CSS pretprocesori

Kotris, Dominik

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:234726

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-09-17**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij Računarstvo

CSS PRETPROCESORI

Završni rad

Dominik Kotris

Osijek, 2017.

SADRŽAJ

1. UVOD

U ovom završnom radu razvijat će se frontend strana web aplikacije za komprimiranje slika na kojoj je cilj prikazati moć i modularnost preprocesora u odnosnu na standardni CSS pri oblikovanju vizualnih elemanata web aplikacije.

Kako je fokus web aplikacije na komprimiranju slike, tako je nužno pojednostaviti i optimizirati sučelje za jednostavno korištenje, tj. omogućiti korisniku da željenu funkcionalnost aplikacije može obaviti intuitivno bez nepotrebnih elemenata unutar sučelja koji će stvarati distrakciju. Da bi se izbjegao prikaz nepotrebnih elemenata unutar sučelja prvo je potrebno napraviti wireframe, odnosno prototip dizajna sučelja kako bi se odredila hijerarhija elemenata, a nadalje i rasporedile određene funkcionalnosti aplikacije.

Za izradu prototipa dizajna koristit će se program Sketch koji omogućava rad sa vektorskom grafikom, a za implementaciju preprocesorske tehnologije open source tekstualni editor Atom. U njemu će biti razvijen cjelokupni frontend development dio koji će kasnije biti dostupan kao datoteke ekstenzija .html i .css kako bi ih preglednik mogao pravilno prikazivati.

1.1. Zadatak završnog rada

Kao zadatak završnog rada teorijski je bilo potrebno spomenuti i opisati najpoznatije CSS preprocesore te usporediti njihove mogućnosti. Za praktični dio razvijeno je jednostavno web sučelje aplikacije na kojoj su se demonstrirale neke od mogućnosti odabranog CSS pretprocesora - *SASS*. Uloga autora ovog završnog rada bila je napraviti *frontend* dio (sučelje) aplikacije za komprimiranje slika ImageClinic koje je dostupna na sljedećoj poveznici - https://github.com/Levara/ImageClinic [5].

Aplikacija funkcionira na način da se u nju učitaju slike, te se pritiskom na tipku optimizacije slike provlače kroz algoritam za komprimiranje koji će na kraju dovesti do smanjenja veličine slike uz manje gubitke kvalitete. Nakon komprimiranja, moguće je odabrati sliku gdje će aplikacija prikazati informacije vezane uz kvalitetu i veličinu slike.

2. UVOD U CSS PREPROCESORE

CSS preprocesori u svakodnevnom frontend developmentu koriste se već određeni niz godina. U prvim implementacijama imali su samo nekoliko funkcionalnosti koje su bile naprednije u odnosu na čisti CSS. Danas su nezaobilazni dio razvoja kompleksnih web aplikacija i bilo kakvih naprednijih sučelja jer omogućavaju modularniji, robisniji, napredniji i brži razvoj sučelja. Ključne funkcionalnosti koje preprocesore diferenciraju u odnosnu na CSS su mogućnost korištenja varijabli, operatora, interpolacija, funkcija, for petlji, mixina i neke druge manje bitne funkcionalnosti.

Najpoznatiji preprocesori su *SASS*, *LESS* i *STYLUS* sa zadnjim dostupnim verzijama:

- Sass: **v3.4.22**
- Less: **v2.7.2**
- Stylus: **v0.54.5**

2.1. Migracija na preprocesore

Sam CSS je prilično nepotpun i dosta primitivan. Pisanje funkcije ili višestruko korištenje definicije neke varijable teško je postići. Na velikim projektima i kompleksnim sučeljima, dugoročno održavanje aplikacije predstavlja najveći problem. S ciljem poboljšanja i konzistentnijeg pisanja CSS-a postojalo je nekoliko pristupa poput odvajanja koda u manje cjeline kako bi se one umetnule u glavni dokument. Ovaj pristup pomogao je razdvojiti komponente na više manjih cjelina, ali nije riješio problem ponavljanja komponenata i lakšeg održavanja koda.

Koristeći napredne funkcionalnosti preprocesori pomažu programerima u pisanju održivog, modularnog i konzistentnog CSS koda. Glavna ideja i namjena korištenja CSS preprocesora je povećanje produktivnosti i smanjanje linija koda za vrijeme izrade nekog projekta.

3. KREIRANJE KORISNIČKOG SUČELJA

Da bi se došlo do faze razvoja *frontend* dijela aplikacije, prvo je potrebno raspisati sve funkcionalnosti i na papiru napraviti par skica sučelja kako bi se dobio jasan pregled i ideja kako će sučelje izgledati te kako će pojedini elementi biti raspoređeni. Ovaj korak štedi vrijeme jer se na brz i efiksan način može napraviti pregled cijeloga sučelja.

