Tallinna Ülikool
Informaatika Instituut

CSS'i laiendusvõimalused läbi Sass'i ning selle kasutamine veebirakenduste loomisel

Seminaritöö

Autor: Martin Koidu
Juhendaja: Jaagup Kippar

Tallinn 2012

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.... (kuupäev) / Martin Koidu /

Sisukord

Sissejuhatus

Internetist on mõne aastakümnega saanud enamike inimeste igapäevaelu lahutamatu osa. Inimesi, kes toodavad internetti igapäevaselt sisu on meie hulgas väga palju. Ettevõtteid, kes tegelevad internetis olevate dokumentide kujundamise ning veebilehekülgede loomisega on lugematu arv. Just selliste ettevõtete *front-end* arendajate seas on viimastel aastatel olnud jutuks mitmesugusid CSS'i laiendusvõimalusi pakkuvad metakeeled.

Käesoleva seminaritöö eesmärgiks on anda ülevaade ühe sellise metakeele, *Sass-i* poolt pakutavatest võimalustest CSS'i funktsionaalsuse laiendamiseks veebilehekülgede ning teiste HTML-vormingus dokumentide kujundamisel. Tehnoloogia tutvustuse juurde kuuluvad koodinäited, mille soovija saab vastavalt vajadusele ise läbi proovida. Seminaritöö lõpus asub kogu kõnealust tehnoloogiat kokkuvõttev näidislehestik.

Autor on valinud kõnealuse teema oma seminaritööks kuna antud tehnoloogia on hetkel suhteliselt uus ning materjali selle kohta leidub vähe. Eestikeelsed väljaanded käesoleva töö teema kohta puuduvad hetkel täielikult. Samuti on üheks argumendiks kõnealuse teema valikul autori suur huvi veebitehnoloogiate vastu.

Seminaritöö on jaotatud neljaks peatükiks. Esimeses peatükis antakse ülevaade CSS'i olemusest ning selle peamistest puudustest. Teises osas räägitakse, mida kujutab endast Sass, kuidas ja miks see tekkinud on, kuidas seda oma arvutisse saada ning milliste võimalike negatiivsete külgede osas tuleks olla tähelepanelik. Neljandas peatükis asub autori poolt loodud näidislehestik, mis aitab mõista, milliseid võimalusi Sass reaalses situatsioonis pakub.

Töös on kasutatud koodinäidetena autori enda teadmisi ning internetis leiduvaid õpetusi. Seminaritöö juurde kuulub CD-plaat, millel on saadaval kõik töös olevad koodinäited ning näidislehestik. Kõik koodinäited ning näidislehestik on saadaval järgmisel aadressil: http://www.tlu.ee/~strin/seminaritoo/.

1 CSS

1.1 Mis on CSS?

CSS on üks peamiseid tehnoloogiaid, mida veebirakenduste loomisel kasutatakse. Lühend CSS on pikalt välja kirjutades *Cascading Style Sheets* (astmelised stiililehed).

CSS'i ülesandeks on määrata, kuidas dokumente vaatajale näidatakse (värvid, elementide paiknemine dokumendis, kirjastiilid jne). Samuti aitab CSS hoida lahus dokumentide sisulist informatsiooni sellest, kuidas seda lõpuks kasutajale esitletakse. Teisisõnu, kujundus ja struktuur on üksteisest lahutatud. Osa, mis kirjeldab, kuidas dokumenti näidatakse, nimetatakse stiiliks (*style*). Hoides stiili sisust eraldi, aitab see kaasa info lihtsamale haldamisele. Samuti aitab vältida kordusi ning sama sisu saab lihtsasti kasutada erinevatel eesmärkidel. Näiteks kui kasutaja siseneb veebilehele, näidatakse sisu ühe stiiliga. Juhul kui kasutaja soovib sama lehekülge välja printida, kasutatakse teist stiili selleks, et mitte näidata lehel olevaid pilte, teha teksti paberilt lugemiseks suuremaks või mõjutada sisu esitamist mõnel muul viisil. (Why use CSS? - CSS | MDN, 2012)

Astmelisus

Eelneva põhjal teame, mis on CSS'i ülesandeks. Pöördudes tagasi selle tehnoloogia nimetuse, astmelised stiililehed juurde, siis võib tekkida küsimus, mida mõeldakse sõna "astmelised" all.

