u kojoj se nalaze *.less* i *.css* datoteke. On ih automatski prepoznaje i vrši kompajliranje. WinLess je dovoljno minimizirati i on će u realnom vremenu u pozadini izvršavati svoj zadatak. Datoteka CSS-a će nakon kompajliranja imati jednu liniju bez praznih mjesta kako bi se smanjila njezina memorija.

Adobe Illustrator je program za izradu vektorske grafike. Vektorska grafika je odlična za internetske stranice jer se tijekom mijenjanja rezolucije ekrana kvaliteta slike ne pogoršava. Illustrator nisam koristio za izradu grafika koja će se nalaziti na stranicama, nego za izradu "*wireframea*", početne skice kako bi stranica trebala izgledati. Nakon što sam odlučio koja je svrha stranice, napravio sam u Illustratoru grubu skicu dizajna.

Adobe Photoshop je najpoznatiji program koji služi za obradu fotografija. Program je vrlo kompleksan s mnogo mogućnosti. Koristio sam ga za oblikovanje slike pozadine na početnoj stranici te slike pozadine izbornika na ostalim stranicama. Sliku sam potamnio te joj promijenio veličinu kako bi što bolje odgovarala stranici.

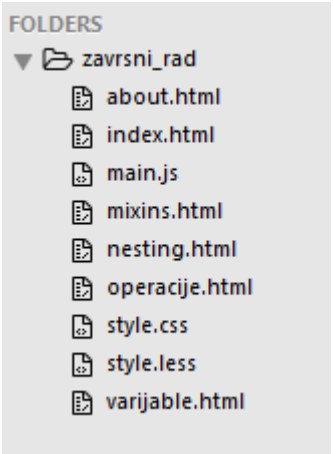
WinSCP je posljednji program korišteni za izradu praktičnog rada. To je program kojim prebacujemo i upravljamo mapama i datotekama na nekom serveru. Ima grafičko sučelje pa je taj proces vrlo jednostavan. Njega sam koristio nakon što je cijeli projekt bio gotov da ga prebacim na server Sveučilišta Sjever kako bi bio dostupan svim studentima u bilo kojem trenutku.

6.2. Početak rada

Za početak na radnoj površini radimo novu mapu imena *zavrsni_rad* i u njoj tri datoteke, *index.html*, *style.css* i *style.less*. Nakon toga pokrećemo WinLess program i biramo prijašnje izrađenu mapu. Program automatski prepoznaje *.css* i *.less* datoteku te ih počinje kompajlirati. U Sublime Text 3 uređivaču teksta možemo početi s pisanjem HTML koda. Prvo pišemo početnu strukturu HTML-a te ga u zaglavlju povezujemo sa CSS datotekom.

```
<link href="style.css" rel="stylesheet" type="text/css" />
```

Nakon provjere da li je *.css* datoteka povezana s *.html* datotekom, izrađujemo još 5 *.html* datoteka. To su *about.html*, *varijable.html*, *nesting.html*, *operacije.html* i *mixin.html*. Također izrađujemo i datoteku *main.js* koja će nam koristiti za pisanje JavaScript i jQuery funkcija.



Slika 1. Datoteke projekta

Sve *.html* datoteke u *head* tagu sadržavaju način kodiranja za jezične znakove, ime stranice u *title* tagu, meta podatak za responzivnu stranicu, link koji *.html* datoteku povezuje s *.css* datotekom i link na ikonu LESS-a koja se pojavljuje u karticama (tabovima) internetskih preglednika.

```
<head>
    <meta charset="utf-8"/>
    <title>LESS</title>

    <meta name="viewport" content="width=device-width, initial-
    scale=1, maximum-scale=1, user-scalable=no">
    <link href="style.css" rel="stylesheet" type="text/css" />
    <link rel="icon" href="img/icon.png">
</head>
```

Nakon HTML-a prelazimo na pisanje koda u *.less* datoteci. U prijašnjim odlomcima spomenuo sam izradu početne skice u Adobe Illustratoru. Najbolje je već tada odabrati boje i fontove koji će se koristiti. To nam omogućuje da na početku *.less* dokumenta izradimo nekoliko varijabli. Prva varijabla koju deklariramo je za font. Varijabla ima jednostavno ime *font*, a za njezinu vrijednost odabiremo font "*Arial*" koji je idealan za čitljivost na zaslonima te je često korišten na internetskim stranicama. Druga varijabla je *fontSize* koja za vrijednost ima veličinu fonta od 16 piksela. Sljedeće varijable odnose se na boje. Boja za naslove na stranicama deklarirana je pod varijablu imena *headlineColor* i ima vrijednost bijele boje. Varijabla za boju teksta, odnosno *textColor* ima vrijednost tamno sive, dok varijabla za sjene *shadowColor* nosi vrijednost crne boje. Font, njegova veličina i boje teksta su jedne od stvari koje se najčešće mijenjanju na internetskim stranicama. Stoga moramo baš za njih napraviti varijable pa ukoliko dođe do potrebe za mijenjanjem, to možemo napraviti na jednom mjestu, odmah na početku *.less* dokumenta u samo nekoliko sekundi. Ovdje su varijable korištene u našem projektu:

```
@font: "Arial", sans-serif;
@fontSize: 16px;
@headlineColor: #f2f2f2;  //bijela boja
@shadowColor: #333333;  //crna boja
@textColor: #525252;  //tamno siva
```