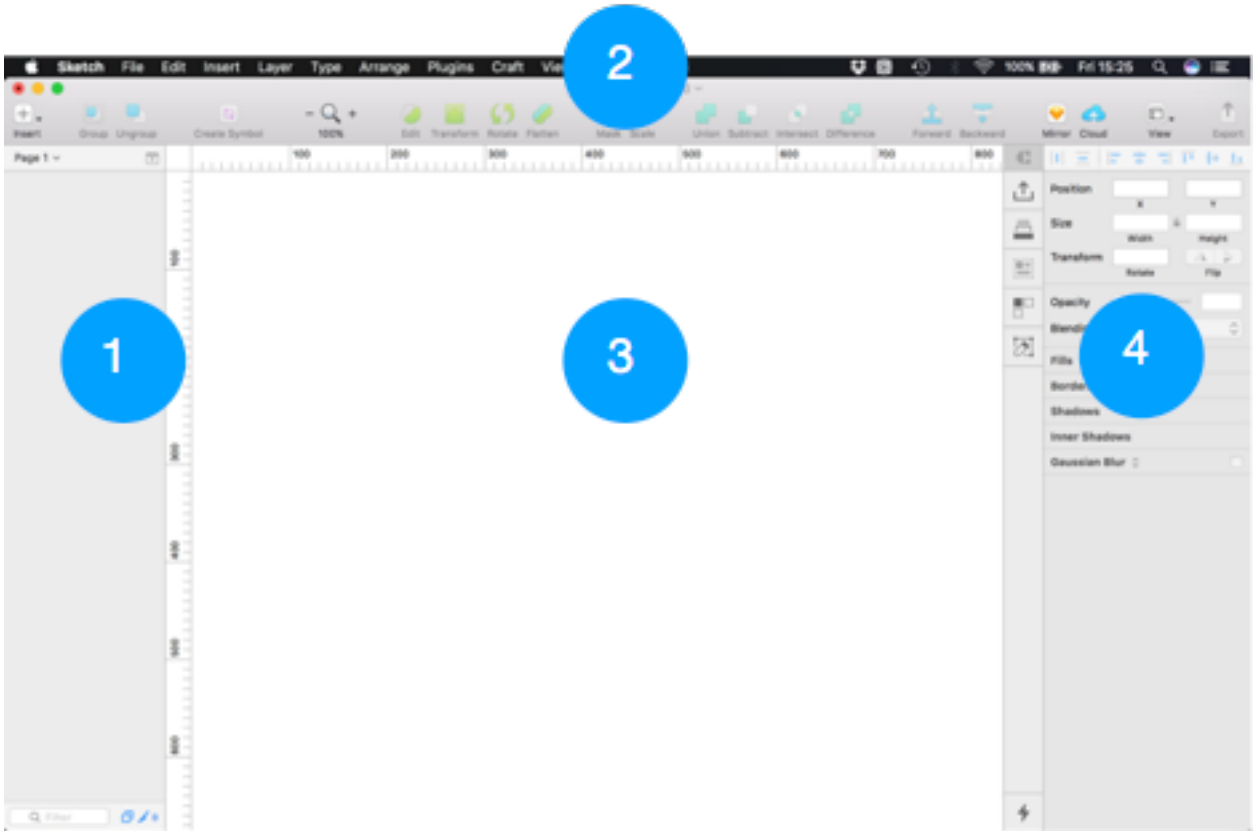
3.1. Popis funkcionalnosti

U ovoj fazi najbitnije je popisati sve funkcionalnosti aplikacije na papir ili u digitalni oblik te ih složiti po prioritetnoj listi korištenja. Takvo sortiriranje funkcionalnosti dat će nam jasan pregled na koje elemente sučelja trebamo staviti fokus, a koji elementi trebaju biti u sekundardnom planu.

3.2. Program za obradu vektorske grafike Sketch

Program Sketch dostupan je samo za OSX operacijske sustave. Iznimno je postao popularan unazad par godina zbog svoje jednostavnosti i velikih mogućnosti. Napravljen je sa ciljem da omogući dizajnerima jednostavnu manipulaciju i izgradnju vektorskih oblika kao što se može u programu Adobe Illustrator. Problem sa Illustratorom je taj što je previše robustan i za web dizajn je potrebno samo nekoliko alata. Na tih nekoliko alata se Sketch i fokusirao i u kratkome vremenu postao stanard za dizajn i korisnička sučelja za web.

Neke od glavnih funkcionalnosti programa Sketch su: kreiranje vektorskih oblika, laka manipulacija vektorskih objekata, predefinirane veličine raznih uređaja poput monitora, mobitela, tableta, televizora i sl. što u startu omogućava lakše postavljanje novoga projekta, grupiranje objekata, lista slojeva, obrada fotografije, "*live view*" prikaz radnoga prostora na mobilnim i tablet uređajima, kreiranje i spremanje palete boja i nekolicina drugih korisnih funkcionalnosti.