Astmelised tähendab seda, et CSS lubab ühe dokumendiga siduda mitu erinevat stiililehte, mis omakorda loob olukorra, kus tõenäoliselt on nende stiililehtede peale kokku mitmeid konfliktseid deklaratsioone. Näiteks võib olla veebilehekülgega seotud kaks stiililehte, mis mõlemad deklareerivad, kuidas peaks välja nägema esimese taseme pealkiri (h1). Siin tuleb mängu astmelisus. Näiteks on leht, mille autor on stiililehega määranud, et esimese taseme pealkiri peaks olema suurusega 22 pikslit ning roheline. Sama lehega on tegelikult seotud veel ka veebilehitseja enda stiilileht, milles on deklareeritud esimese taseme pealkiri, mis on musta värvi ja suurusega 18 pikslit. Nüüd kui külastaja läheb oma veebilehitsejaga konkreetsele lehele, siis näidatakse talle ikkagi

rohelist 22 piksli suurust pealkirja. Seda sellepärast, et veebilehitseja enda stiililleht on taseme poolest madalamal astmel kui veebilehe looja poolt koostatud stiililleht. (Robbins, 2007)

Astmelisuse kohta saab lähemalt lugeda järgmiselt aadressilt: http://www.blooberry.com/indexdot/css/topics/cascade.htm

1.1.1 Reeglid

CSS'i reeglid võib jagada kaheks osaks. Esimene osa kannab nimetust selektor (*selector*). Selektori eesmärgiks on määrata, millist elementi dokumendis soovime parajasti kujundada. Üldiselt on selektoriks mingisugune HTML-dokumendis olev märgend (*tag*) kuid natuke suurema lehestiku puhul võib selektorit kirjeldada veel spetsiifilisemalt. Näiteks saab valida kasutaja poolt täitmisele minevast registreerimisvormist kõik tekstikastid, mille juures on ära määratud atribuut class="required" ning vastavalt sellele kujundada ainult seda atribuuti omavad tekstiväljad jättes kõik ülejäänud puutumata.

Teine osa kannab nimetust omadus (*property*) ning see annab veebilehitsejale teada, millist parameetrit me selle elemendi juures kujundada soovime. Parameetriks võib olla näiteks teksti suurus, värv, paigutus, reavahe või mõni muu kujundatava elemendi omadus. Omaduse juurde kuulub alati ka vastav väärtus (*value*), mis määrab, kui palju ning kuidas täpselt seda parameetrit soovitakse kujundada. Näiteks teksti suurus defineeritakse pikslites. Omadused ja nende väärtused kirjutatakse alati loogeliste sulgude vahele. (Schmitt, 2010)

Kõige lihtsam CSS'is kirjutatud reegel näeb välja järgmine:

```
p {
    color: red;
}
```

Koodinäide 1.1. *CSS*'is defineeritud reegel.

Eelneva reegliga määratakse ära lehel oleva paragrahvi teksti värv punaseks. Selle konkreetse reegli puhul on selektoriks “p”, mis tähistab HTML-dokumendis paragrahvi. Omaduseks, mida soovitakse muuta on teksti värv ehk “color”, mille väärtuseks on punane, “red”.

1.1.2 Stiilifaili sidumine HTML-dokumendiga

CSS’i seostamiseks HTML’is kirjeldatud struktuuriga on mitmeid erinevaid võimalusi. Üldiselt kirjutatakse CSS eraldi faili ja seejärel seotakse HTML-dokumendiga kasutates selleks otstarbeks ettenähtud koodi (vt. koodinäide 1.2). Käesolevas seminaritöös kasutatakse sama lähenemist.

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

Koodinäide 1.2. Stiilifaili sidumine HTML-dokumendiga

Täpsemat infot selle kohta, kuidas stiilifail oma dokumendiga siduda saab järgmisest koodinäitest:

```
<!DOCTYPE HTML>
<html lang="et">
<head>
    <meta charset="UTF-8">
    <title>Tutvus CSS'iga</title>
    <link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
    <h1>Tutvus CSS'iga</h1>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit.
    </p>
</body>
</html>
```

Koodinäide 1.3. Lihtne HTML-dokument, millega on seotud stiilifail "style.css".

Kuna käesoleva seminaritöö eesmärgiks ei ole tutvustada võimalusi, mida pakub tavapärane CSS. , siis siinkohal peaks lugeja, kellel on CSS’iga vähe kokkupuudet tutvuma internetis leiduvate materjalide ning õpetustega.

