Karakteristike SASS sintakse i primjena u praksi

Buljan, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FILOZOFSKI FAKULTET

ODSJEK ZA INFORMACIJSKE ZNANOSTI

MENTOR: doc.dr.sc. BORIS BADURINA

KOMENTOR: asistent TOMISLAV JAKOPEC

STUDENT: LUKA BULJAN

KARAKTERISTIKE SASS SINTAKSE I PRIMJENA U PRAKSI

Završni rad

OSIJEK, 2014.

Sadržaj

SAŽETAK

U ovom radu pobliže se prikazuju okolnosti nastanka te razvoj stilskog jezika CSS kroz godine. U uvodu rada prikazuju se uvjeti u kojima je nastao navedeni stilski jezik, a kroz sljedeća poglavlja govori se o postupnom razvoju *CSS*-a kao standardnog jezika za stilsko oblikovanje *HTML* datoteka, ali i njegovim karakteristikama i dodanim funkcijama sa svakom novom inačicom istog. Nakon detaljnog i velikog uvodnog dijela, u radu se govori o *SASS* preprocesoru (SASS: Syntactically Awesome Style Sheets), točnije, o karakteristikama njegovih dvaju postojećih sintaksi i njegovoj svrsi. Tomu slijedi odjeljak o instalaciji i praktičnoj primjeni istog na postojećem projektu, nakon čega iznosimo zaključne napomene o *SASS* preprocesoru.

KLJUČNE RIJEČI

Sass, SCSS, CSS, HTML, stilski jezik, označiteljski jezik, Koala

1. UVOD

Početkom 1990-ih i zahvaljujući *HTML-u, World Wide Web* se počeo koristiti kao platforma za elektroničko izdavaštvo. Na svijet dolaze brojni preglednici poput Tim Berners Lee-ovog *NeXT-a, Viole, Harmony-a*, no nedostajala je jedna ključna komponenta u tom procesu. Naime, *HTML* se koristio isključivo za sadržaj i strukturu dokumenta, nikada nije bio zamišljen kao jezik za stilsko oblikovanje – taj posao na kratko je vrijeme bio prepušten preglednicima, odnosno korisnicima[1]. U teoretskom smislu, takva je funkcionalnost preglednika – omogućavanje izmjene stilskih karakteristika na zahtjev korisnika – dobrodošla u očima korisnika, no predstavljala je veliki problem u očima autora, koji nisu htjeli da drugi izmjenjuju izgled njihovog sadržaja. Nije bilo moguće postići kompromis između tih dvaju tehničkih nesukladnosti; drugim riječima, nije bilo moguće istovremeno ne izmijeniti strukturu dokumenta i ponuditi mogućnost izmijene istog. Iz tog razloga, mogućnost izmjene stilskih karakteristika dokumenta u preglednicima polako je, ali sigurno počela nestajati te se 1993. godine, točnije izlaskom *NCSA Mosaic* preglednika svela samo na promjenu boje i fonta teksta. Ako se uzme u obzir da alternative za stilsko uređenje dokumenta nije bilo, Web izdavaštvo suočilo se s velikim problemom kojeg su u najvećem obujmu osjetili sami autori. Štoviše, jedno od najvećih pitanja bilo je upravo pitanje kako promijeniti boju i font teksta. To je, naime, poprilično jednostavno postići u programima poput *Microsoft Office Word-a* i njemu sličnih, ali *HTML* tu mogućnost nije podržavao. Uvidjevši taj problem, Hakon W. Lie predstavio je na konferenciji *Mosaic and the Web* u Chicagu 1994. Godine prvi predložak *Cascading HTML style sheets* (dalje *CHSS*)[2], koji je u konačnici zaslužan za nastanak stilskog jezika *Cascading Style Sheets* (dalje *CSS*), kojeg danas poznajemo.

[1]Wium Lie, Hakon; Bos, Bert. Cascading Style Sheets: designing for the Web, 1999. URL: http://www.w3.org/Style/LieBos2e/history/Overview.html(1.8.2014.)
[2]Isto

2. CASCADING HTML STYLE SHEETS

Predložak u pitanju po prvi je puta predstavio sveobuhvatni stilski jezik koji je primjenjiv na *HTML* dokument bez obzira o pregledniku koji se pritom koristi. Štoviše, *CHSS* je i dalje omogućavao korisnicima primjenu stilskih promjena putem preglednika. Iako su preglednici i dalje omogućavali primjenu stilskih promjena na dokument, H. W. Lie smatrao je da takva primjena stilskih rješenja nije adekvatna iz tri konkretna razloga:

Stilski jezici bili su statični i rijetko su se mijenjali razvojem preglednika. Također, stilski jezici rađeni su isključivo za određene platforme, što je sprječavalo razvoj i primjenu sveobuhvatnog i jedinstvenog stilskog jezika te autori nisu imali nikakvu kontrolu nad prezentacijom vlastitog sadržaja; iako korisnici uvijek imaju zadnju riječ, autor treba imati pravo sadržaj oblikovati prema vlastitoj želji.

Jedna od većih prednosti globalno primjenjivog stilskog jezika postizanje je blagog kompromisa između autora i korisnika oko stilskog uređenja sadržaja, što je spomenuto u prethodnom poglavlju kao jedan od većih problema *Web-a* u to vrijeme, kao i pružanje autorima i izdavačima jednog jedinstvenog jezika za stilsko oblikovanje, čime ih se lišava potrebe za korištenjem različitih stilskih jezika brojnih preglednika[3]. Štoviše, u današnje vrijeme, kada *Web* ima puno veći obuhvat nego 1990-ih i kada su na snazi društvene mreže koje su iznimno personalizirane stranice, taj kompromis oko stilskog uređenja i dalje je od iznimne važnosti kao i na samom početku *Web-a*. Kompleksnost tih društvenih mreža u tehničkom je pogledu značajno veća od bilo koje stranice 1990-ih, a stilsko uređenje ima iznimno veliki utjecaj – dizajn i funkcionalnost popularnih stranica često se mijenjaju, a korisnicima se omogućava primjena različitih stilova i tema. Korisnici čak predlažu promjene u dizajnu ili funkciji, čime se autor i korisnik više ne nalaze na dvije suprotne strane nego rade kolektivno na poboljšavanju iskustva za sve.

Već prilikom kratkog pregleda predloška *CHSS-a* i kratkog uvida u primjere koje je predstavio H. W. Lie, lako je uočiti utjecaj *CHSS-a* na *CSS* kakvog danas znamo. Iako je *CHSS* na prvi pogled nesofisticiran, sva osnovna funkcionalnost koju danas pronalazimo u *CSS-u* ipak je prisutna, s ponekim razlikama u sintaksi. *CHSS* je grubi predložak i to se najprije vidi u samoj strukturi dokumenta. U *CHSS-u* *<h1>*oznaku bi poravnali po sredini te joj promijenili font naredbama *h1.align.style = center* i *h1.font.family = helvetica*, dok bi u *CSS-u* danas ta naredba

[3]Wium Lie, Hakon. Cascading HTML style sheets: a proposal, 10.10.1994. URL: http://www.w3.org/People/howcome/p/cascade.html (1.8.2014.)

jednostavno bila *h1 { text-align: center; font-family: 'Helvetica'; }*. Još jedna od sličnosti *CHSS-a* i *CSS-a* jest način na koji se stilska datoteka poziva unutar *HTML* datoteke – oznakom *<link>* u zaglavlju. Naravno, postoji mala razlika u sintaksi – u *CHSS-u* ta je naredba glasila *<LINK REL="style" HREF="../style">*, dok u *CSS-u* glasi *<link rel="stylesheet" type="text/CSS" href="style.css">*[4]. Uzevši prijašnje primjere u obzir, nije teško primijetiti utjecaj *CHSS-a* na stilski jezik koji koristimo danas i shvatiti njegovu važnost u stvaranju istog – *CSS-a*.

3. CSS

Daleko najbitniji stilski jezik u kontekstu ovog rada je, naravno, *CSS*. Od dana kada je predstavljen sve do danas, *CSS* predstavlja globalni standard za stilsko uređivanje *HTML* dokumenata. Od početka je zamišljen kao jednostavan, univerzalan stilski jezik, primjenjiv na sve preglednike i *HTML* dokumente s jednim jedinstvenim ciljem – istovremeno ugoditi stilskim željama i autora i korisnika[5]. *CSS* se godinama usavršavao, nedostaci su dorađeni, a temeljna karakteristika – jednostavnost – još uvijek je ključna karakteristika stilskog jezika *CSS*. Stoga su sintaksa i semantika *CSS-a* poprilično jednostavne, a osnovna obilježja i karakteristike sintakse i semantike *CSS-a* opisane su u sljedećem poglavlju.