S.l. 3.1. *Sučelje programa Sketch*

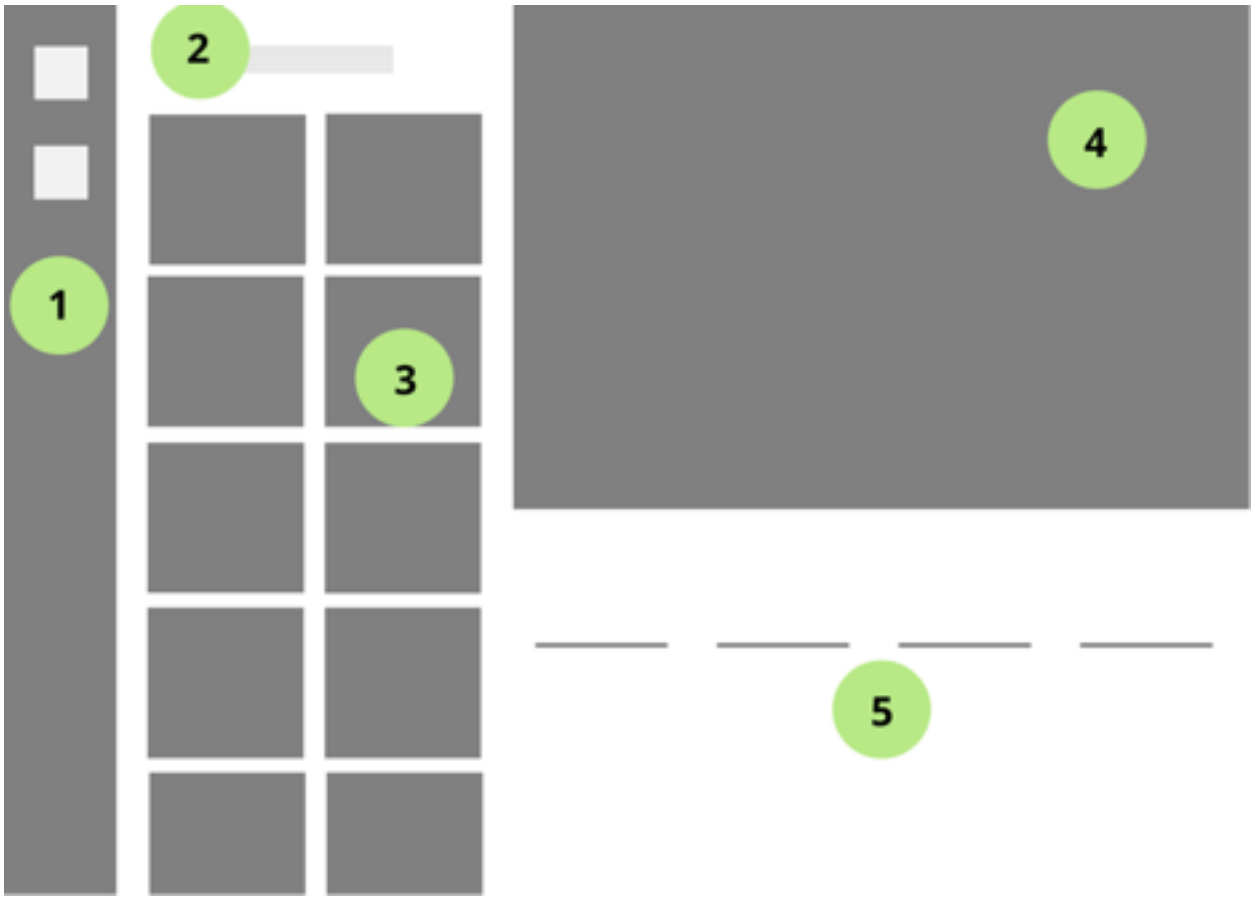
Popis elemenata sučelja programa Sketch po brojevima prema Slici 3.1.

1. Popis slojeva radnoga prostora
2. Alatna traka
3. Radni prostor
4. Vrijednosti i parametri trenutno odabranog alata

U Sketchu je kreiran wireframe koji će biti korišten kao baza sučelja. Nadogradnjom wireframe-a dobiva se pregled i raspored elemenata sučelja kao i uvid u korištenje pojedine funkcionalnosti. Cilj je dobiti "*high-fidelity*" wireframe, odnosno prototip dizajna prema kojemu će se iskodirati sučelje u web tehnologijama HTML, CSS i po potrebi Javascript.

3.3. Izrada wireframe-a za prikaz sučelja aplikacije

U ovoj fazi razvoja projekta u Sketchu je napravljen jednostavn prikaz pozicije elemenata sučelja kako bi se odmah moglo prijeći na fazu “development-a”, odnosno kodiranja vizualnog dijela sučelja.



S.l. 3.2. *Lo-fidelity wireframe za prikaz rasporeda elemenata sučelja*

Popis elemenata sučelja programa Sketch po brojevima prema Slici 3.2.