Eestikeelsete õpetuste hulk ei ole küll väga suur, kuid kindlasti väärivad väljatoomist järgmised materjalid:

- http://www.cs.tlu.ee/~rinde/www_materjal/css.pdf – kokkuvõtlik juhend, mis annab ülevaate enimkasutatavatest CSS keelde kuuluvatest vahenditest;
- http://www.kakupesa.net/kakk/veebiABC/index.php – Ülevaatlik juhend, mis käsitleb lisaks CSS'i põhifunktsionaalsusele ka HTML'i. Hea juhend alustajale;
- http://et.wikibooks.org/wiki/CSS – samuti kokkuvõtlik juhend paljude näidetega.

Ingliskeelset materjali leidub tunduvalt rohkem. Järgnevalt on välja toodud autori arvates ühed paremad tasuta saadaval olevad õpetused:

- http://www.w3schools.com/css/default.asp – väga põhjalik ülevaade, mis katab nii levinumad kui vähemlevinumad võimalused;
- http://www.w3schools.com/css3/default.asp – tutvustab täiendavaid võimalusi, mis on lisandunud CSS3-e tulekuga.

1.2 Lühiülevaade ajaloost

90ndate aastate keskpaigas kui internetist oli saanud laialdasemalt tarbitav meedium, tekkis kasutajatel vajadus saada suuremat kontrolli HTML-dokumentide kujundusliku poole üle. HTML'i vahendid ei rahuldanud professionaalse ajakirjanduse vajadusi luua ajakirja või ajalehte meenutavat veebis esitatavat dokumenti. Kuigi CSS oli olemas juba HTML'i alguspäevadest aastal 1990, puudus kindel standard, mis oleks määranud, kuidas CSS'is kirjutatud kujundusreeglid peaksid välja nägema. See tõi aga kaasa olukorra, kus erinevad veebilehitsejate tootjad kasutasid oma toodetes enda nägemust CSS reeglistikust, mis tegi keeruliseks veebiloojate töö, kes pidid tagama, et inimesed saaksid info kätte olenemata sellest, mis programmi nad veebi sirvimiseks kasutasid. (York, 2005)

Aastal 1994 asutas veebi loojaks peetav ning esimese veebilehitseja autor Tim Berners-Lee organisatsiooni nimega World Wide Web Consortium. Vaid mõned päevad hiljem

avaldas Håkon Wium Lie oma ettepaneku[1] selle kohta, kuidas HTML-dokumendid võiksid olla kujundatud kasutades lihtsaid deklaratsioone. (Lie & Bos, 1999)

Håkon'i mõttega ühines ka Bert Bos, kes oli samal ajal välja töötamas oma visiooni reeglistikust. 1996. aastal tuldi koos välja dokumendiga, millest sai W3C soovitus ning mis kandis nime Cascading Style Sheets, level 1[2] (CSS1). Esimene versioon CSS'ist sisaldas elementaarseid võimalusi määrata teksti värvi ning suurust, dokumendi taustavärvi või pilti jne. Kuigi algselt oli neid, kes uuele standardile vastu seisid, võttis interneti kogukond selle lõpuks vastu. (Lie & Bos, 1999)

Ajal, mil CSS'i väljatöötamine alles pooleli oli, hakkasid veebilehitsejate tootjad looma aga omalt poolt võimalusi, mis lubaksid kontrollida dokumendi kujunduslikku poolt. Kuigi HTML ei olnud loodud selleks, et kontrollida otseselt dokumendi välimust, oli temast saamas selliste omadustega keel. Peale seda kui CSS1 muutus W3C ametlikuks soovituseks, jäid HTML'i ikka veel elemendid, mis lubasid sisu kujundada. Selliseid võimalusi leidub HTML'is veel tänini. (York, 2005)

1997. aasta veebruaris loodi W3C-s eraldi töörühm, kelle ülesandeks sai töö lahendustega, mida CSS1 veel ei võimaldanud. CSS level 2 sai W3C poolt soovituslikuks 1998 aasta mais. (Lie & Bos, 1999)

CSS level 2[3] tõi endaga kaasa päris suure hulga täiendusi. Lisandusid erinevad meediatüübid vastavalt sellele, kuidas dokumenti esitada soovitakse (printides paberile, ekraanil vms). Samuti lisandusid elementide paigutamisel (*positioning*) sellised nagu tüübid relatiivne (*relative*), absoluutne (*absolute*) ning fikseeritud (*fixed*). Selektorite hulk täienes läbi uute pseudo-klasside nagu `:first-child`, `:hover`, `:focus`, `:lang`. (Bos, Lie, Lilley, & Jacobs, 2008)