3.1 CSS SEMANTIKA I SINTAKSA

Svaki programski, označiteljski ili stilski jezik ima svoju sintaksu, stoga je ovo poglavlje posvećeno objašnjavanju i predstavljanju semantike i sintakse *CSS-a*, što je nužan preduvjet za praćenje sljedećih poglavlja rada. U prethodnom poglavlju spomenuto je kako je *CSS* oduvijek bio zamišljen kao jednostavan stilski jezik i to se nizom godina nije promijenilo. Doduše, *CSS* podrazumijeva osnovno predznanje o *HTML-u* i njemu pripadnim elementima i atributima, no u ovom slučaju to ne predstavlja problem.

Postavimo prvo primjer *CSS* oznake – *h1 { color: blue }*.

Svaka *CSS* oznaka sastoji se od dva osnovna dijela – selektora i deklaracije. U navedenom primjeru, selektor je *h1*, a deklaracija je *color: blue*.

Selektor ima ulogu povezivanja *HTML* dokumenta i *CSS* stilskog dokumenta te svaki *HTML* element predstavlja mogući *CSS* selektor. Različite su vrste selektora. Selektor elementa već smo upoznali u gore navedenom primjeru – selektor je u tom slučaju *HTML* element.

[4]Wium Lie, Hakon. Cascading HTML style sheets: a proposal, 10.10.1994. URL: http://www.w3.org/People/howcome/p/cascade.html (1.8.2014.)
[5]Wium Lie, Hakon; Bos, Bert. Cascading Style Sheets: designing for the Web, 1999. URL: http://www.w3.org/Style/LieBos2e/history/Overview.html(1.8.2014.)

Selektor klase može se primijeniti na bilo koji *HTML* element kojemu je atribut na primjer *class="klasa"*, a *CSS* selektor u tom slučaju glasi .klasa. Selektor *ID* ima sličnu svrhu kao selektor klase, no atribut *ID* u *HTML-u* označava jedinstvenu varijantu nekog elementa – primjerice jedinstveni *h1* s atributom *id="ident"* koji se ne ponavlja u dokumentu. *CSS* selektor u tom slučaju glasi *#ident*. Kontekstualni selektor koristi se kada pod-elementu unutar nekog *HTML* elementa želimo primijeniti drugačiju vrijednost[6]. U *HTML-u, n*a primjer, unutar *<h1>* elementa kojemu smo dali plavu boju teksta putem *CSS-a* postoji *<a>* element. Ako želimo promijeniti boju teksta samo tom jednom *<a>* elementu unutar *<h1>* elementa *CSS* selektor glasi *h1 a { color: red; }* .

Nadalje, deklaracija se dijeli na dva dijela – svojstvo i vrijednost. U prije navedenom primjeru *color* predstavlja svojstvo, a *blue* predstavlja vrijednost svojstva. Svojstvo i vrijednost uvijek se nalaze unutar vitičastih zagrada, no selektori i deklaracije mogu se grupirati u svrhu smanjivanja završne veličine stilskog dokumenta.

Ako deklaraciju *color: blue* želimo primijeniti na više *HTML* elemenata, selektore grupiramo na način da svaki selektor prije deklaracije odvojimo zarezom. Na kraju oznaka glasi *h1, h2, h3 { color: blue }*. Ako na određeni selektor želimo primijeniti više deklaracija, svaka se deklaracija odvaja točka-zarezom. U primjeru ta oznaka glasi *h1 { color: blue; font-weight: bold; font-family: 'Segoe UI'; }*. Određena svojstva deklaracija imaju vlastitu grupacijsku sintaksu, primjerice svojstvo *font*. Stoga, prošla oznaka može se napisati na jednostavniji način, a postiže se jednaki učinak – *h1 { font: bold 12pt Segoe UI }*.[7]

Pseudo-klase su određene varijante nekih *HTML* elemenata, ponajprije elementa *<a>* - poveznice. Postojeće pseudo-klase za navedeni element glase *:link*, *:visited* i *:active* koje označavaju posebna stanja *<a>* elementa – je li poveznica posjećena ili ne te nalazimo li se trenutno na toj poveznici. Putem pseudo-klasa u *CSS-u* može se odrediti određena deklaracija za svaku podvrstu elementa *<a>*.[8] Još jedan primjer pseudo-klasa jest *:hover*, pseudo klasa koja se može dodati na bilo koji element, pri čemu se deklaracije navedene za taj selektor primjenjuju na dotični *HTML* element samo kada se pokazivač miša nalazi preko tog elementa. Jednostavnije rečeno, ako uzmemo u obzir prije postavljeni primjer *CSS* oznake u kojem smo naveli da je tekst *<h1>* elementa plave boje i nakon njega u *CSS* datoteku ubacimo novu oznaku koja glasi *h1:hover { color: green; }*, tekst unutar elementa *<h1>* bit će plave boje sve dok

[6]Cascading Style Sheets: level 1, 17.12.1996. URL: http://www.w3.org/TR/REC-CSS1/ (4.8.2014.)
[7]Isto
[8]Isto

pokazivačem miša ne pređemo preko njega – u tom trenutku boja teksta elementa mijenja se u zelenu.

Jedno od pravila koja su *CSS* učinile standardnim stilskim jezikom u Web okruženju je, naravno, činjenica da su stilske datoteke kaskadne – autor postavlja osnovnu stilsku datoteku, a korisnik nakon nje može primijeniti svoje stilske promjene. Točnije rečeno, *CSS* je kaskadni zbog činjenice da one stilske promjene koje preglednik zadnje učita pregaze, odnosno zamjenjuju one koje se nalaze prije njih, što omogućava primjenu različitih stilskih datoteka na jednom dokumentu[9]. Naravno, *CSS* nije toliko jednostavan i svoj uspjeh ne može bazirati samo na jednom pravilu, stoga *CSS* ima nekoliko izuzetno korisnih pravila vezanih uz kaskadnost.

Primjerice, stilske karakteristike mogu se naslijediti od elementa roditelja[10]. Recimo da u *HTML* datoteci imamo element *<div style=font-family: 'Segoe UI'; border:1px solid red>*, a unutar njega element * Tekst *. Neka stilska svojstva elementa automatski se prenose na sve elemente sadržane unutar istog – u ovom primjeru to je *font-family*. Element * *je stoga naslijedio font teksta, no nije naslijedio i svojstvo *border*. Kako bi svojstvo *border* primijenili na element * *, potrebno je unutar početne oznake elementa ubaciti stilsko svojstvo *border* i vrijednost *inherit*. Završni kod izgledao bi ovako: *<div style=font-family: 'Segoe UI'; border:1px solid red> <p style=border: inherit;> Tekst *.

Još jedno od pravila *CSS* stilskog jezika jest da se stilske vrijednosti na *HTML* element mogu primijeniti na tri načina, ne uključujući nasljeđivanje vrijednosti od elementa roditelja. Daleko najpraktičniji način je pravljenje zasebne, vanjske *.css* datoteke[11] koja sadrži sve *CSS* oznake te njeno povezivanje s *HTML* datotekom putem *<link>*elementa u zaglavlju *HTML* dokumenta na koji se stilske promjene primjenjuju. Drugi način je pisanje *CSS* oznaka unutar zaglavlja *HTML* dokumenta, točnije unutar oznake *<style type=text/CSS>…</style>*[12], što stvara dodatni nered unutar dokumenta ako se radi o većem broju *CSS* oznaka. Treći način je pisanje stilskih deklaracija unutar *style* atributa *HTML* elementa na koji se stilska promjena želi primijeniti[13], što je praktično u slučaju da je taj element jedinstven i nema ga potrebe posebno označavati unutar *CSS* datoteke.*css* semantika i sintaksa ostale su nepromijenjene u svim

[9]Hunt, Ben. Inheritance and Cascading Styles in CSS Explained. URL: http://webdesignfromscratch.com/html-css/css-inheritance-cascade/#1 (4.8.2014.)
[10]Cascading Style Sheets: level 1, 17.12.1996. URL: http://www.w3.org/TR/REC-CSS1/ (4.8.2014.)
[11]Hunt, Ben. Inheritance and Cascading Styles in CSS Explained. URL: http://webdesignfromscratch.com/html-css/css-inheritance-cascade/#1 (4.8.2014.)
[12] Isto
[13]Isto

verzijama, a novi elementi redovito su dodavani sa svakom verzijom *CSS* stilskog jezika – što je tema sljedećih dvaju poglavlja.