1. Sidebar navigacija sa 2 linka
2. Tipka (button) pomoću koje će se pokretati algoritam za komprimiranje slika
3. Skup slika
4. Prikaz odabrane komprimirane slike
5. Sekcija sa specificiranim informacijama o odabranoj komprimiranoj slici

3.4. Izrada elemenata sučelja pomoću opisnog jezika HTML

Nakon definiranih pozicija elementa sučelja koje će aplikacija ImageClinic sadržavati, pomoću opisnog jezika HTML definirani su semantički elementi sučelja na koje će se kasnije primjeniti određeni stil pomoću preprocesorskog jezika *SASS.*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Photo Compression Tool</title>

    <!-- Custom CSS -->
    <link rel="stylesheet" href="style/style.css">

    <!-- Font Awesome CDN -->
    <script src="https://use.fontawesome.com/98a1849afa.js"></script>
</head>
<body>

```

S.I. 3.3. *Prikaz <head> dijela HTML koda; povezivanje sa eksternim style.css stilom*

Na Slici 3.3. može se vidjeti <head> dio HTML dokumenta u kojemu je povezan eksterni dokument za uređivanje elemenata "style.css". Bitno je napomenuti kako je taj dokument već prethodno "*preveden*" iz *SASS* dokumenta u kojemu su pisana sva svojstva elemenata sučelja. Iduće poglavlje sadržava više detalja o procesu "*prevođenja*" iz SASS u CSS i općenito o funkcionalnostima SASS jezika.

Također, kako je vidljivo iz Slike 3.3. unutar <head> tag-a nalazi se i "*library*" za "Font Awesome" svg font koji je korišten u svrhe prikaza linkova u *sidebar* navigaciji.

```
<div class="app--layout">

    <!-- NAVIGATION SIDEBAR -->
    <nav class="sidebar-navigation">
        <div class="navigation__items">
            <div class="photos item">
                <i class="fa fa-picture-o fa-2x" aria-hidden="true"></i>
            </div>
            <div class="settings item">
                <i class="fa fa-cog fa-2x" aria-hidden="true"></i>
            </div>
        </div>
    </nav>
```

S.l. 3.4. *Prikaz 1. dijela klase "app—layout" koja prikazuje elemente korištene u sidebar-u*

U ovome dijelu prema Slici 3.4. definirana je struktura koja će se prikazivati kao *sidebar* navigacija sa 2 Font Awesome "*sprite-a*" koji će služiti kao linkovi za određenu akciju prilikom pritiska miša na link.

```
    <!-- IMAGE LIBRARY -->
    <div class="library">
        <div class="optimize__section">
            <a class="button" href="#">Optimize</a>
        </div>
        <div class="library__items">
            <div class="img__wrapper optimized">
                <img src="http://bit.ly/2wCbAFY" alt="optimized-image">
            </div>
            <div class="img__wrapper not__optimized">
                <img src="http://bit.ly/2gY0PeF" alt="not_optimized-image">
            </div>
        </div>
    </div>
```

S.l. 3.5. *Prikaz 2. dijela klase "app—layout" koja prikazuje elemente korištene za prikaz slika*

U ovome dijelu prema Slici 3.5. definirana je struktura koja će se prikazivati kao prikaz galerije slika i 1 tipke koje će služiti kao "okidač" za pokretanje algoritma komprimiranja koji će se primjeniti na svim nekomprimiranim slikama unutar galerije.