Praegu on kasutusel CSS'i redaktsioon 2.1, mis loodi selleks, et parandada CSS level 2-ga kaasa tulnud vigu ning puuduseid. Võrreldes CSS2-ga eemaldati osad funktsioonid

[1] http://www.w3.org/People/howcome/p/cascade.html
[2] http://www.w3.org/TR/REC-CSS1/
[3] http://www.w3.org/TR/2008/REC-CSS2-20080411/

täielikult. Teistes tehti lihtsalt muudatusi ning osa funktsioone toodi CSS2.1 spetsifikatsiooni otse, muutmata kujul üle. (The World Wide Web Consortium, 2011)

Praegu on väljatöötamisel CSS versioon 3, mis erinevalt eelnevatest versioonidest on muudetud moodulipõhiseks. Iga moodul on iseseisev spetsifikatsioon, mille kallal töötab kindel grupp inimesi ning mis muutub W3C poolt soovituslikuks eraldiseisvalt olenemata sellest, mis staadiumis on teised moodulid. Nagu varasemate versioonide puhul, on ka CSS3 puhul mooduleid, mis on üle toodud eelnevast CSS'i versioonist. Siia hulka kuuluvad näiteks meediapäringud[4] (*media queries*), mis on juba praegu täielikult välja arendatud ning muutunud ametlikult soovituslikuks. Mõned uued moodulid, mis CSS3-ga kaasa tulevad on uued selektorid, taustad ja piirjooned (*backgrounds and borders*), 2D/3D transformatsioon[5], animatsioonid[6], mitmeveeruline asetus[7] (*multiple column layout*) (Gillenwater, 2011)

Erinevate moodulite tööjärge, saab jälgida järgmisel aadressil:
http://www.w3.org/Style/CSS/current-work

1.3 Peamised puudused

Stiilifailide haldus suuremate projektide puhul

Kuigi esmapilgul tundub kõik hästi olevat on ka CSS'il mitmesugused puudused. Esiteks on suuremate stiilifailide haldamine küllaltki raske ning aeganõudev. Tihtipeale tuleb selleks, et teha dokumendi kujunduses muudatusi, muuta stiilifailis kirjeldatud elemente mitmel, suuremate projektide puhul võibolla ka mitmekümnel või lausa mitmesajal erineval real. See on aga ajamahukas tegevus eriti juhul kui on vaja muuta elemendi suurust, millega on seotud veel mõni teine element lehel. Samuti tuleb ette olukordi, kus stiilifailides peaks kasutama mitme erineva elemendi kujunduse määramisel sama koodi, mida juba varem on kasutatud.

[4] http://www.w3.org/TR/css3-mediaqueries/
[5] http://www.w3.org/TR/css3-transforms/
[6] http://www.w3.org/TR/css3-animations/
[7] http://www.w3.org/TR/css3-multicol/

HTML-prototüüpimine

Kuna CSS'i loomisel ei ole arvestatud elementaarse koodi taaskasutamisega, siis on HTML-prototüüpimine väga aeganõudev. Muutujate puudumise tõttu kulub muudatuste tegemiseks palju aega ning sealjuures on vead kerged tulema.

CSS3 uue funktsionaalsuse kasutamine

Üks puudus, milles ei saa küll otseselt CSS'i ennast süüdistada, kuid mis on siiski sellega seotud, on CSS'i kolmanda versiooniga lisandunud uute, kuid ametlikult veel mitte kinnitatud moodulite kasutamine. Näiteks selleks, et saada gradientse taustavärviga kasti, mis oleks samasugune nii Firefox, Safari, Chrome, Opera kui ka Internet Exploreri erinevate versioonidega vaadates, tuleks kasutada järgmist deklaratsiooni:

```
div {
    background: -moz-linear-gradient(left,  #7abcff 0%, #60abf8 44%, #4096ee 100%); /* FF3.6+ */
    background: -webkit-gradient(linear, left top, right top, color-stop(0%,#7abcff), color-stop(44%,#60abf8), color-stop(100%,#4096ee)); /* Chrome,Safari4+ */
    background: -webkit-linear-gradient(left,  #7abcff 0%,#60abf8 44%,#4096ee 100%); /* Chrome10+,Safari5.1+ */
    background: -o-linear-gradient(left,  #7abcff 0%,#60abf8 44%,#4096ee 100%); /* Opera 11.10+ */
    background: -ms-linear-gradient(left,  #7abcff 0%,#60abf8 44%,#4096ee 100%); /* IE10+ */
    background: linear-gradient(to right,  #7abcff 0%,#60abf8 44%,#4096ee 100%); /* W3C */
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#7abcff', endColorstr='#4096ee',GradientType=1 ); /* IE6-9 */
}
```

Koodinäide 1.4. CSS3'is defineeritud gradientne värv.