6.3. Početna stranica

Početna stranica je izgledom zamišljena drugačije od ostalih. Ona na vrhu sadržava izbornik, na sredini naslov s kratkim opisom ispod njega te na dnu stranice sadrži ime autora stranice. Prvo ćemo napraviti pozadinu stranice, a zatim naslov i paragraf ispod njega. Za pozadinu ćemo stavit plavu sliku prijašnje obrađenu u Adobe Photoshop programu. Da to možemo ostvariti moramo napraviti *div* tag s klasom imena *background*. Nakon toga moramo napisati odgovarajući LESS kod kojim ćemo dobiti da ta stranica na svim širinama zaslona prekriva cijeli zaslon kako ne bi dobili neiskorišteni bijeli prostor. U *.less* datoteci ispod deklariranih varijabli pozovemo klasu *background*. Njoj prvo dodajemo naredbu kojom pozivamo sliku preko poveznice. Nakon toga napišemo naredbe kojom će ta slika uvijek ostati u punom zaslonu. To su *min-width* i *min-height* kojima za vrijednost dajemo 100%. Isto tako nam je bitna i pozicija slike koja mora biti fiksirana. Još možemo dodati naredbu da se ta slika ne ponavlja, nego se proširuje po cijelom zaslonu. Da stranica izgledamo malo efektnije, izrađujemo i gradijent pomoću LESS koda na način da selektoru *background* dodamo pseudo selektor *:after*. Taj selektor stavlja neki sadržaj nakon prijašnjeg sadržaja. Njemu stavimo da je pozicija apsolutna i linearni gradijent crne boje s prozirnošću vrijednosti 0.6. Također mu stavljamo vrijednost z-indeksa na -10. Z-index slaže elemente jedan na drugoga. Ukoliko ne bi imali z-index postavljen na negativnoj vrijednosti,

gradijent bi prekrivao naslov i tekst te bi korisnicima bio teže čitljiv. Ispod je primjer HTML i LESS koda za pozadinu početne stranice.

```
//HTML kod
<div class="background">
<div>

//LESS kod
.background    {
     background-image: url(img/background.png);
     min-width: 100%;
     min-height: 100%;
     margin: 0 auto;
     background-repeat: no-repeat;
     background-size: cover;
     position: fixed;

     &:after {
          content: '';
          position: absolute;
          top: 0;
          right: 0;
          bottom: 0;
          left: 0;
          background: linear-gradient(#000 0,rgba(0,0,0,0) 80%);
          opacity: 0.6;
          z-index: -10;
     }
```

Nakon što smo napravili pozadinu, možemo krenuti s izradom naslova i paragrafa ispod njega. Njih ćemo staviti u sredinu stranice i napisati kod da tamo uvijek i ostanu neovisno o rezoluciji zaslona. Prvo pišemo novi *div* klase *headline* kojega stavljamo unutar prijašnjeg *diva* klase *background*. To je potrebno jer želimo da nam se naslov nalazi na pozadini koju smo postavili na stranici. Nakon toga u novom *divu* upisujemo *h1* i *p* tag. Unutar *h1* upisujemo naslov, a unutar *p* opis tog CSS pred-procesora. Oblikovanje ćemo im dati pomoću LESS-a. Za selektor ćemo staviti klasu *headline* i u deklaracijski blok napisati vrijednosti za širinu koja je 100% te njezin z-index koji je 100. Također selektoru moramo odrediti poziciju koja je apsolutna kako bi mu mogli zadati naredbu da je od vrha stranice uvijek udaljeni 40%. Sada upotrebljavamo jednu od mogućnosti LESS-a, a to je *nesting*. Unutar deklaracijskog bloka selektora *headline* upisujemo selektor *h1*. Tako svi *h1* elementi koji se nalaze unutar klase *headline* imaju posebno oblikovanje. Nakon toga možemo upotrijebiti varijable koje smo deklarirali na početku. Upotrebljavamo varijable za font i boju naslova. Veličina fonta je 75 piksela, naslov centriramo u sredinu i dodamo mu crnu sjenu pomoću varijable. Slične postavke dodjeljujemo i paragrafu. Ovdje je primjer koda koji smo koristili:

```
//HTML kod
<div class="headline">
```

```
            <h1>LESS</h1>
            <p> "Lorem ipsum dolor sit amet, consectetur
            adipiscing elit... </p>
</div>

//LESS kod
.headline {
     position: absolute;
    width: 100%;
    top: 40%;
    z-index: 100;

     h1 {
            font-size: 75px;
            font-family: @font;
            color: @headlineColor;
            text-align: center;
            text-shadow: 3px 3px @shadowColor;
     }

     p {
            font-family: @font;
            font-size: @fontSize;
            color: @headlineColor;
            width: 50%;
            margin: auto;
            margin-top: 20px;
            text-align: center;
            animation-name: fadeIn;
            animation-duration: 5s;
     }
}
```