```
<!-- IMAGE PREVIEW -->
<div class="preview">
    <div class="image__preview">
        <img src="http://bit.ly/2jEVoT1" alt="compressed-image_preview" >
    </div>
    <div class="image__data">
        <p><b>Image resolution:</b> 3580x2960px</p>
        <p><b>Original size:</b> 13 MB</p>
        <p><b>Compressed size:</b> 9 MB</p>
        <p><b>Compression rate:</b> 87%</p>
    </div>
</div>
```

S.l. 3.6. *Prikaz 3. dijela klase "app—layout" koja prikazuje odabranu sliku i njene detalje*

U ovome dijelu prema Slici 3.6. definirana su 2 bloka tipa koji se nalaze unutar klase "preview". Prvi blok element klase "image__preview" predstavlja prostor u kojemu će se prikazivati odabrana slika iz galerije, dok drugi blok element klase "image__data" predstavlja sekciju gdje će biti navedene karakteristične informacije o odabroj slici.

Primjer karakterističnih informacija sadržavat će:

a) Rezoluciju slike izraženu u pixelima - px
b) Originalnu veličinu slike prije komprimiranja
c) Veličinu slike nakon komprimiranja
d) Razinu komprimiranja izraženu u postotcima - %

4. KORIŠTENJE CSS PREPROCESORA

Budući da se u praktičnome dijelu rada izradila web aplikacija, tehnologije koje su najviše korištene su HTML za definiranje semantičkih elemenata sučelja i CSS koji se koristio za oblikovanje tih elemenata.

Važno je znati kako CSS preprocesori nemaju drugačiju sintaksu u odnosu na čisti CSS. Kao što i samo ime kaže oni se "preprocesiraju". Odnosno "kompajliranjem" se prevode u čisti CSS koji na kraju definira oblikovanje i stil elemanata aplikacije.

4.1. Prevođenje SASS preprocesora u CSS

Koristeći naredbu "*sass-watch"* preko Terminala može se narediti računalu da pazi na promjene unutar preprocesora (prilikom spremanja dokumenta) te da ih automatski prevodi u CSS kod koji je povezan za HTML dokumentom.

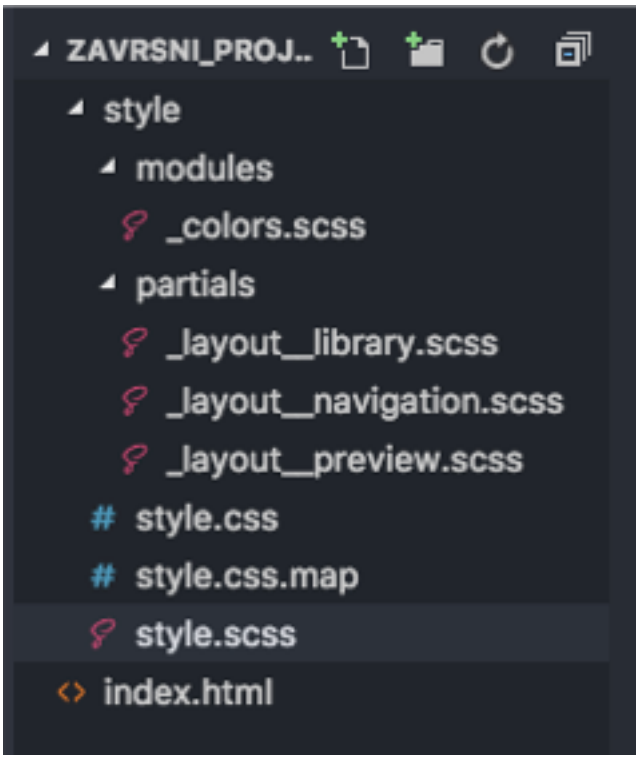
Prvo je potrebno instalirati SASS na računalo. To se radi preko Terminala kako je napisano u koraku 1, a potom se pokreće naredba *sass-watch*. Pomoću druge naredbe naređuje se računalu da dokument style.scss iz foldera style kompajlira u dokument style.css koji se nalazi u istom folderu.

Naredbe u Terminal aplikaciji koje su potrebne za prevođenje SASS koda u CSS:

1. **npm install -g sass-watch**
2. **sass-watch style/style.scss:style.css**

4.2. Postavljanje SASS arhitekture

Prilikom izrade novoga projekta vrlo je bitno dobro postaviti arhitekturu kako bi se odmah u početku napravila dobra podloga za lakše održavanje programskog koda u budućnosti.
U slučaju ovoga završna rada arhitektura se sastoji od glavnoga .scss (SASS) dokumenta i više modula koji se ubacuju u glavni modul. Takav modularni pristup omogućava lakše izmjene koda potrebne za određeni modul, kao i za lakše snalaženje te čitljivost koda općenito.



S.l. 4.1. *Strukturalni prikaz arhitekture projekta*

Iz Slike 4.1. vidljivo je kako su SASS dokumenti raspoređeni u nekoliko cjelina. Glavni .scss dokument nalazi se u Style folderu zajedno za prevedenim ("kompajliranim") CSS dokumentom dok se .scss moduli nalaze unutar podjeljenih foldera Modules i Partials.

Folder Modules sadžava dokument "_colors.scss" koji u sebi sadrzava imena varijabli boja, a folder Partials u sebi sadrzava 3 odvojena .scss dokumenta u kojima su spremljena svojstva HTML elemenata za određenu sekciju/dio prikaza web aplikacije. Tako "_layout__navigation.scss" sadrzi kod uređenih klasa za "sidebar" navigaciju, "_layout__library.scss" za galeriju slika, a "_layout__preview.scss" uređene klase za prikaz odabrane slike i njene informacije.

4.3. Povezivanje .scss modula

Budući da je stil elemenata sučelja web aplikacije podjeljen u nekoliko SASS modula, njih je potrebno međusobno povezati kako bi se prevođenje ("kompilacija") u CSS kod odradila korektno. Povezivanje se napravilo na način umetanja manjih modula koji su vezane za određene sekcije sučelja u glavni *style.scss* dokument.

```scss
// Modules
@import "modules/colors.scss";

// Partials
@import "partials/layout__navigation.scss";
@import "partials/layout__library.scss";
@import "partials/layout__preview.scss";

body {
    margin: 0;
    background-color: rgb(248, 247, 245);
    font-family: Helvetica;
}