3.2 CSS2

Druga verzija *CSS* stilskog jezika objavljena je 1998. godine i službeno slovi kao standard za stilsko oblikovanje čak 16 godina nakon njezinog objavljivanja, budući da je *CSS3* još uvijek u izradi. Sa sobom je donijela dodatne funkcije u pogledu novih načina označavanja selektora *CSS* oznake, uspoređivanja *CSS* oznaka i *HTML* atributa, pozicioniranja/razmještaja elemenata i stvaranja slojeva, mijenjanja sadržaja unutar *HTML* elemenata te dodatne funkcije za pozadinske slike, boju, uređenje teksta i promjenu pokazivača miša.[14]

Svima koji su se susreli s računalom poznat je način funkcioniranja pokazivača miša u operativnom sustavu – postoje određene varijante različitih izgleda koje se prikazuju ovisno o kontekstu – ako operativni sustav vrši određeni zadatak, pokazivač se, primjerice, u Windows 8.1 operativnom sustavu pretvara u animiranu plavu kružnicu, ako pokazivač miša stoji nad hiperlinkom pretvara se u ruku s ispruženim kažiprstom, ako povećavamo ili smanjujemo prozor pokazivač se pretvara u dijagonalnu liniju sa strjelicama na oba kraja i tako dalje. *CSS2* sadrži mogućnost određivanja koja će se točno vrsta pokazivača prikazati u dokumentu i pod kojim okolnostima – moguće je odrediti glavni pokazivač u dokumentu i drugu vrstu pokazivača ako isti stoji nad slikom. Svojstvo pomoću kojeg se ta promjena vrši glasi, jednostavno, *cursor*, a neke od vrijednosti su *pointer*, *crosshair*, *wait,* a postoji i mogućnost primjene vlastitog pokazivača vrijednošću *url(putanja/cursor.gif)*. Kod uređivanja teksta dodano je novo svojstvo – *text-shadow*. Radi se o sjeni koju bacaju slova teksta, a ovim svojstvom mogu se odrediti vrijednosti poput veličine i boje. Pozadinske slike mogu se ponavljati u horizontalnom, vertikalnom ili oba pravca te se slika može fiksirati, čime se ona neće pomicati prilikom listanja stranice. Istim redoslijedom svojstva glase *background-repeat: repeat-x/repeat-y/repeat* te *background-attachment: fixed*.[15]

Također, *CSS2* posjeduje mogućnosti promjene sadržaja unutar *HTML* elementa, odnosno putem pseudo-klasa :before i :after može se odrediti tekst koji će se pojaviti na početku i na kraju navedenog *HTML* elementa *.css* oznake *p:before {content: 'Početak'}* i *p:after*

[14]Lee, Xah. CSS1 and CSS2 Differences, 2005. URL: http://xahlee.info/js/css2.html (5.8.2014.)
[15]Isto

{content: 'Kraj'} će u sadržaj svakog elementa *<p>* u *HTML* datoteci ubaciti prije navedeni tekst – '*Početak*' na početku, '*Kraj*' na kraju.[16]

Jedna od važnijih novih sposobnosti stilskog jezika *CSS2* je mogućnost određivanja pozicije *HTML* elemenata i stvaranje slojeva. Pozicioniranje *HTML* elemenata vrši se pomoću svojstva *position* i dvaju atributa – *top/bottom* i *left/right*, koji služe za određivanje udaljenosti elementa od odabranog ruba prozora. Svojstvo *position* ima dvije vrijednosti – *absolute* i *relative*.[17]Vrijednost *absolute* pozicionira element u pitanju od ruba prozora, dok vrijednost *relative* pozicionira element u odnosu na njegovo određeno mjesto u *HTML* datoteci te je uz nju obavezno navesti i atribut smjera u kojem se element namjerava pomaknuti. Vrijednosti atributa *top/bottom/left/right* izražavaju se u brojčanoj vrijednosti piksela – npr. *left: 200px*. Pomoću stilskog jezika *CSS2* jednostavno je i odrediti slojeve i objasniti kako se to čini. U pitanju je jedno svojstvo deklaracije pomoću kojeg se određeni element stavlja ispod ili iznad nekog drugog elementa. Riječ je o svojstvu *z-index* čija se vrijednost iskazuje brojevima, a stvaranje slojeva je jednostavno – što je veći broj, to je sloj više pozicioniran ispred. Ako je *z-index* za sliku u *HTML* dokumentu 1, a *z-index* za tekst kojeg smo pozicionirali na isto mjesto 2, tekst će se pojaviti ispred slike, odnosno slika će biti u pozadini teksta.[18]

Novi načini označavanja *CSS* selektora su *, kojom se označava svaki *HTML* element, znak > označava odnos nad-element > pod-element – primjerice *body > p.tekst > a { color: red }* označava isključivo element *<a>* koji se nalazi unutar elementa *<p class="tekst">* unutar elementa *<body>*, razmak između elemenata u selektoru označava element koji obavezno ispred sebe ima prethodni element – primjerice *table img {border: none}* znači da svaka slika unutar tablice nema obrub, a posljednja nova oznaka selektora je + koji označava neko ograničenje unutar *HTML* dokumenta – primjerice *img + p {color: red}* označava svaki element *<p>* koji slijedi element *<img>*. Uspoređivanje *CSS* oznake s *HTML* atributom vrši se na 3 moguća načina. Prvi način je provjeravanje postoji li *HTML* atribut, a selektor u tom slučaju glasi *a[title] {color: red}*. *CSS* pritom provjerava ima li dotični element, u ovom slučaju element *<a>* atribut *title* te, ako atribut postoji, tekst elementa *<a>* prikazuje se crvenom bojom.[19] Svaka od navedenih funkcija donijela je nešto novo, što je čak 16 godina bilo dovoljno

[16]Isto
[17] Lee, Xah. CSS1 and CSS2 Differences, 2005. URL: http://xahlee.info/js/css2.html (5.8.2014.)
[18] Isto
[19] Isto

za stilsko uređivanje *HTML* dokumenata. No, naravno, uvijek se može napraviti više, stoga je trenutno u izradi treća inačica *CSS-a*, o kojoj više u sljedećem poglavlju.

3.3 CSS3

CSS3 se razlikuje od *CSS2* čak i prije nego se zaviri u oznake i nove funkcije. Naime, *CSS2* je World Wide Web Konzorciju predan kao jedan, dovršeni dokument koji je u sebi sadržavao sve informacije vezane uz navedenu inačicu. *CSS3* se ne predaje na taj način, nego je cijeli dokument podijeljen u module, koji svaki zasebno prolazi provjeru i implementaciju u World Wide Web Konzorciju.[20] Ovaj pristup omogućava raniju implementaciju *CSS3* stilskog jezika u preglednike, čime se sam proces prilagodbe i prelaska na novu verziju znatno olakšava i ubrzava, dok je sam stilski jezik još uvijek u izradi.

U *CSS3* dodani su novi selektori atributa. Sveukupno su 3 nova načina uspoređivanja selektora i *HTML* atributa – jedan način je provjera odgovara li atribut točno onome što je u *CSS-u* navedeno, a vrši se pomoću oznake *p[foo^="bar"]*, čime se *CSS* oznaka primjenjuje na onaj <*p*> element koji sadrži atribut *foo="bar"*. Drugi način je provjera odgovara li početak atributa onome što je u *CSS-u* navedeno, a vrši se pomoću oznake *p[foo$="bar"]*, čime se *CSS* oznaka primjenjuje na onaj <*p*> element čiji atribut *foo* počinje s *bar* – primjerice, <*p foo="barka"*>. Posljednji novi način uspoređivanja *CSS* selektora i *HTML* atributa jest provjera odgovara li kraj vrijednosti atributa onome što je u *CSS-u* navedeno, a vrši se pomoću oznake *p[foo*="bar"]*, čime se *CSS* oznaka primjenjuje na onaj <*p*> element čiji atribut *foo* završava s *bar* – primjerice, <*p foo="dabar"*>.[21]

Dodano je 16 novih pseudo-klasa, neke od kojih su *:root* koja označava korijenski element dokumenta, što je uvijek <*HTML*> kod *HTML* dokumenta, *:last-child* koja označava posljednji pod-element odabranog elementa, *:only-of-type* koja označava element koji je jedini te vrste u dokumentu, *:checked* koji označava radijalnu tipku ili *checkbox* kada je uključena i brojne druge pseudo-klase. Među novim dodacima također se nalazi novi kombinator *CSS* selektora - *p ~ img*, čime se u ovom primjeru označava element <*img*>koji slijedi element <*p*>, no ne mora ga nužno odmah slijediti[22].