Prije nego napravimo animacije na početnoj stranici, preostaje nam za izraditi izbornik i podnožje (*footer*). Izbornik smo stavili u novi *div* klase *menu index* koji se nalazi unutar *diva* klase *background*. Novi *div* ima dva imena klase jer se animacija za izbornik na početnoj strani razlikuje od animacija za izbornik na ostalim stranicama. U novi *div* napravimo listu s poveznicama. U LESS-u pomoću selektora klase imena *menu* oblikujemo izbornik. Dajemo mu širinu od 80% kako bi na svim širinama zaslona izbornik popunjavao jednaki postotak stranice. Centriramo ga pomoću naredbe "*margin*: *0 auto;*" kako bi se uvijek nalazio na sredini. Pomoću *a* selektora možemo oblikovati tekst pa mu dodajemo font, veličinu fonta, boju i razmak. Korisničko iskustvo je vrlo bitno kod pregledavanja stranica pa *a* selektoru možemo dodati pseudo selektor :*hover*. Pseudo selektoru dajemo određeno oblikovanje pa kada će korisnik prelaziti mišem preko poveznica u izborniku, one će promijeniti boju i dobiti sjenu. Evo primjera koda:

```
//HTML kod
<div class="menu index">
     <ul>
            <li><a href="index.html">Naslovnica</a></li>
```

```
            <li><a href="about.html">O LESS-u</a></li>
            <li><a href="varijable.html">Varijable</a></li>
            <li><a href="nesting.html">Nesting</a></li>
            <li><a href="operacije.html">Operacije</a></li>
            <li><a href="mixins.html">Mixins</a></li>
      </ul>
</div>

//LESS kod
.menu {
      width: 80%;
      margin: 0 auto;
      margin-top: 30px;
      text-align: center;
      animation-name: fadeIn;
      animation-duration: 0.5s;

      ul {
            list-style-type: none;
            overflow: hidden;
      }

      li {
            display: inline-block;
            width: 15%;
            margin: 0 auto;
      }

      a {
          display: block;
          color: @headlineColor;
          font-size: 18px;
          font-family: @font;
          text-align: center;
          padding: 14px 16px;
          text-decoration: none;

            &:hover {
                  color: #dddddd;
                  transition: 0.2s;
                  text-shadow: 2px 2px @shadowColor;
            }
      }
}
```

Podnožje ćemo napraviti pomoću značajke mixin. U LESS-u napravimo selektor klase imena *footer*. Njemu damo određena oblikovanja koja želimo da se primjenjuju na naše podnožje. Ovdje je primjer koda mixina za podnožje koji sadrži neka oblikovanja i za ostale stranice.

```
//LESS kod
.footer {
      width: 100%;
      margin: 0 auto;
      text-align: center;
      clear: both;
      display: inline-block;
```

```
    position: absolute;
    bottom: 5px;
    margin-top: 40px;

    h5 {
        font-size: 12px;
        font-family: @font;
        color: @textColor;
    }
}
```

Zatim u *.html* datoteci, ponovno u *divu* klase *background*, napravimo *footer* tag u kojem ćemo pomoću *h5* taga napisati neki tekst. *Footer* tag će imati klasu *footer-index* jer će se razlikovati od podnožja na ostalim stranicama. Podnožje na početnoj stranici mora sadržavati drugačije oblikovanje za tekst i za animacije. Stoga stavljamo selektor klase imena *footer-index* i u njega pozivamo selektor *footer* kojeg smo prije oblikovali. Sada imamo oblikovanje selektora *footer* pa samo još dodatno napišemo naredbe koje želimo promijeniti ili dodati. Evo primjera:

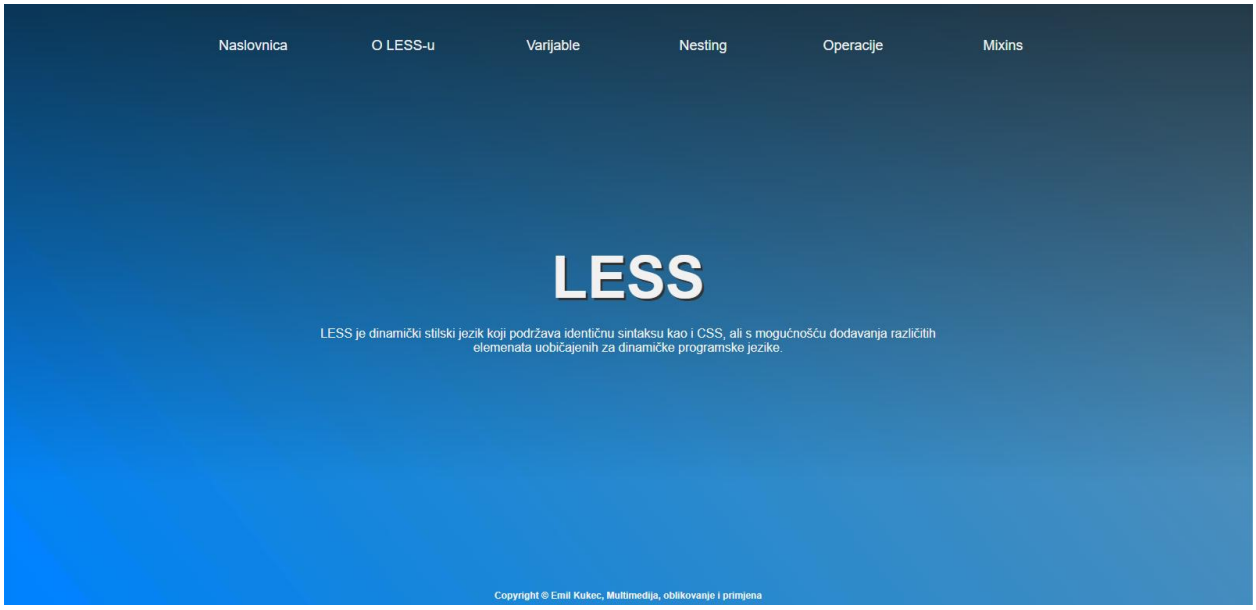
```
//HTML kod
<footer class="footer-index">
    <h5>Copyright &copy; Emil Kukec, Multimedija, oblikovanje i
    primjena</h5>
</footer>

//LESS kod
.footer-index {
    .footer;
    animation-name: fadeIn;
    animation-duration: 6s;

    h5 {
        color: @headlineColor;
    }
}
```

Za kraj nam je ostala izrada animacije. One će nam otvaranje stranica učiniti vizualno fluidnijim. Animacije se rade pomoću *@keyframesa*. On sliči selektoru, ali mu dajemo ime i naredbe unutar deklaracijskog bloka koje objašnjavaju kako će se neki element animirati. Na našoj stranici smo koristili *@keyframes* imena *fadeIn* kojem smo dali naredbe da se elementi poput naslova i teksta lagano pojavljuju. Važno je da definiramo početak animacije, a to je 0% i kraj animacije, odnosno 100%. Zatim unutar imena selektora elementa kojeg želimo animirati upišemo naredbu kojom pozivamo animaciju i njezino trajanje. Evo primjera koda:

```
@keyframes fadeIn {
  0% {
    opacity: 0;
  }
```

```
  50% {
     opacity: 0
  }

  100% {
     opacity: 1;
  }
}


//Pozivanje animacije
.selektor {
     animation-name: fadeIn;
     animation-duration: 0.5s;
}
```

Ovako izgleda stranica kada se učita:



Slika 2. Izgled početne stranice

6.4. Oblikovanje preostalih stranica

Ostalo nam je za oblikovati još 5 *.html* datoteka. One su istog oblikovanja, ali s različitim sadržajem. Pa tako imamo jednaki izbornik kao i na početnoj s plavom pozadinom, ali i naslovom ispod njega. Zatim slijedi tekst s lijeve strane i sivi pravokutnik s tekstom na desnoj strani. Na kraju stranice se nalazi podnožje.

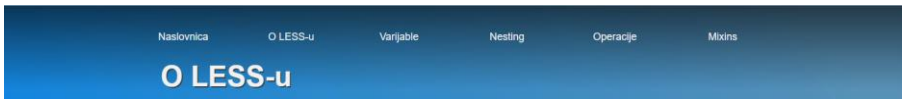
Za početak ćemo objasniti izradu izbornika. HTML kod je isti kao i za izbornik na početnoj stranici, a oblikovanje se razlikuje u nekoliko vrijednosti. Osim širine i drugačije animacije, najvažnija razlika je u njegovoj poziciji. Za svojstvo *position* dali smo mu vrijednost *fixed*. To

znači da će izbornik cijelo vrijeme biti na vrhu stranice iako mi klizimo gore-dolje po njoj. Pošto izbornik ima zadanu visinu pozadine koja se nalazi iza njega i naslova stranice, to sve izgleda preveliko i zauzima previše mjesta na zaslonu. Stoga moramo koristiti jQuery funkciju da to sredimo. Da bi jQuery kod radio, moramo povezati *.js* datoteku s *.html* datotekom. Također moramo pozvati datoteku s interneta koja aktivira jQuery. Koristit ćemo funkciju koja će prepoznati kada počinjemo kliziti po stranici i kada se spustimo ispod 75 piksela. Tada će jQuery dati izborniku novi selektor klase *fixed-menu* koji smo deklarirali u *.less* dokumentu. Pojednostavljeno možemo reći da kad se spustimo na stranici ispod 75 piksela, izbornik će se promijeniti izgled koji smo mu zadali u novom selektoru klase pomoću LESS-a. Kada ćemo se vratiti natrag na vrh stranice, izbornik će vratiti početno oblikovanje. Evo koda:

```
//LESS kod za izbornik
.menu-headline {
     min-width: 100%;
     margin: 0 auto;
     background-image: url(img/background.png);
     background-repeat: no-repeat;
     background-size: cover;
     position: fixed;

     &:after {
          content: '';
          position: absolute;
          top: 0;
          right: 0;
          bottom: 0;
          left: 0;
          background: linear-gradient(#000 0,rgba(0,0,0,0) 90%);
          opacity: 0.6;
          z-index: -10;
     }
}

//jQuery
$(document).on("scroll", function() {

     if($(document).scrollTop()> 75) {
          $("body").addClass("fixed-menu");
     } else {
          $("body").removeClass("fixed-menu");
     }

});


//LESS kod za izbornik nakon klizanja
.fixed-menu {
     .menu-headline {
          height: auto;
     }
     .menu {
          display: none;
```

```
    }
    .subtitle {
        h2 {
            font-size: 35px;
            float: left;
        }
    }
}
```