img {
    max-width: 100%;
    height: auto;
}

/* CSS Grid Code */
.app--layout {
    display: grid;
    grid-template-columns: 80px 480px 1fr;
    grid-template-rows: 100vh;
}
```

S.I. 4.2. *Primjer umetnutih modula u glavni .scss dokument*

Nakon sto su se moduli umetnuli u glavni .scss dokument, nakon spremanja automatski se pokreće proces prevođenja (“kompajliranja”) u Terminal-u kojega se ranije pripremilo da gleda na promjene unutar dokumenta, te da se prilikom spremanja automatski pokrene skripta. Nakon prevođenja (“kompajliranja”) sav kod koji je nalazi unutar modula zajedno sa kodom koji se nalazi u style.scss dokumentu je preveden i spremljen u obliku nešto dužeg, ali standardnijeg CSS koda.

U ovoj fazi razvoja projekta, sučelje poprima svoj konačni oblik te se može obaviti interakcija sa određenim elementima poput tipke za pokretanje komprimiranja ili listanja galerije slike. Bitno je usredotočiti se na neke dijelove SASS koda koji je olakšao i ubrzao razvoj sučelja.

U idućoj sekciji nalazi se komparativna analiza pojedinih dijelova SASS i CSS koda. Cilj je prikazati na koji način su se koristile neke od SASS funkcionalnosti te kako se u konačnici broj linija koda za stiliziranje pojedinih elemenata razlikovao između SASS i CSS koda.

4.4. Prikaz korištenih funkcionalnosti SASS preprocesora

Imports

Umjesto korištenja jednog velikog dokumenta, podjeljivanje koda u više manjih cjelina (modula) korisno je za izražavanje specifičnih deklaracija poput varijabli ili globalnih funkcija, a u konačnici i učinkovitija metoda za nadzor koda i buduće održavanje. Slične dijelove koda moguće je grupirati u foldere radi lakšega snalaženja i tako ih umetnuti (“importati”) u glavni SASS dokument.

```
1  // Modules
2  @import "modules/colors.scss";
3
4  // Partials
5  @import "partials/layout__navigation.scss";
6  @import "partials/layout__library.scss";
7  @import "partials/layout__preview.scss";
8
```

S.I. 4.3. *Primjer Import funkcionalnosti*

Varijable

Varijable su jedna od najčešće korištenih funkcionalnosti preprocesorskih jezika. Kako se svaki puta nebi morao ponavljati isti kod za dobivanje željene vrijednosti kroz cijeli dokument, varijable omogućuje da se jednom globalno ili lokalno deklarira određena vrijednost za specifični tip podatka koji se kasnije jednostavno poziva unutar cijeloga dokumenta ukoliko ima potrebe za korištenjem istih vrijednosti.

Najbolju uporabu varijable pronalaze u deklaraciji heksadekadskih vrijednosti za selektiranje određenih boja, veličine poput visine ili širine elemenata izraženih u statičnim vrijednostima - *px* ili responzivnim - *em*/*rem.* Varijable se još koriste za definiranje tipografske skale, fontova, specifičnih vrijednosti poput obruba *border* i sličnih svojstava koje često moraju imati točno definiran izgled za uporabu u cijelome projektu.

S.l. 4.4. *Hex vrijednosti boja definirane kao varijable*

Ugnježđivanje (nesting)

CSS nema mogućnost pisanja koda kroz vizualnu hijerarhiju koja je vrlo bitna kada se radi sa *parent* i *child* selektorima stoga je jako dobro koristiti preprocesorske jezike kako bi se *child* selektori mogli uvlačiti ispod parent *selektora* i pratiti hijerarhiju HTML elemenata.

Primjer *Nesting* funkcionalnosti pokazan je na primjeru ispod (Slika 4.5.) koji opisuje elemenate kojima je stvorena *sidebar* navigacija sučelja web aplikacije. Vidljivo je kako se unutar *parent* selektora "navigation__items" nalazi selektor "item" za koji su definirana svojstva.

```
6
7   .navigation__items {
8       display: flex;
9       flex-direction: column;
10      text-align: center;
11
12      .item {
13          height: auto;
14          padding: 20px 0;
15          transition: all .35s ease-in-out;
16
17          &:hover {
18              background-color: #22252E;
19              cursor: pointer;
20          }
21
22          &.active {
23              background-color: #22252E;
24          }
25      }
26
27  }
28
```

S.I. 4.5. *Nesting selektora i & (ampersand) nesting*

Ampersand (&) nesting

Ampersand nesting omogućava korištenje znaka "&" u svrhu pozivanja imena klase *parent* selektora kako bi se olakšalo pozivanje roditeljskih klasa. Primjer koda sa Slike 4.5. će se prevesti u CSS kod sa Slike 4.6. Može se vidjeti kako je SASS preprocesor omogućio znatno jasniju hijerarhijsku strukturu i manje ponavljanja imena klasa prilikom selekcije istih. U velikim projektima prilikom potrebe za velikom selektivnošću elemenata CSS ne nudi adekvatno rješenje te je programeru iznimno teško pratiti ovakvu hijerarhijsku strukturu stoga je *Nesting* obavezan dio konvencije pisanja programskog koda na svakome projeku kojemu je cilj laka i dugoročna održivost.