[20]Kyrnin, Jennifer. What is the difference between CSS2 and CSS3: The major changes to CSS3. URL: http://webdesign.about.com/od/css3/a/differences-css2-css3.htm (5.8.2014.)
[21] Isto
[22] Isto

U prethodnom poglavlju, koje se ticalo *CSS2* i novih oznaka koje je sa sobom donio, spomenute su nove funkcije koje se tiču pozadinskih slika. *CSS3* nastavlja tradiciju koju je *CSS2* započeo te donosi mogućnost postavljanja više pozadinskih slika pomoću slojeva i pozicioniranja. Uz to, dodane su i nova svojstva povezana s pozadinskim slikama, a to su *background-clip, background-origin* i *background-size*. Svojstvo *background-clip* omogućava definiranje na koju će se marginu pozadinska slika primijeniti – *border box, padding box* ili *content box*. Neka od postojećih svojstava za pozadinsku sliku dobila su nove vrijednosti. Svojstvo *background-repeat* dobilo je dvije nove vrijednosti – *space* i *round*[23]. Vrijednost *space*pozadinsku sliku ponavlja isto kao *repeat*, ali razmak između slika bude toliki da nijedan dio slike ne bude odsječen rubom prozora. Vrijednost *round* povećava pozadinsku sliku tako da se postigne isti učinak kao s vrijednošću *space* – nijedan dio pozadinske slike neće biti odsječen, ali u ovom slučaju neće ni biti razmaka između ponovljenih slika[24].

CSS3 sa sobom je donio i nova svojstva za obrub elemenata. Svojstvo *border-radius, border-top-right-radius, border-top-left-radius, border-bottom-right-radius, border-bottom-left-radius* omogućavaju zaobljavanje uglova obruba, a vrijednosti svojstva iskazuju se u pikselima – što je veći broj piksela, više se rub zaobljuje. Svojstvo *border-image-source* određuje izvornu putanju slike koju želimo koristiti umjesto obruba koji su već definirani, *border-image-slice* predstavlja unutrašnji odmak slike obruba od ruba iste, *border-image-width* određuje širinu slike obruba, *border-image-outset* određuje koliko će slika obruba izlaziti izvan određenog područja elementa, *border-image-stretch* određuje kako će stranice i sredina slike obruba biti posložene ili prilagođene u veličini. Svojstvo *border-image* skraćeni je oblik svojstva u kojem se mogu upisati vrijednosti za sve prije navedene *border-image* elemente. Također, jedan manji dodatak je dodavanje pozadinske sjene za *box* elemente putem svojstva *box-shadow*.[25]

Jedna od značajnijih funkcija koje *CSS3* pruža jest podjela <*body*> elementa *HTML* dokumenta u stupce jednakih veličina. Svojstva *column-width* i *column-count* određuju širinu stupaca u koje će se sadržaj *HTML-a* smjestiti, odnosno određuju broj stupaca na stranici, koje će preglednik potom stvoriti onolike širine koja je dovoljna da se širina preglednika popuni

[23]Kyrnin, Jennifer. What is the difference between CSS2 and CSS3: The major changes to CSS3. URL: http://webdesign.about.com/od/css3/a/differences-css2-css3.htm (5.8.2014.)
[24]Lazaris, Louis. CSS3's 'space' and 'round' values for background-repeat, 15.8.2011. URL: http://www.impressivewebs.com/space-round-css3-background/ (5.8.2014.)
[25]Kyrnin, Jennifer. What is the difference between CSS2 and CSS3: The major changes to CSS3. URL: http://webdesign.about.com/od/css3/a/differences-css2-css3.htm (5.8.2014.)

brojem stupaca koji je *CSS-om* naveden. Skraćeni oblik svojstva glasi jednostavno *columns,* u kojem se mogu upisati vrijednosti širine ili broja – ili čak oboje, iako to nije funkcionalno. Dodatna svojstva za određivanje karakteristika stupaca su *column-gap*, koje određuje širinu razmaka između svakog stupca, *column-rule-color*, koje određuje boju linije koja dijeli stupce, *column-rule-style* koje određuje stil prikazivanja linije – puna crta, točkasta, dupla itd., *column-rule-width* određuje širinu linije koja dijeli stupce, a skraćeni oblik svojstva glasi *column-rule,* kojim se u jednom redu mogu odrediti vrijednosti za svako svojstvo vezano uz liniju koja dijeli stupce.[26] Sva navedena svojstva već su dostupna i podržana od strane preglednika, a brojna svojstva još su u izradi, čime *CSS3* postaje najopširnija inačica *CSS-a* do sada.

Kroz prethodnih par poglavlja objašnjene su karakteristike semantike i sintakse dostupnih inačica *CSS-a* te je objašnjena razlika između postojećih inačica navedenog stilskog jezika. Posebno je objašnjena sintaksa *CSS* oznake, koja je podijeljena na dva dijela - selektor i deklaraciju te je, u usporedbi s *CHSS-om* puno jednostavnija i izravnija. No, kao što se kroz naslov rada može natuknuti, postoje načini da se *CSS* dodatno pojednostavi, a količina potrebnog upisivanja oznaka izrazito smanji. Sljedeće poglavlje govori upravo o tome.

4. CSS PREPROCESORI

Već je spomenuto kako je *CSS* od samog početka zamišljen kao jednostavan, no, uvidom u neke karakteristike *CSS3* stilskog jezika može se vidjeti kako njegove sposobnosti postaju sve brojnije, što direktno utječe i na samu jednostavnost. Naravno, sve promjene koje *CSS* prolazi, a i koje će prolaziti, diktirane su potražnjom korisnika. Web dizajn kao struka izuzetno je dobio na popularnosti, a mnoštvo dostupnih web stranica zahtijevaju distinktivni identitet svake – nešto što nije moguće napraviti s izrazito jednostavnim stilskim jezikom. *CSS3* nudi mnoštvo novih mogućnosti, no polako postaje napuhan. K tome, kao stilski jezik nije svemoćan. No, postoje načini za dodatno pojednostavljivanje i proširenje funkcionalnosti *CSS-a – CSS* preprocesori.

Trenutno su najpopularniji *CSS* preprocesori *Stylus, Less* i, prikazan u nastavku, *Sass*. Prednosti koje gotovo svaki *CSS* preprocesor pruža su konciznost, dodatna funkcionalnost i podrška za dodatke. Konciznost svaki preprocesor postiže na način da uklanja nepotrebnu sintaksu te svaki preprocesor pruža mogućnost ugnježđivanja selektora unutar drugih, čime se

[26]Kyrnin, Jennifer. What is the difference between CSS2 and CSS3: The major changes to CSS3. URL: http://webdesign.about.com/od/css3/a/differences-css2-css3.htm (5.8.2014.)

smanjuje nadutost i potreba za ponavljanjem *CSS* oznaka[27]. Sljedeće poglavlje okreće se općenitim detaljima o *Sass-u*.

5. SASS

Kao što je već u prošlom poglavlju spomenuto, *CSS* preprocesori dodaci su *CSS* stilskom jeziku koji dodaju istom na funkcionalnosti i konciznost, a samim time objašnjava se i temeljna funkcionalnost *Sass-a*. Na osnovu toga, *Sass* je ekstenzija *CSS-u* koja u potpunosti podržava *CSS3* te proširuje stilski jezik dodavanjem podrške za varijable, ugnježđivanje selektora i takozvane *mixins-e*. Još jedna od bitnih funkcija *Sass* preprocesora je mogućnost naprednog miješanja boje, odnosno sastavljanja boje od RGB, RGBA i HSL kanala.

Dvije su dostupne sintakse *Sass-a* i međusobno su kompatibilne - *Sassy CSS* (dalje *SCSS*) i starija „uvučena“ sintaksa, jednostavno zvana *Sass*[28].

5.1 SASS SINTAKSA

Starija *Sass*, odnosno „uvučena“ sintaksa služi isključivo kao koncizniji način pisanja *CSS-a*. Umjesto vitičastih zagrada, deklaracija se od selektora odvajaju uvlačenjem reda, a prelazak u novi red razdvaja dva svojstva, što u *CSS-u* predstavlja točka-zarez. Time se smanjuje količina bespotrebnog tipkanja, a ujedno i povećava čitljivost *CSS* dokumenta.