Allikas: http://www.colorzilla.com/gradient-editor/

Tulemuseks saame alljärgneva gradientse taustavärviga kasti:

Ekraanipilt 1.1. Tulemus

Teises peatükis tutvustatav tehnoloogia “Sass” pakub selliste probleemidega tegelemiseks mugavaid vahendeid. Ühte sellist vahendit käsitletakse peatükis 3.4.

2 SASS

2.1 Mis on Sass?

Sass ei ole tavapärase CSS'i asendus vaid vahend, mis aitab seda kiiremini ja efektiivsemalt kirjutada ning teeb suuremate projektide kujundusfailid hõlpsamini hallatavaks. (Catlin & Catlin, Pragmatic Guide to Sass, 2011)

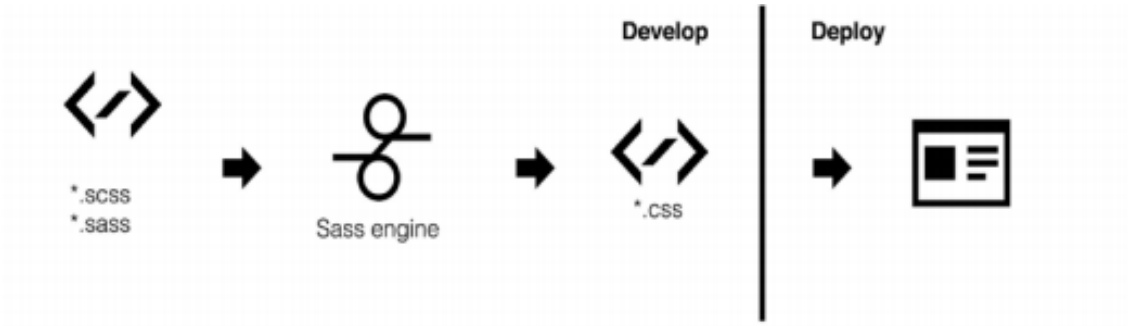
Sass (*Syntactically Awesome Style Sheets*) on oma olemuselt metakeel, mille kood kompileeritakse tavaliseks CSS koodiks. Sass laiendab tavalist CSS'i võimaldades kasutada mitmesuguseid kasulikke ning arendamist kiirendavaid funktsioone, millede hulka kuuluvad näiteks muutujad (*variables*), üksteise sees paiknevad reeglid (*nested rules*), koodiosade taaskasutus (Mixin Directives), mitmesugused funktsioonid värvide ja teiste väärtustega manipuleerimiseks, aritmeetilised operatsioonid, loogikaavaldised jne. Selle tulemusena on suurte stiilifailide haldamine ning organiseerimine tavalise CSS´iga võrreldes lihtsam ja kiirem. Lisaks eelnevale lisab Sass võimaluse varasemalt valmis kirjutatud koodi hõlpsasti taaskasutada, mis aitab samuti aega märgatavalt kokku hoida. Sass'i poolt lisatavad tsüklid ning loogikaavaldised teevad mugavamaks ka paindlike raamistike kirjutamise, mis vähendavad tunduvalt koodi kirjutamiseks kuluvat aega. (Catlin, Eppstein, & Weizenbaum, File: SASS_REFERENCE, 2012)

Sass'is kirjutatud koodi võib kutsuda ka nimega SassScript.
Täpsemalt vaadeldakse Sass'iga lisanduvaid võimalusi kolmandas peatükis.

2.2 Kuidas Sass töötab?

Kasutades Sass'i, kompileeritakse kõik lähtekoodis olevad stiilifailid ümber veebilehitsejatele vastuvõetavaks CSS-koodiks, mis seejärel seotakse HTML-dokumendiga (vt joonis 2.1). Käesoleval hetkel ei toeta ükski veebilehitseja Sass'i failide tõlgendamist otse ning tõenäosus, et selline veebilehitseja kunagi üldse ilmavalgust näeb, on väike. Seega on tulemuseks tavaline staatiline CSS fail selle vahega, et läbi esialgse lähtekoodi saab arendaja ära kasutada kõiki Sass'i poolt pakutavaid

lisavõimalusi. (Netherland, Weizenbaum, Eppstein, & Mathis, 2012)



Joonis 2.1. Arendusprotsess kasutades Sass'i.