```
.navigation__items {
  display: flex;
  flex-direction: column;
  text-align: center;
}

.navigation__items .item {
  height: auto;
  padding: 20px 0;
  transition: all .35s ease-in-out;
}

.navigation__items .item:hover {
  background-color: #22252E;
  cursor: pointer;
}

.navigation__items .item.active {
  background-color: #22252E;
}
```

***S.l. 4.6**. Prevedeni CSS kod za SASS kod sa Slike 4.5.*

4.5. Naprednije funkcionalnosti SASS preprocesora

Matematičke operacije

Preprocesori (SASS, Stylus) podržavaju svojstvo aritmetičkih funkcija i međusobne pretvorbe jedinica. Kao dodatak na jednostavne matematičke funkcije, preprocesori podržavaju i kompleksnije operacije poput zaokruživanja na veći broj, zaokruživanja na točnu decimalu, izvlačenja maksimalne ili minimalne vrijednosti iz liste i slično. Primjer nekih matematičkih operacija implementiranih unutar preprocesorkih jezika može se vidjeti na Slici 4.7.

Sass

```
1cm * 1em => 1 cm * em
2in * 3in => 6 in * in
(1cm / 1em) * 4em => 4cm
2in + 3cm + 2pc => 3.514in
3in / 2in => 1.5
```

Stylus

```
1cm * 1em => 1 cm * em
2in * 3in => 6in
(1cm / 1em) * 4em => 4cm
2in + 3cm + 2pc => 5.181in
3in / 2in => 1.5in
```

S.l. 4.7. *Primjer matematičkih operacija u SASS-u i Stylus-u*

Petlje

Petlje su korisne kada preprocesor treba proći kroz polje i primjeniti određeni stil na više elemenata u nizu. Korisno je za stvaranje *grid* sustava ili kada treba napisati if/else uvjet. Primjer vidljiv na Slici 4.8.

```sass
@for $i from 1px to 3px {
    .border-#{i} {
        border: $i solid blue;
    }
}
```

S.l. 4.8. *Korištenje FOR petlje u SASS preprocesoru*

Mixins

Mixini su set definicija koje se prevode("kompajliraju") ovisno o parametrima ili statičnim vrijednostima koje sadrže. *Mixini* se mogu zamisliti kao funkcije koje primaju određene parametre te se prilikom pozivanja *mixina* ti parametri obrađuju pomoću definicija koje ta funkcija sadrži.

Iz primjera na Slici 4.9. (SASS) se može vidjeti kako se *mixin* "bordered" deklarira. U sebi sadržava parametrar $width koji označava širinu elementa, a statički je deklariran obrub i mijenjanje boje obruba prelaskom miša preko tog elemenata. Nadalje kada se želi pozvati *mixin* "bordered" za neki element, dovoljno je napisati njegovo ime kao instancu i u zagradi napisati željeni paramter, u ovom slučaju širinu obruba - 5px.

Sass

```
@mixin bordered($width) {
    border: $width solid #ddd;

    &:hover {
        border-color: #999;
    }
}

h1 {
    @include bordered(5px);
}
```

Less

```
.bordered (@width) {
    border: @width solid #ddd;