Iako predstavlja puno koncizniji način stiliziranja *HTML* dokumenata, *Sass* sintaksa ima jednu veliku manu – previše se razlikuje od *CSS*-a, što donekle dovodi do potrebe za učenjem novog stilskog jezika. Štoviše, upravo zbog te razlike teško bi bilo pratiti *CSS* kroz njegove promjene, odnosno *Sass* je predstavljao previše posla kako za autore, tako i za ljude koji iza njega stoje. Dakako, *Sass* sintaksa ima i svojih prednosti – prije navedenu konciznost, nema potrebu za vitičastim zagradama i točka-zarezima, čime automatski postiže lakšu čitljivost i iz tog razloga još je uvijek podržana uz noviju *SCSS* sintaksu[29].

[27]Walsh, Nick. CSS Preprocessors: Focused Decisions. URL: http://www.awwwards.com/css-preprocessors-focused-decisions.html (6. 8. 2014.)
[28]Sass: Documentation. URL: http://sass-lang.com/documentation/file.SASS_REFERENCE.html (6.8.2014.)
[29] Long, John W.Sass vs SCSS: Which syntax is better?, 19.2.2011. URL:http://thesassway.com/editorial/sass-vs-scss-which-syntax-is-better (16.8.2014.)

5.2 SCSS SINTAKSA

Upravo zbog razlike između *Sass* sintakse i *CSS* sintakse, uz verziju 3 *Sass* stilskog jezika dodana je i nova *Sass*y *CSS* (*SCSS*) sintaksa. *SCSS* sintaksa nije se razlikovala od *CSS*-a poput *Sass* sintakse – vitičaste zagrade i točka-zarez ponovno su u upotrebi, prelasci u novi red za novi selektor i uvučeni redovi za deklaracije više se ne mogu koristiti, no i dalje pruža iste prednosti *Sass* stilskog jezika kao i starija sintaksa te se ne mora posebno učiti što je sa starijom *Sass* sintaksom praktički bio zasebni stilski jezik[30]. *SCSS* sintaksa ekstenzija je sintaksi *CSS3*, što u prijevodu znači da je svaki valjani *CSS3* dokument ujedno valjani *SCSS* dokument i obratno. *SCSS* razumije i poznate *CSS hackove* poput *!important, @import* i prije spomenuto uspoređivanje *CSS* selektora s atributom *HTML* elementa[31]. Dakako, mana *SCSS*-a je što gubi konciznost *Sass* sintakse i samim time teže je čitati *SCSS* dokumente, no prednosti koje pruža daleko su bitnije i korisnije od konciznosti. Primjerice, *SCSS* je ekspresivniji od uvučene *Sass* sintakse. Recimo da trebamo samo promijeniti boju teksta kada pokazivač miša pređe preko *HTML* elementa. Kako autor navodi, u starijoj *Sass* sintaksi to bi zauzelo dva reda – red za označavanje selektora, prelazak u novi red i uvlačenje teksta za deklaraciju. U *SCSS* sintaksi, jednako kao u samom *CSS*-u, istu naredbu može se napisati u jednom redu[32]. Iako je ova prednost poprilično subjektivna, činjenica stoji da kod kratkih oznaka starija *Sass* sintaksa zauzima previše prostora. Još jedna velika prednost *SCSS* sintakse nad *Sass* sintaksom jest što postojeći alati za *CSS* savršeno funkcioniraju u *SCSS*-u zbog velike sličnosti istih. Treća velika prednost *SCSS*-a jest integracija s postojećim *CSS* datotekama, ukoliko ih se koristi više. Prilikom korištenja starije *Sass* sintakse bilo bi potrebno sve postojeće *CSS* datoteke preobraziti u *Sass* datoteke, što je iziskivalo dodatne sate posla i truda, a *SCSS* zbog prije spomenute sličnosti s *CSS*-om radi iz prve. Od ostalih prednosti mogu se nabrojati lakša prilagodba na *SCSS* nego na *Sass* sintaksu – ponajviše zbog toga što je *Sass* sintaksa poput potpuno drugog stilskog jezika, dok *SCSS* pruža dodatne funkcije koristeći već poznatu *CSS* sintaksu[33]. No, kako *Sass* u cjelini izgleda u praksi?

[30] Isto
[31] Sass: Documentation. URL: http://sass-lang.com/documentation/file.SASS_REFERENCE.html (16.8.2014.)
[32] Long, John W. Sass vs SCSS: Which syntax is better?, 19.2.2011. URL: http://thesassway.com/editorial/sass-vs-scss-which-syntax-is-better (16.8.2014.)
[33] Long, John W. Sass vs SCSS: Which syntax is better?, 19.2.2011. URL: http://thesassway.com/editorial/sass-vs-scss-which-syntax-is-better (16.8.2014.)

6. INSTALACIJA SASS-A

CSS, kao što je prije spomenuto, ključna je komponenta u stiliziranju *HTML* dokumenta, kao takav je zamišljen i nije ga potrebno zasebno instalirati. No, *SASS*, kao dodatak za *CSS*, nije dio samog *CSS*-a nego ga je potrebno posebno instalirati, što se trenutno može učiniti na dva načina –putem aplikacije ili putem *Terminala* ili *Command Prompt-a*.

Prvi način je poprilično izravan – putem različitih *SASS* aplikacija. Postoji veliki broj aplikacija za *Linux, Mac* i *Windows* operativne sustave koje u par minuta na jednostavan način pripreme *SASS*, a i druge *CSS* preprocesore za upotrebu. Neke se od ponuđenih aplikacija naplaćuju, primjerice *Hammer, Compass, CodeKit, Mixture, LiveReload i Prepros*, no postoje i besplatne aplikacije *Scout i Koala*[34], na kojoj se sljedeći primjer bazira.

6.1 INSTALACIJA PUTEM APLIKACIJE

U svrhu ovog primjera, kao što je prije rečeno, koristi se Koala, koja je *GUI* (*Graphical User Interface*) aplikacija za *Mac, Linux* i *Windows* operativne sustave, a pruža jednostavniji način kompiliranja *Sass*-a, *Less*-a, *Compass*-a *i CoffeeScript*-a[35]. Nakon kratke instalacije, otvaranjem Koala aplikacije otvara se mali prozor u koji se dodaje mapa projekta na kojem se radi, nakon čega Koala učitava sve zasebne stilske i *jQuery* datoteke koje se nalaze u mapi navedenog projekta. Od tog trenutka Koala proučava sve izmjene nad datotekama koje se u mapi nalaze, očitava greške u sintaksi istih te prilikom svakog spremanja datoteke Koala kompilira *Sass* ili Less u *.css* datoteku – što znači da nakon instalacije Koale i dodavanja projekta jedini posao koji autoru preostaje jest pisanje, a sve ostalo je automatizirani proces. Na neki način, *Sass* aplikacije mogu se smatrati tihim promatračima u službi autora.

6.2 INSTALACIJA PUTEM TERMINALA

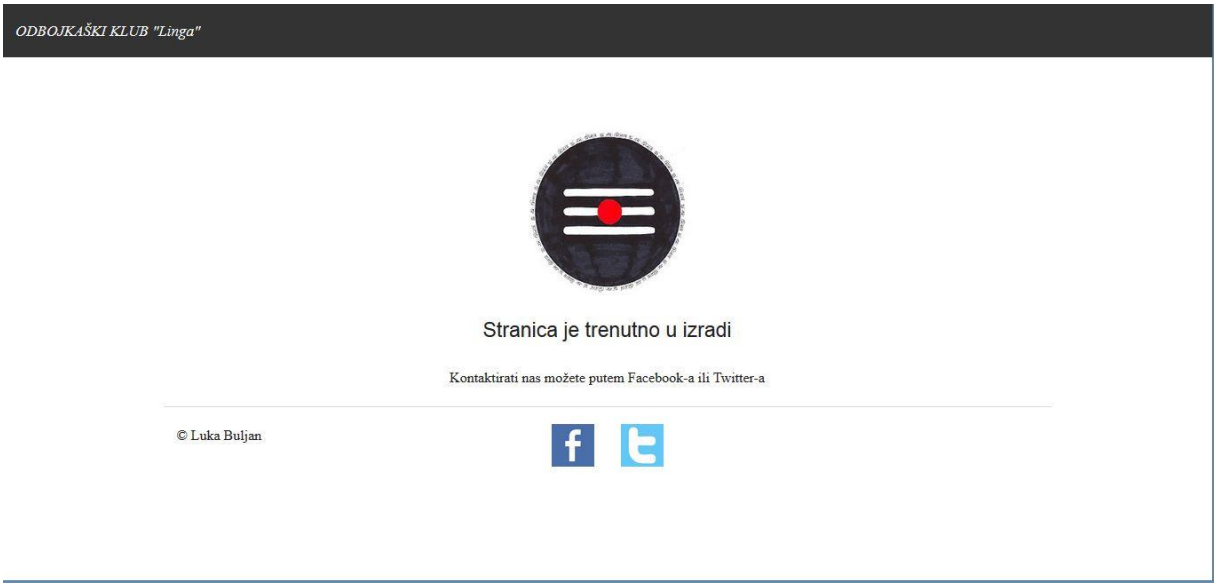
Sass compiler izgrađen je Ruby programskim jezikom, stoga je i sam Ruby potreban na računalu kako bi *Sass* mogao funkcionirati. Mac operativni sustav u sebi već sadrži Ruby, pa je instalacija za jedan korak kraća, dok *Windows* i *Linux* operativni sustavi u sebi ne sadrže Ruby – stoga ih je potrebno instalirati. Korisnici *Windows* operativnog sustava preuzmu Ruby instalaciju s Interneta te izvrše istu, dok korisnici Linux operativnog sustava mogu instalirati

[34] Install SASS. URL: http://sass-lang.com/install (16.8.2014.)
[35] Koala. URL: http://koala-app.com/ (16.8.2014.)