(Netherland, Weizenbaum, Eppstein, & Mathis, 2012)

2.3 Ajalugu

CSS on loomu poolest väga lihtne: fail, kus on kirjeldatud reeglid, mille järgi paneb veebilehitseja paika dokumendi kujunduse. Suuremate ja keerulisemate lehtede puhul muutus aga väga mahukate stiilifailide haldamine aeganõudvaks ning ebamugavaks.

Otsides võimalust olukorra parandamiseks, tulid Hampton Catlin ja Nathan Weizenbaum 2007. aastal välja lahendusega, mis aitaks kaasa situatsiooni parandamisele. Uue lahenduse nimeks sai Sass (Syntactically Awesome Style Sheets).

Sass on kirjutatud kasutades programmeerimiskeelt Ruby. Esialgne versioon Sass'ist oli küllaltki erinev sellest, mida kasutatakse käesoleval hetkel. Algselt puudusid Sass'i süntaksis loogelised sulud ning deklaratsioonid pidid olema trepitud kasutades kindlat arvu tühikuid. (Demaree, 2011)

Loodud süntaksil oli aga üks suur viga, millega Sass'i autorid ei olnud arvestanud. Nimelt ei olnud selline süntaks kokkusobiv tavapärase laialt kasutuseloleva CSS'iga, mistõttu pidid suuremate veebilehekülgedega töötavad arendajad tegema palju lisatööd, et kirjutada juba olemasoleva projekti kujunduse kood ümber Sass'i. (Demaree, 2011)

Alates Sass'i versioonist 3.0 võeti paralleelselt vana süntaksiga kasutusele uus, mis hakkas kandma nime SCSS (*Sassy CSS*). Uue süntaksi eeliseks oli see, et ta oli olemasoleva CSS'i ülemhulk, mis praktikas tähendab, et kogu valideeruv CSS kood on ühtlasi ka valideeruv SCSS kood. Uue süntaksi kasutuselevõtt tegi vanade projektide puhul Sass'ile ülemineku märgatavalt kergemaks. (Demaree, 2011)

2.4 Sass'i paigaldamine arvutisse

Enne kui Sass'i võimalusi saab katsetama hakata, peab selle arvutisse paigaldama. Kuna Sass on kirjutatud kasutades programmeerimiskeelt Ruby, tuleb Sass'i kasutamiseks ka see oma arvutisse paigaldada.

2.4.1 Microsoft Windows

Windowsi kasutajad peavad enne Sass'i paigaldamist installeerima oma masinasse Ruby. Kõige lihtsam viis selleks on kasutada RubyInstaller'it, mille saab järgmiselt aadressilt: http://rubyinstaller.org/downloads.

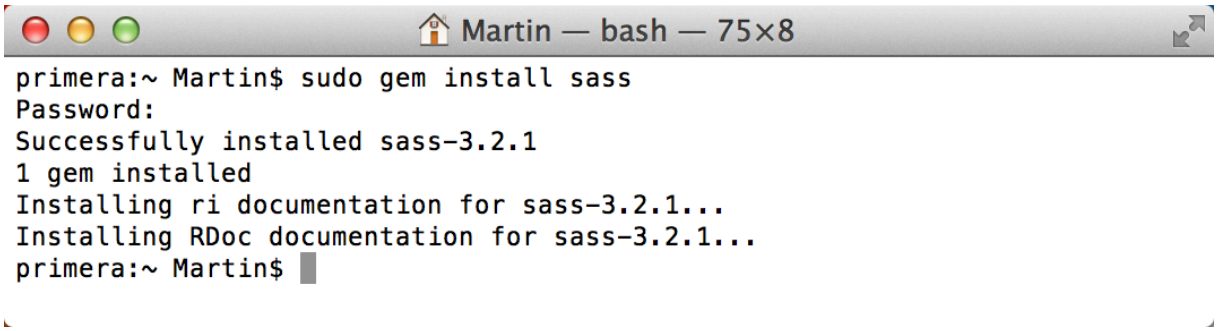
Kui Ruby on paigaldatud jääb üle vaid Sass installeerida. Selleks tuleb avada käsurida ning trükkida avanevasse aknasse järgmine käsk:

```
gem install sass
```

2.4.2 Mac OS X

Mac OS X'i puhul on paigaldamine erinevalt Windowsi omast lihtsam kuna seal on Ruby juba eelnevalt installeeritud. Jääb ainult installida Sass, mis käib läbi rakenduse Terminal. Selleks, et Sass oma arvutisse paigaldada, on vaja trükkida Terminali aknasse:

```
sudo gem install sass
```



Ekraanipilt 2.1. Sass'i paigaldamine Mac OS X keskkonnas.