Ovo je slika kako izbornik izgleda na početku klizanja:



Slika 3. Izbornik prije klizanja

Ovo je slika kako izbornik izgleda nakon klizanja:

Slika 4. Izbornik nakon klizanja

Možemo vidjeti da nakon klizanja izbornik nestaje. Zato smo s desne strane napravili ikonu pomoću tri *div* elementa s odgovarajućim oblikovanjem. Klikom na ikonu otvara se izbornik ispod plave pozadine, ponovnim klikom on se zatvara. Time korisnicima dopuštamo da u bilo kojem trenutku imaju mogućnost odabrati drugu stranicu pomoću izbornika, ali mu ne smeta prilikom čitanja sadržaja. To smo postigli pomoću JavaScript i jQueriy koda. JavaScript kod je vrlo jednostavan. Napravili smo funkciju imena *menuFunction* i pomoću naredbe *toggle* odredili da se klikom na ikonu ona promijeni u oblik x. Ponovnim klikom ona se vraća na staro oblikovanje. S jQuerijem smo također pomoću naredbe *toggle* postigli da se klikom na tu ikonu otvara novi izbornik klase *submenu* sa svojim oblikovanjem. Važno je napomenuti da smo oblikovanje izbornika klase *submenu* stavili unutar oblikovanja klase *fixed-menu* kako bi se taj izbornik prikazivao samo kada se nalazimo na stranici ispod 75 piksela, odnosno kada klizimo prema dolje. Ovdje je primjer koda:

```
//HTML kod
<div class="container" onclick="menuFunction(this)">
          <div class="bar1"></div>
          <div class="bar2"></div>
          <div class="bar3"></div>
</div>
<div class="submenu">
     <ul>
          <li><a href="index.html">Naslovnica</a></li>
          <li><a href="about.html">O LESS-u</a></li>
          <li><a href="varijable.html">Varijable</a></li>
          <li><a href="nesting.html">Nesting</a></li>
```

```
          <li><a href="operacije.html">Operacije</a></li>
          <li><a href="mixins.html">Mixins</a></li>
     </ul>
</div>

//LESS kod
.container {
    display: none;
    cursor: pointer;
}


.bar1, .bar2, .bar3 {
    width: 35px;
    height: 5px;
    background-color: @headlineColor;
    margin: 6px 0;
    transition: 0.4s;
}

.change .bar1 {
    -webkit-transform: rotate(-45deg) translate(-9px, 6px) ;
    transform: rotate(-45deg) translate(-9px, 7px) ;
}

.change .bar2 {opacity: 0;}

.change .bar3 {
    -webkit-transform: rotate(45deg) translate(-8px, -8px) ;
    transform: rotate(45deg) translate(-8px, -7px) ;
}

.submenu {
     display: none;
     background: linear-gradient(#fff 0,rgba(0,0,0,0) 100%);
     height: 800px;
     position: absolute;
     top: 35px;
     ul {
          width: 76%;
          margin: auto;
     }

     a {
          color: @textColor;

          &:hover {
               color: #7c7c7c;
               transition: 0.2s;
               text-shadow: none;
          }
     }
}

//JavaScript kod
function menuFunction(x) {
    x.classList.toggle("change");
}
```

```
//jQuery kod
$(document).ready(function(){
    $(".container").click(function(){
        $(".submenu").toggle();
    });
});
```

Zatim krećemo na izradu sadržaja. Prvo ćemo napraviti *div* klase *content* koji će biti širine 65% cijele stranice i centriran na sredini. Unutar njega ćemo napraviti dva *diva* klase *article* i *box*. *Article* će sadržavati sve definicije, objašnjenja i primjere LESS-a, dok je *box* ovdje da popuni desnu stranu sadržaja sa sivim pravokutnikom koji sadrži tekst o završnom radu. *Article* će također biti širine 65% i nalazit će se s lijeve strane kako bi se postigla čim bolja čitljivost. Sadržaj će biti podijeljeni i na podnaslove kako bi se postiglo čim bolje korisničko iskustvo. Evo primjera koda za *div* klase *article*:

```
//HTML kod
<div class="content>
     <div class="article">
     //sadrzaj
     </div
     <div class="box">
     //sadrzaj
     </div>
</div

//LESS kod
.article {
     width: 65%;
     float: left;
     padding-top: 180px;
     padding-bottom: 100px;
     animation-name: fadeIn;
     animation-duration: 1s;

     h4 {
          font-size: 23px;
          font-family: @font;
          color: #2e6f9d;
          margin-top: 80px;
     }

     p {
          font-family: @font;
          font-size: 15px;
          text-align: left;
          line-height: 1.6;
          margin-top: 30px;
     }
}
```