Ruby putem *apt package managera*. Sljedeći korak – *Linux* i *Mac* korisnici pokreću *Terminal*, *Windows* korisnici pokreću *Command Prompt* te se *Sass* instalira naredbom *gem install Sass* ili pak *sudo gem install Sass*[36]. Nakon toga *Sass* funkcionira na isti način kao i *Sass* aplikacija – promatra promjene nad projektnim datotekama te prilikom spremanja svake kompilira *.css* datoteku.

7. PRAKTIČNA PRIMJENA SASS-A

Za primjer praktične primjene *Sass*-a na projektu koristiti će se web stranica prikazana na slici koja slijedi te Koala aplikacija za *Sass* stilski jezik.

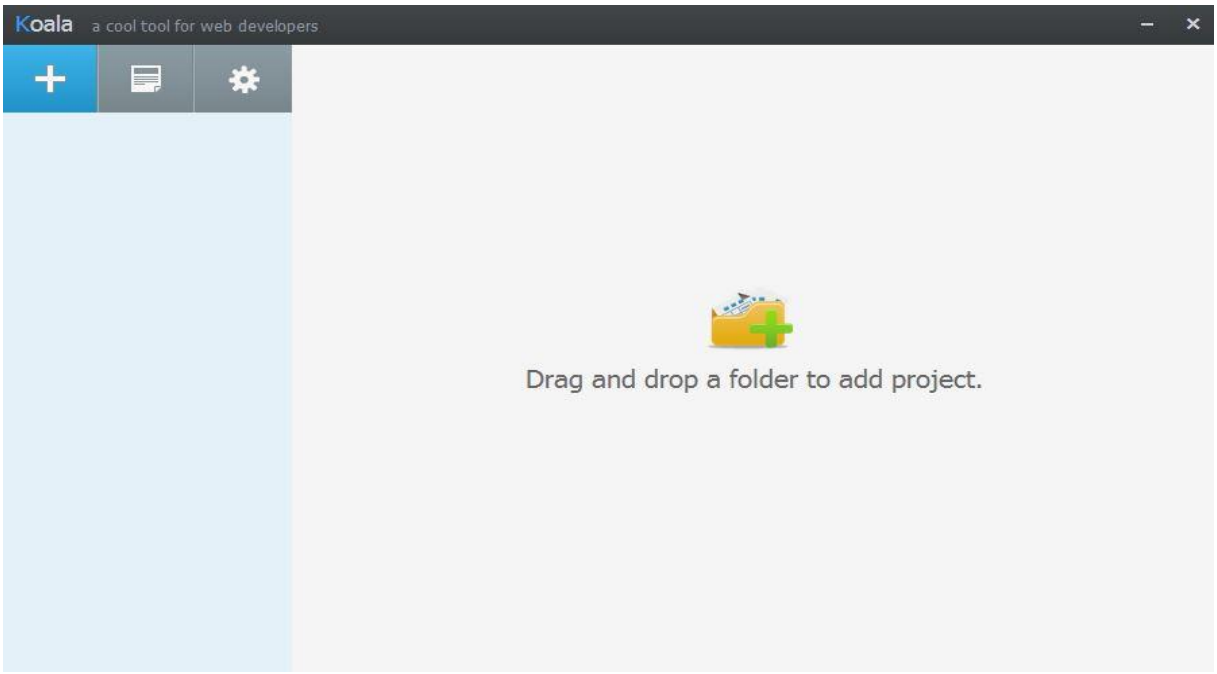


Slika 1 – Primjer web stranice

Stranica ne sadrži puno sadržaja, a samim time i pripadajući *CSS* nije izrazito kompliciran, no poslužiti će u svrhu primjera prednosti *Sass*-a nad *CSS*-om.

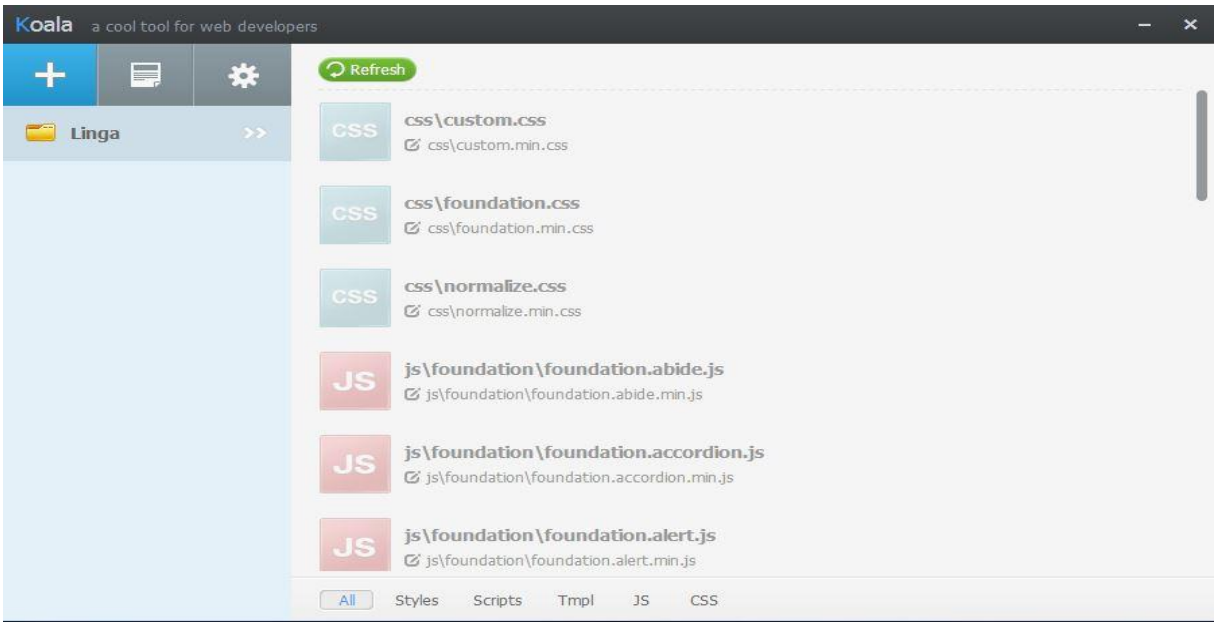
[36] Install SASS. URL: http://sass-lang.com/install (16.8.2014.)

Prvi korak u primjeni *Sass*-a na projekt jest, kao što je u prijašnjem poglavlju navedeno, dodavanje mape projekta u Koala aplikaciju.



Slika 2- Koala početni prozor

Sama aplikacija poprilično je jednostavna i jedini korak koji je potrebno poduzeti s Koalom jest dodavanje mape projekta sa svim pripadajućim datotekama. Nakon dodavanje mape projekta, sve datoteke koje Koala nadgleda prikazane su u prozoru aplikacije.



Slika 3 - Koala nakon dodavanja mape projekta

Koala se nakon toga može minimizirati, a sama će aplikacija i dalje raditi u pozadini i nadgledati promjene unutar mape projekta, kao što je prije objašnjeno. S time je priprema gotova te se može započeti sa stilskim oblikovanjem. Kao što je na slici 3 vidljivo, postoje 3 *.css* datoteke u projektnoj mapi, no ona koja je za ovaj primjer bitna je *custom.css* datoteka.