2.4.3 Linux

Linuxi puhul on esimese asjana vaja installeerida omale Ruby, mis käib läbi paketihaldusrakenduse. Peale Ruby paigaldamist käib Sass'i paigaldamine sarnaselt Mac OS X'le, läbi käsurea.

2.5 Süntaks

Sass'i kasutamiseks on olemas kaks erinevat süntaksit. Esimene neist kannab nimetust *SCSS* (Sassy CSS*).* Kasutades seda varianti näeb kirjutatud kood välja kui tavaline CSS, kuid vaatamata sellele on tal olemas Sass'i poolt lisatud laiendused. Selline süntaks teeb Sass'i kasutuselevõtu väga lihtsaks inimestele, kes on varem CSS'i kirjutanud. Nagu ikka ümbritsetakse koodi plokk loogeliste sulgudega "{" ja "}" ja deklaratsioonid semikooloniga (vt koodinäide 2.1). SCSS süntaksis kirjutatud failid peavad kasutama laiendit "*.scss*".

```
.news-item {
    color: #ccc;
    font-size: 1.4em;
}
```

Koodinäide 2.1. SCSS süntaks.

Teine kasutusel olev süntaks on tuntud kui taandega süntaks (*indented syntax*). See süntaks Sass'iga kaasas olnud selle loomisest peale ning on oma olemuselt kompaktsem. Koodiblokkide eraldamiseks kasutakse tabulaatorit ning deklaratsioonide eraldamiseks uut rida. Failid, mis kasutavad taandega süntaksit peavad olema ".sass" laiendiga.

```
.news-item
    color: #ccc
    font-size:  1.4em
```

Koodinäide 2.2. SASS süntaks.

Funktsionaalsuse poolest ei ole vahet, kumba süntaksit kasutada. Peale kompileerimist on saadavaks tulemuseks puhas CSS (vt koodinäide 2.3).

```
.news-item {
    color: #cccccc;
    font-size: 1.4em;
}
```

Koodinäide 2.3. Kompileerimisel väljastatav CSS.

(Catlin, Eppstein, & Weizenbaum, File: SASS_REFERENCE, 2012)

Tegelikult võib ühe projekti raames kasutada mõlemat süntaksit. Sealjuures tuleb aga silmas pidada, et ühe faili raames peab piirduma ühe süntaksiga. (Netherland, Weizenbaum, Eppstein, & Mathis, 2012)

Kuigi funktsionaalsuse poolest ei ole neil vahet, siis on mõlemal süntaksil on olemas omad eelised ning puudused. Seda, kumb kellelegi rohkem sobib, võib otsustada igaüks ise. Käesolevas seminaritöös kasutatakse edaspidi uuemat süntaksit.

Tööriist, millele pääseb ligi Sass´i käsurealt, võimaldab ühte süntaksit kasutavat lähtekoodifaili ümber konvertida käsuga `sass-convert [sisendfail] [väljundfail]`

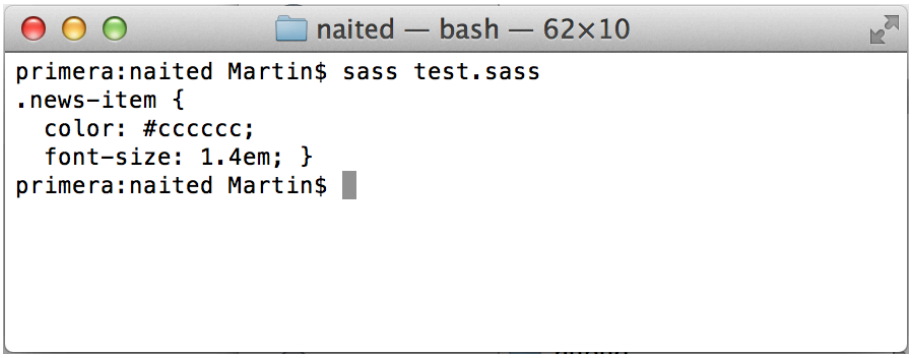
2.6 Sass'i kasutamine käsurealt

Kui Sass on eelnevalt arvutisse paigaldatud, siis saab lähtekoodi kompileerida läbi käsurearakenduse.