    &:hover {
        border-color: #999;
    }
}

h1 {
    .bordered(5px);
}
```

Stylus

```
bordered(w)
    border: n solid #ddd
    &:hover
        border-color: #999

h1
    bordered(5px)
```

Css

```
h1 { border: 5px solid #ddd; }
h1:hover { border-color: #999; }
```

S.I. 4.9. *Mixini u SASS/LESS/STYLUS preprocesorima i primjerom prevedenog CSS koda*

5. ZAKLJUČAK

Svrha ovog završnog rada predstavljanje je pojma "preprocesori/pretprocesori" kao i uvrštavanje CSS preprocesora u sam proces izrade sučelja za web aplikacije. U sklopu rada "CSS pretprocesori" predstavljena je teorijska podloga koja stoji iza preprocesora te je kroz praktični i teorijski dio razrađen postupak izrade sučelja web aplikacije gdje su se prikazale osnovne funkcionalnosti SASS preprocesora, a isto tako i navele neke druge naprednije funkcionalnosti.

Korištenje CSS preprocesora pokazalo je mnogo više prednosti u odnosu na pisanje klasičnim CSS kodom. Preprocesorima se izbjegava nepotrebno ponavljanje koda, hijerarhijska struktura elemenata u kodu postaje puno lakša za praćenje, a ono najbitnije jest da su preprocesori cijeli postupak oblikovanja elemenata pomoću CSS jezika sveli na modularnost i jednostavnost pa je tako održavanje koda puno brže jer programeri imaju pristup manjim segmentima koda koji je vezan za određene cjeline. Po potrebi izmjena određenog dijela koda pristupa se samo segmentu (.scss dokumentu u slučaju ovoga rada) u kojemu je on zapisan.

Tijekom pisanja ovoga rada (prva polovica 2017. godine) unutar CSS-a su se službeno pojavile i počele koristiti neke nove funkcionalnosti poput mogućnosti kreiranja i korištenja varijabli, jednostavnih matematičkih operacija pomoću "calc()" funkcije i definiranja *grid* sustava pomocu "CSS Grid Layout" funkcionalnosti što je uvelike unaprijedilo CSS kao glavni jezik za oblikovanje HTML elemenata. Ove nove funkcionalnosti smanjit će potrebu za korištenjem dodatnih ekstenzija prilikom razvoja web sučelja, dok za potrebe velikih projekata preprocesori i dalje vode po broju funkcionalnosti, lakoći održavanja i modularnosti koda.

LITERATURA

[1] Sass [online], SASS- Lang, dostupno na: http://sass-lang.com [13.9.2017.]

[2] Sass vs Less [online], keyCDN, dostupno na: https://www.keycdn.com/blog/sass-vs-less [13.9.2017.]

[3] Prabhu, Anirudh, Beginning CSS Preprocessors, 2015.

[4] Gallery image compression framework frontend [online], Github, dostupno na: https://github.com/Levara/ImageClinic/tree/frontend [18.9.2017.]

[5] ImageClinic [online], Github, dostupno na: https://github.com/Levara/ImageClinic [17.9.2017.]

[6] CSS preprocessor [online], MDN web docs, dostupno na: https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor [13.9.2017.]

[7] Stylus [online], Stylus-Lang, dostupno na: http://stylus-lang.com [13.9.2017.]

[8] CSS [online], MDN web docs, dostupno na: https://developer.mozilla.org/en-US/docs/Web/CSS [18.9.2017.]

[9] Cascading Style Sheets PhD Thesis [online], Hakon Wium Lie, dostupno na: https://www.wiumlie.no/2006/phd [17.9.2017.]

[10] Sass::Script::Functions [online], dostupno na: http://sass-lang.com/documentation/Sass/Script/Functions.html

SAŽETAK

U modernom frontend development svijetu dolazi do potrebe za modularnim i efikasnim CSS-om radi lakšeg održavanja i bržeg razvoja web aplikacija i kompleksnih sučelja. Zbog prevelikog broja jQuery *pluginova* i CSS *polyfillova* došlo je vrijeme da se zajednica pozabavi i smisli način kako da CSS učiti učinkovitijim. Na tržištu se prije par godina pojavljuju CSS preprocesori koji omogućavaju brži, jednostavniji i modularniji pristup pisanju CSS-a. Sintaksta preprocesora je ista kao i CSS, ali uz dodatne funkcionalnosti popust pisanja for petlji, funkcija, varijabli, interpolacije i sl. postaju broj 1 kada je u pitanju frontend development. Preprocesori su najbolju slavu odnjeli zbog mogućnosti jednostavnog i modularnog pisanje koda, onosno cijelo sučelje neke aplikacije može se zapisati u neograničen broj preprocesorskih datoteka koje se *importaju* u glavnu datoteku koja se prevodi ("kompajlira") u CSS datoteku. Ta glavna CSS datoteka se povezuje sa HTML dokumentom te se sav sadržaj i svojstva sučelja povlače i prikazuju iz te datoteke.

Na primjeru web aplikacije prikazan je proces dizajniranja i razvoja sučelja kroz određene korake, a također je i prikazano modularno kako CSS preprocesori (SASS) omogućavaju brži razvoj aplikacije te jasnije dokumentiranje projekta koje kasnije znatno olakšava posao održavanja aplikacije.

Ključne riječi: CSS, web aplikacija, preprocesori, SASS

ABSTRACT

Modern frontend developent world needs more modular and efficient way to write CSS in order to prevent complexity of maintaining web applications and more complex layouts. In order to write better CSS, there were different approaches such as separating definitions into smaller files and importing them in to one main file. Preprocessors with their advanced features, helped to achieve writing reusable, maintainable and extensible codes in CSS. By using a pre-processor, you can easily increase your productivity, and decrease the amount of code you are writing in a project.

We showed the process of designing and developing web application layout through several steps and also we concluded how preprocessors can make developers's life easier and project overall more maintainable.

Keywords: CSS, web application, preprocessors, SASS

ŽIVOTOPIS

Dominik Kotris rođen je 9. lipnja 1994. godine u Osijeku. Živi u Osijeku gdje je i odrastao. Završio je osnovnu školu Grigora Viteza u Osijeku nakon čega upisuje III. Prirodoslovno-matematičku gimnaziju Osijek. Za vrijeme srednje škole počinje se aktivno baviti web developmentom, točnije dizajnom i razvojem sučelja web aplikacija. Pri završetku srednje škole odnosi 3. finalno mjesto na hackathonu u organizaciji OSC-a gdje se detaljno upoznaje sa pojmom "startup" te dobiva iskustvo u pisanju poslovnih planova. Izravnim upisom upisuje Elektrotehnički Fakultet u Osijeku smjer Računarstvo. Tijekom studija odnosi razne pobjede na natjecanjima najboljih poslovnih planova i hackathona, a najveći uspjeh bila mu je pobjeda sveučilišnog natjecanja "Idejom do Amerike" gdje je sa svojom poduzetničkom idejom KOORD i timom došao do svjetskoga natjecanja studenatskih startupa u gradu Fort Worth, Texas. Od početka studiranja profesionalno se bavi freelance Web i UX dizajnom te aktivno prati radionice i predavanja iz IT industrije gdje se stalno obrazuje i usavršava znanje iz područja web developmenta i dizajna.