U sadržaju stranica, kao što sam već napomenuo, nalaze se i primjeri kodova. Da bih njih prikazali što točnije i čitljivije na svim širinama zaslona, jedno od boljih rješenja je korištenje

highlights.js dodatka. On nam na malim širinama zaslona čak omogućuje klizanje lijevo-desno ukoliko je kod preširok. Isto tako ima mogućnosti bojanja koda, mijenjanje fonta, veličine, pozadine i slično. Sve u svemu omogućuje mnogo optimizacije. Njega je dovoljno skinuti, otpakirati i spremiti datoteke u našu mapu s projektom. Tada u *.html* datoteke kopiramo dvije poveznice kojima povežemo dokument i on automatski prepoznaje kod i oblikuje ga. Moram naglasiti da je važno kod u *.html* datoteci staviti u *code* i *pre* tagove. Evo primjera poveznica:

```
<script src="highlight.pack.js"></script>
<script>hljs.initHighlightingOnLoad();</script>
```

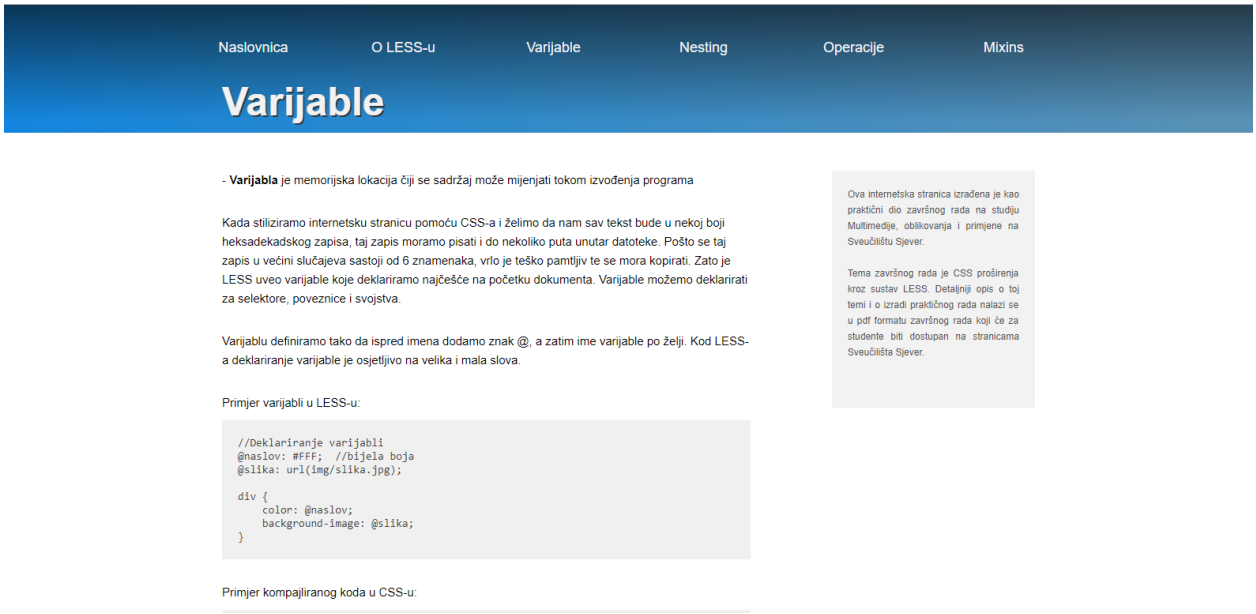
Div klase *box* se nalazi s desne strane internetske stranice i razbija privid nepreglednog i nečitljivog teksta. On je pravokutnog oblika s tekstom koji opisuje internetsku stranicu i čemu služi. Na ovom *divu* sam iskoristio jednu od najboljih značajki LESS-a, a to je mixin. Napravio sam mixin funkciju koja *divu* klase *box* mijenja boju pozadine ovisno o boji teksta. Pa tako ukoliko nam je tekst tamnije boje poput tamno plave ili crne, pozadina će biti svijetlo siva. Ukoliko želimo da nam je tekst svjetlije boje poput bijele, pozadina će biti plave boje kao i pozadina izbornika. Točnije rečeno, funkcija prepoznaje svjetlinu teksta i ukoliko je ona veća od 50%, pozadina će biti tamnija i obratno. Dovoljno je kod oblikovanja *diva* klase *box* dodati selektor za mixin funkciju koja je imena *mixin* i ona će raditi. Evo primjera koda:

```
//HTML kod

<div class="box">
     <p> //sadrzaj
     </p>
</div>

//LESS kod
.mixin (@a) when (lightness(@a) >= 50%) {
     color: @a;
     background-color: #2e6f9d;
}
.mixin (@a) when (lightness(@a) < 50%) {
     color: @a;
     background-color: @headlineColor;
}

.mixin (@a) {
    width: 100%;
    height: auto;
    padding: 20px;
}


.box {
     width: 25%;
     float: right;
```

```
        margin-top: 210px;
        animation-name: fadeIn;
        animation-duration: 1s;

        p {
              font-size: 12px;
              line-height: 20px;
              font-family: @font;
              color: @textColor;
              text-align: justify;
              .mixin (@textColor);
              min-height: 300px;
        }
}
```