Slika 4 - custom.css datoteka i pripadajuće CSS oznake

Prema slici 4, selektori se ponavljaju u nekoliko primjera, a i font *Segoe UI* spominje se u dva navrata. Ikone koje služe kao poveznica na Facebook i Twitter profil imaju određenu širinu i visinu od 48 piksela, no pseudo-klasom *:hover* određena je širina i visina od 64 piksela – što znači da u trenutku kada pokazivač miša pređe preko navedene ikone, njena širina i visina mijenjaju se s 48px na 64px, a kada se pokazivač miša makne s ikone ista se vraća na početnu širinu i visinu od 48px. Ta funkcija zauzima sveukupno četiri retka u *.css* datoteci, dva po ikoni.

Kad bi se radilo o kompliciranijoj stranici – web portalu, recimo, *.css* datoteka bila bi puno napuhanija, selektori i vrijednosti ponavljali bi se u velikom broju i snalaženje u *.css* datoteci bilo bi izuzetno teško, a jedna pogreška mogla bi imati teške posljedice zbog velikog broja *CSS* oznaka. *Sass* u takvim situacijama pruža izuzetnu pomoć autoru. U sljedećim poglavljima, koristeći obje *Sass* sintakse, pojednostaviti će se *.css* datoteka prikazana na slici 4.

7.1 PRIMJER SCSS SINTAKSE

Prvi korak je pregledavanje *CSS* oznaka i grupiranje istih te provjeravanje koje se deklaracije ponavljaju te se mogu izdvojiti kao varijable. Potom je potrebno u programu za uređivanje teksta, u ovom slučaju *Notepad ++*, stvoriti dvije nove *.scss* datoteke koje služe za *Sass* stilski jezik – *stil.scss* koja služi kao „glavna“ datoteka i *basic.scss* za izdvojene kratke oznake. Navedene datoteke potrebno je spremiti unutar mape projekta kako bi Koala aplikacija imala pristup istima. Nakon toga u Koala aplikaciji klikne se na dugme *Refresh*, nakon čega aplikacija pronalazi novonastale *.scss* datoteke. Potrebno je desnim klikom kliknuti na glavnu *.scss* datoteku te kliknuti opciju *Set Output Path*, nakon čega se otvara prozor u koji upisujemo naziv *stil.css* te datoteku spremamo u isti direktorij u kojem se nalazi *HTML* datoteka – time se prilikom spremanja *.scss* datoteke automatski kompilira navedena *stil.css* datoteka. Sljedeći korak je prepravljanje ili dodavanje putanje do *.css* datoteke unutar *HTML* datoteke. U zaglavlju *HTML* datoteke treba dodati oznaku *<link rel="stylesheet" href="stil.css" />*. Navedena oznaka u ovom slučaju traži datoteku *stil.css* u istom direktoriju u kojem se nalazi *HTML* datoteka.

Time se završava priprema datoteka za *Sass* i sukladno tome, sljedeći korak je pisanje same *Sass*, odnosno *SCSS* datoteke. Kao što je prije rečeno, vrijednosti koje se ponavljaju su veličine ikona za društvene mreže te font teksta. Povodom toga, pri vrhu *basic.scss* datoteke izdvajamo varijable *$small_icon; 48px;* i *$font: 'Segoe-UI';* te koristimo *@mixin* oznaku. Korištenjem oznake *@mixin* može se stvoriti novi, proizvoljni selektor bez potrebe korištenja istog u *HTML* datoteci. Unutar selektora *li.name* dva su svojstva koja vrše istu funkciju - *-moz-border-radius* i *border-radius*. Kako bi se smanjila količina svojstava unutar deklaracije selektora *li.name* koristi se oznaka *@mixin*, kojom se stvara selektor imena *obrub* s vrijednošću *$radius* te se u deklaraciju odabranog selektora uvrštavaju svojstva *–moz-border-radius, border-radius, -webkit-border-radius* i *–ms-border-radius* s vrijednošću *$radius* za svako svojstvo. To znači da je unutar selektora *li.name* sada dovoljno napisati oznaku *@include obrub(5px);* kako bi se sve 4 prije navedene vrijednosti primijenile na odabrani element. Nakon toga prelazi se na datoteku *stil.scss.* Na samom početku *stil.scss* datoteke koristi se naredba *@import 'basic.scss'* kojom se oznake unutar datoteke *basic.scss* unose u datoteku *stil.scss*. U svrhu korištenja što manje prostora, svaka oznaka s kratkim deklaracijama piše se u jednom redu. Selektori koji se ponavljaju ili imaju pod-selektore ugnježđuju se unutar selektora koji ih obuhvaća. Prema tome, *body, .center i nav.top-bar* sada zauzimaju svaki jedan red, a u selektore *#fb* i *#twit*, koji se odnose na ikone za društvene mreže, ugnježđuju se sve njima pripadne vrijednosti. Pošto je

jedina razlika između *#fb* i *#twit* svojstvo *float*, korištenjem naredbe *@extend #fb;* unutar selektora *#twit* preuzimaju se svojstva i vrijednosti selektora *#fb* i primjenjuju na selektor *#twit.* Nakon svojstva *@extend* pišu se svojstvo i vrijednost *float: left;* čime se autor štedi ponavljanja četiriju svojstava. Unutar selektora *li.name* ugnježđuju se oznaka *@include obrub(5px);* i selektori za njemu pripadni *h1* element i *:hover* pseudo-klasu. Umjesto vrijednosti koje se konstantno ponavljaju ubacuju se određene varijable, čime deklaracije *height: 48px;* i *width: 48px;* postaju *height: $small_icon; i width: $small_icon;* odnosno *height: $small_icon + 16;* i *width: $small_icon + 16;* za 64px veličinu. Korištenjem operatora poput +, -, *, /, i % vrše se matematičke funkcije kojima se vrijednosti svojstva mogu fleksibilnije odrediti bez upotrebe dodatnih varijabli. Primjerice, ako se vrijednost varijable *$small_icon* promjeni s 48px na 64px, svojstvo u samoj deklaraciji ne mora se posebno mijenjati jer compiler sam vrši izračun vrijednosti dotičnog svojstva – ukratko, korištenjem operatora smanjuje se količina posla.

Što se time dobije?

stil.scss

```
@import 'basic.scss';
body { text-align: justify; font-family: $font; }
.center { text-align: center; margin: 0 auto; }
#logo { height: 220px; width: 200px; }
nav.top-bar { height: 60px; padding-top: 7px; }
#fb {
    height: $small_icon;
    width: $small_icon;
    float: right;
    &:hover {height: $small_icon + 16; width: $small_icon + 16;}
}
#twit {
    @extend #fb;
    float: left;
}
li.name {
    @include obrub(5px);
    &:hover { background-color: #555; }
    h1 {font-family: $font;}
}
```

basic.scss

```
$small_icon: 48px;
$font: 'Segoe-UI';
@mixin obrub($radius) {
    -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
    -ms-border-radius: $radius;
    border-radius: $radius;
}
```

Slika 5- stil.scss (lijevo) i basic.scss (desno)

Direktno u usporedbi može se primijetiti kako je svaki selektor koji se ponavljao bolje organiziran te da se iskoristilo puno manje prostora u usporedbi s *CSS*-om – u *CSS* datoteci zauzeto je 38 redova, a s obje *SCSS* datoteke ukupno 28. Sintaksa je gotovo identična *CSS*-u, jedina razlika su dodatne funkcije koje *SCSS* pruža – mixinsi, operatori, import, extend, varijable i ugnježđivanje selektora, što dokazuje kako je *SCSS* doista jednostavan za korištenje ako je autor vješt s *CSS* stilskim jezikom. U sljedećem poglavlju na isti primjer primijeniti će se starija, uvučena *Sass* sintaksa.

7.2 PRIMJER SASS SINTAKSE

Kao što je prije navedeno, starija *Sass* sintaksa pruža koncizniji način stiliziranja *HTML* dokumenata, no sa sobom donosi i neka ograničenja i vlastita pravila. Najkraće objašnjeno, *Sass* sintaksa i *SCSS* sintaksa slične su u funkcionalnosti, no razlika između njih je ta što se kraće oznake u *Sass* sintaksi ne mogu pisati u jednom redu – svaka deklaracija i svaki selektor moraju se novim redom odvojiti od prethodne jer se vitičaste zagrade i točka-zarez ne koriste. Također, oznake poput *@include* i *@mixin* pišu se drugačije - *@mixin* je =, *@include* je +, a *@import* i *@extend* ostaju isti. Iako je *Sass* sintaksa puno preglednija, korištenjem *SCSS* sintakse oznake zauzimaju puno manje prostora. Pozitivni aspekt je što, osim navedenih razlika, *Sass* i *SCSS* nisu toliko različiti – *SCSS* je samo bliži *CSS*-u nego *Sass*. U sljedećih par koraka usporediti će se *SCSS* i *Sass* na primjeru iz prošlog poglavlja.