Za kraj nam je preostala izrada podnožja. On se razlikuje u nekoliko manjih sitnica poput boje teksta od podnožja na početnoj stranici. U prijašnjim odlomcima sve je detaljnije objašnjeno.

Ovdje je slika kako izgledaju ostale stranice praktičnog dijela završnog rada. Opet napominjem da ih je 5, ali su sve jednake osim po sadržaju.



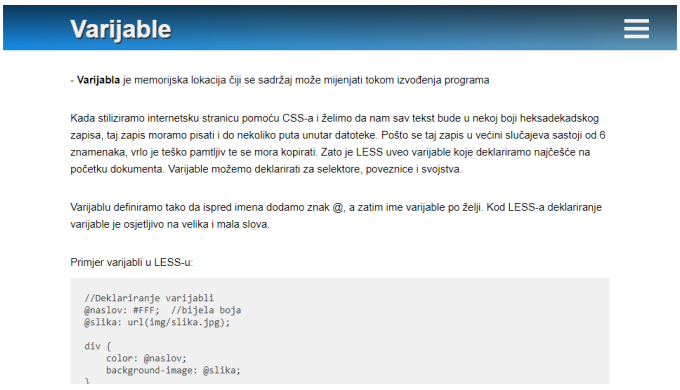
Slika 5. Izgled stranice Varijable

6.5. Izrada responzivnog web dizajna

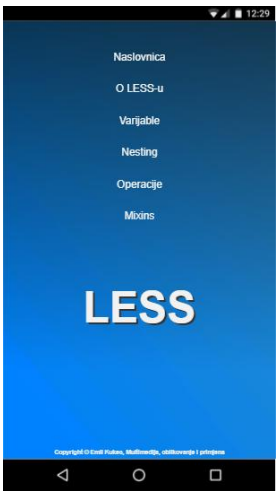
Stranicu moramo prilagoditi zaslonima svim širinama, odnosno mobilnom prikazu. Korisnici masovno koriste mobilne uređaje za otvaranje stranica, a neki od njih su čak prestali koristiti računala za pregledavanje interneta. Možemo predvidjeti da bi se u neko skorije vrijeme više

internetskih stranica otvaralo preko mobitela, nego preko računala. [17] Zato moramo prilagoditi naš praktični dio završnog rada svim korisnicima pa tako i onim na mobilnim uređajima. Da bi internetsku stranicu prilagodili mobilnom prikazu, koristimo se medijskim upitima (*media queries*). Koristit ćemo 3 zadane maksimalne širine. To su 1200 piksela, 960 piksela i 560 piksela. Širina od 1200 i 960 su prilagođene za tablete, dok je širina od 560 piksela prilagođena za mobilne uređaje. Nećemo objašnjavati svaki dio koda jer je dosta dugačak, a radi se samo o izmjenama postojećeg oblikovanja. Pa tako se mijenja veličina teksta, raspored izbornika ili se izbacuju neki manje važne elementi poput *diva* klase *box* da ne smeta korisničkom iskustvu. Ovo nije kompliciran dio pisanja koda, ali treba mnogo vremena da se svaki dio prilagodi na što bolji način. Evo primjera slika za prikaz na tabletu i mobilnom uređaju:

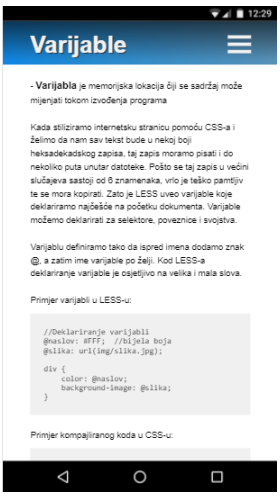
Slika 6. Izgled početne stranice - tablet



Slika 7. Izgled stranice Varijable - tablet



Slika 8. Izgled početne stranice - mobilni prikaz



Slika 9. Izgled stranice Varijable - mobilni prikaz

6.6. Prijenos praktičnog rada na server

Prijenos praktičnog dijela rada ćemo izvršiti preko WinSCP-a. Taj program sam opisao u prethodnim odlomcima, ali da ukratko ponovimo; on nam služi da prebacimo datoteka s našeg računala na internetski server. Za početak odaberemo da nam je protokol datoteka FTP, naziv poslužitelja je arwen.unin.hr i upišemo naše korisničko ime i lozinku. Nakon toga uđemo na server u našu mapu. Program ima grafičko sučelje pa se vrlo jednostavno snalaziti u njemu. Nakon što smo u našoj mapi, odaberemo mapu na našem računalu i prebacimo je na server te je s time naš posao gotov. Ovdje je poveznica na praktični dio završnog rada na temu CSS proširenja kroz sustav LESS:

http://arwen.unin.hr/~emkukec/zavrsni_rad/index.html

7. Zaključak

CSS pred-procesori svojom su pojavom proširili mogućnosti CSS-a i olakšali programerima pisanje i mijenjanje koda. Kod se piše brže, jednostavniji je i vrlo je lako promijeniti veliki dio koda na samo jednom mjestu u nekoliko sekundi. U praksi rijetko koji front-end developer ne koristi neki od CSS pred-procesora. To je uobičajeno u tom poslu jer se nikome ne isplatili gubiti vrijeme na nepotrebno pisanje i ponavljanje istog koda nekoliko puta na jednom projektu.

LESS je jedan od najpoznatijih i najkorištenijih CSS pred-procesora. Vrlo je jednostavan za korištenje zbog jednake sintakse kao i kod CSS-a, a opet nudi mnogo mogućnosti kako poboljšati našu efikasnost u pisanju koda. Početnicima neće biti problem savladati LESS u samo nekoliko sati. Baš zbog njegove sintakse jednake CSS-u i vrlo jednostavne implementacije, preporučljiv je početnicima među konkurencijom. Vjerujem da bi taj pred-procesor koristilo puno programera samo radi varijabli, a tu su još operacije, nesting, funkcije i tako dalje. Kada se sve to isproba i koristi, rijetko tko se vraća pisanju čistog CSS-a za oblikovanje internetskih stranica. Kao što je Bass Jobsena u svojoj knjizi napisao: "LESS nam pomaže pisati CSS kod bez da ponavljamo sami sebe." [5]

Praktični dio završnog rada je internetska stranica napravljena sa svrhom učenja i pomoći početnicima kod savladavanja LESS-a. Na njoj se nalaze sve najvažnije značajke koje su potrebne početnicima. Stranica je oblikovana pomoću LESS-a s većinom njegovih značajki.

U Varaždinu, _____

_____

8. Literatura

[1] Eric A. Meyer: CSS The Definitive Guide, O'Reilly Media Inc., 3rd edition, 2007.
[2] http://www.webtech.com.hr/css.php, dostupno 14.06.2017.
[3] https://www.slant.co/topics/217/~best-css-preprocessors-postprocessors, dostupno 15.06.2017.
[4] http://gledaj-uci.com/text-tutoriali/sto-je-i-kako-koristiti-less-js-sa-wordpressom/, dostupno 15.06.2017.
[5] Bass Jobsen, Less Web Development Essentials, Packt Publishing Ltd., 2014.
[6] http://hardwarebase.net/teme/11739/less-za-efikasnije-i-pametnije-css-stilove, dostupno 15.06.2017.
[7] http://www.tutorijali.net/teorija-programiranja/varijable, dostupno 16.06.2017.
[8] http://lesscss.org/features/#variables-feature-variable-interpolation, dostupno 17.06.2017.
[9] http://lesscss.org/features/#features-overview-feature-operations, dostupno 18.06.2017.
[10] http://lesscss.org/features/#mixin-guards-feature, dostupno 18.06.2017.
[11] https://www.sitepoint.com/understanding-less-guards-loops/, dostupno 18.06.2017.
[12] https://www.gadgetdaily.xyz/sass-v-less-v-stylus-the-pros-and-cons/,dostupno 23.06.2017.
[13] https://en.wikipedia.org/wiki/Sass_(stylesheet_language), dostupno 23.06.2017.
[14] http://sass-lang.com/guide, dostupno 23.06.2017.
[15] http://stylus-lang.com/, dostupno 23.06.2017.
[16] https://codemyviews.com/blog/mobilefirst, dostupno 20.08.2017.

Popis slika

Slika 1. Datoteke projekta 17
Slika 2. Izgled početne stranice 23
Slika 3. Izbornik prije klizanja 25
Slika 4. Izbornik nakon klizanja 25
Slika 5. Izgled stranice Varijable 29
Slika 6. Izgled početne stranice - tablet 30
Slika 7. Izgled stranice Varijable - tablet 30
Slika 8. Izgled početne stranice - mobilni prikaz 31
Slika 9. Izgled stranice Varijable - mobilni prikaz 31

Sveučilište
Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, EMIL KUKEC *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom CSS PROŠIRENJA KROZ SUSTAV LESS *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Emil Kukec
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, EMIL KUKEC *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom CSS PROŠIRENJA KROZ SUSTAV LESS *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Emil Kukec
(vlastoručni potpis)