Kao što je u prethodnom poglavlju spomenuto, *SCSS* datoteka svela je 38 redova *CSS* oznaka na ukupno 28 redova unutar dvije datoteke. Za početak pisanja *Sass*-a potrebno je napraviti dvije nove datoteke na isti način kao u prošlom poglavlju. Potrebno je stvoriti nove datoteke u mapi u kojoj se nalazi *HTML* datoteka, a ekstenzije tih datoteke u ovom slučaju više nisu *.scss* nego *.sass*. Prema tome, dvije nove datoteke zovu se *stil.sass* i *basic.sass.* Nakon toga potrebno je otvoriti Koala aplikaciju i, kao sa *SCSS* datotekom u prošlom poglavlju, stisnuti desni klik na novonastalu *stil.sass* datoteku te kliknuti na *Set Output Path*, nakon čega upisujemo ime *CSS* datoteke koja će se kompilirati od *stil.sass* datoteke – a to je *stil.css* – te ju smještamo u isti direktorij u kojemu se nalazi *HTML* datoteka. Nakon toga sve se oznake iz *stil.scss* i *basic.scss* datoteka kopiraju u pripadajuće *.sass* datoteke. Sljedeći korak u pretvaranju *SCSS* sintakse u *Sass* sintaksu jest brisanje nepotrebnih znakova – točke-zareza i vitičastih zagrada. Nakon toga potrebno je *CSS* oznake koje zauzimaju samo jedan red razdvojiti sukladno pravilima *Sass* sintakse – svaka deklaracija treba ići u zaseban, uvučeni red, naredba *@extend* ostaje ista, a umjesto *@include* oznake koristi se znak +. Varijable postavljene u *basic.sass* datoteci funkcioniraju u obje sintakse koje *Sass* stilski jezik pruža, stoga ih nije potrebno uređivati – osim, naravno, brisanja točke-zareza, a umjesto *@mixin* oznake piše se znak =.

Što se time dobije?

stil.sass

```
@import basic.sass
body
    text-align: justify
    font-family: $font
.center
    text-align: center
    margin: 0 auto
#logo
    height: 220px
    width: 200px
nav.top-bar
    height: 60px
    padding-top: 7px
#fb
    height: $small_icon
    width: $small_icon
    float: right
    &:hover
        height: $small_icon + 16
        width: $small_icon + 16
#twit
    @extend #fb
    float: left
li.name
    +obrub(5px)
    &:hover
        background-color: #555
    h1
        font-family: $font
```

basic.sass

```
$small_icon: 48px
$font: 'Segoe-UI'
=obrub($radius)
    -webkit-border-radius: $radius
    -moz-border-radius: $radius
    -ms-border-radius: $radius
    border-radius: $radius
```

Slika 6- stil.sass (lijevo) i basic.sass (desno)

Ono što se odmah na početku može primijetiti je koliko više redova *Sass* sintaksa zauzima u usporedbi sa *SCSS* sintaksom – broj redova popeo se s 28 na 36. Također, ono što je odmah uočljivo je koliko je *Sass* sintaksa preglednija od *SCSS* ili pak same *CSS* sintakse. Korištenje novih i uvučenih redova za odvajanje selektora i pripadnih deklaracija lišava autore pisanja vitičastih zagrada i točke-zareza nakon svakog svojstva i njegove vrijednosti, što stilsku datoteku ujedno čini preglednijom, ali i zahtjeva manje posla. Dakako, kod kompliciranijih stilskih datoteka koje koriste više funkcija od primjera prikazanog u ovom radu *SCSS* ima veću prednost zbog veće sličnosti sa samim *CSS-om*. *Sass* sintaksa u takvim situacijama koristi potpuno drugačije selektore za određene funkcije, što zahtjeva učenje i navikavanje na pisanje različitih oznaka od onih na koje je autor navikao. U svakom slučaju, *Sass* sintaksa pruža određene prednosti naspram osnovnog *CSS-a*, što je dobrodošlo bez obzira na pokoji problem koji dolazi u paketu. Što se iz priloženih primjera može zaključiti?

8. ZAKLJUČAK

CSS je već dugi niz godina standard za stilsko uređivanje web stranica, no ne pruža neke funkcije koje su u današnje vrijeme nužne – bilo za uštedu vremena ili za napredno uređivanje *HTML* datoteka. Kao rješenje na taj problem postoje brojni *CSS* preprocesori, koji *CSS-u* pružaju dodatne funkcije koje autorima olakšavaju rad te pružaju naprednije načine uređivanja njihovog web sadržaja. *Sass* je jedan od navedenih *CSS* preprocesora – štoviše, *Sass* je najpopularniji među njima i puno korisniji od samog *CSS*-a. Instalacija i korištenje istog ni u najmanju ruku nije komplicirano, a prednosti korištenja *Sass*-a brojne su i korisne. Kao što je u radu navedeno, *Sass* stilski jezik pruža dvije slične, ali opet različite sintakse – *Sass* i *SCSS*. Svaka sintaksa ima svoje prednosti i mane, no sa sigurnošću se može reći da nijedna nije bolja od druge. Ono što autore privlači određenoj sintaksi jednostavno je osobni stil pisanja *HTML-a* i *CSS-a* – nekima je korisnije zauzeti što manje redova u datoteci, nekima je bitnija preglednost i što manje kompliciranе oznake. Neovisno o stajalištu prema određenoj sintaksi, *Sass* pruža brži, jednostavniji, pregledniji i moćniji način stiliziranja *HTML-a*. No, *Sass* ni u kom pogledu nije standard za stilsko uređivanje *HTML* datoteka. Razlog tome je činjenica da je *Sass* preprocesor relativno nova ekstenzija za *CSS* i upravo činjenica da je to samo ekstenzija za već postojeći stilski jezik – bez obzira koliko se korisnijom pokazuje. U preglednicima ne postoji podrška za *Sass*, samo podrška za *CSS*, stoga *Sass* teško može postati nešto više od toga što trenutno je. Tehnički gledano, *Sass* nema vlastitu semantiku i sintaksu – praktički je identičan *CSS*-u pa, kao takav, nema šansu zamijeniti *CSS* stilski jezik. Ono što *Sass* je, doduše, jest nezamjenjivi alat autorima i kao takav zaslužuje popularnost i širu upotrebu, što je nešto čemu bi *web framework*-ovi itekako mogli i trebali pridonijeti.

9. LITERATURA

Cascading Style Sheets: level 1, 17.12.1996. URL: http://www.w3.org/TR/REC-CSS1/ (4.8.2014.)

Hunt, Ben. Inheritance and Cascading Styles in CSS Explained. URL: http://webdesignfromscratch.com/HTML-CSS/CSS-inheritance-cascade/#1 (4.8.2014.)

Install SASS. URL: http://Sass-lang.com/install (16.8.2014.)

Koala. URL: http://koala-app.com/(16.8.2014.)

Kyrnin, Jennifer. What is the difference between CSS2 and CSS3: The major changes to CSS3. URL: http://webdesign.about.com/od/CSS3/a/differences-CSS2-CSS3.htm (5.8.2014.)

Lazaris, Louis. CSS3's 'space' and 'round' values for background-repeat, 15.8.2011. URL: http://www.impressivewebs.com/space-round-CSS3-background/(5.8.2014.)

Lee, Xah. CSS1 and CSS2 Differences, 2005. URL: http://xahlee.info/js/CSS2.HTML (5.8.2014.)

Long, John W. Sass vs SCSS: Which syntax is better?, 19.2.2011. URL: http://theSassway.com/editorial/Sass-vs-SCSS-which-syntax-is-better (16.8.2014.)

Ossenbruggen, Jacco van; Lynda Hardman; Lloyd Rutledge; Anton Eliens. Style Sheet Languages for Hypertext, 1997. URL: http://homepages.cwi.nl/~jrvosse/publications/1997/dv:siglink.pdf (4.8.2014.)

Sass: Documentation. URL: http://Sass-lang.com/documentation/file.*sass*_REFERENCE.HTML (16.8.2014.)

Walsh, Nick. CSS Preprocessors: Focused Decisions. URL: http://www.awwwards.com/CSS-preprocessors-focused-decisions.HTML (6. 8. 2014.)

Wium Lie, Hakon; Bos, Bert. Cascading Style Sheets: designing for the Web, 1999. URL: http://www.w3.org/Style/LieBos2e/history/Overview.HTML (1.8.2014.)

Wium Lie, Hakon. Cascading HTML style sheets: a proposal, 10.10.1994. URL: http://www.w3.org/People/howcome/p/cascade.HTML (1.8.2014.)