Esimeseks prooviks võib võtta koodinäite 2.2, mis tuleb salvestada nimega "test.sass".

Lähtekoodi kompileerimiseks tarvitseb käsureale kirjutada `sass test.sass`, millest esimese argumendina tuleb ette anda lähtekoodi sisaldava faili nimi.

Kui lähtekoodis vigu polnud, siis peaks avanema järgmine pilt:



Ekraanipilt 2.2. Lähtekoodi kompileerimine kasutades käsurida.

Peale käsu sisestamist kompileeriti lähtekoodi sisaldav fail ümber tavaliseks CSS'iks ning väljastati otse käsureale. Käsureale koodi väljastamisest suurt kasu ei ole. Parem oleks kui selle saaks kompileerida eraldi faili. Selle tegemiseks tuleb käsureaprogrammile ette teise argumendina väljundfaili nime.

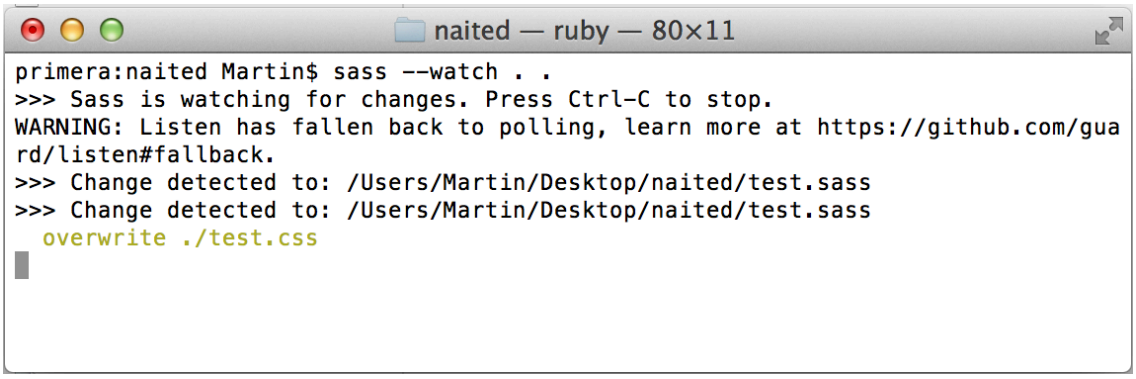
```
sass test.sass test.css
```

Peale eelneva käsu käivitamist peaks kataloogi olema lisaks lähtekoodi sisaldavale failile olema tekkinud fail "test.css", mille saab siduda otse HTML-dokumendiga.

Kuna lähtekoodi kompileerimine käsitsi peale iga muudatust on küllaltki ebamugav, on selleks puhuks Sass´iga kaasas võimalus kompileerida fail automaatselt peale iga muudatust failis. Selleks, et lasta Sass'il kompileerida lähtekood automaatselt tuleb käivitada järgmine käsk

```
sass --watch . .
```

Peale selle käsu käivitamist taustale jooksma protsess, mis kontrollib iga lühikese intervalliga, kas lähtekoodi failid on uuenenud. Juhul kui need on muutunud, siis kirjutatakse vanad väljundfailid üle. (Catlin & Catlin, Pragmatic Guide to Sass, 2011)



Ekraanipilt 2.3. Automaatne Sass failide compileerimine.

2.7 Väljundi formaadid

Vaikimisi kasutab Sass väljundfaili loomisel pesastatud formaati (*nested*). Kuna aga erinevatel põhjustel võib olla tarvis kasutada hoopis teistsugust formaati, siis on Sass'i ametlikul käsurearakendusel võimalik valida nelja erineva formaadi vahel: pesastatud (*nested*), laiendatud (*expanded*), kompaktne (*compact*) ning kokkupakitud (*compressed*).

Selleks, et panna Sass kasutama soovitud formaati on kasutusel käsk `sass --style [soovitud-stiili-nimi] [sisendfail] [väljundfail]`. Näiteks soovides näha tulemust laiendatud formaadis tuleb kirjutada `sass --style expanded input.scss output.css`

(Catlin & Catlin, Pragmatic Guide to Sass, 2011)

2.7.1 Pesastatud

Pesastatud (*nested*) stiil annab aimu sellest, milline on HTML- ning CSS-dokumendi struktuur. Iga reegel on pesastatud vastavalt sellele, milline on selektor ning kuidas paikneb element DOM'is. See on kasulik juhul kui on vaja vaadata suuremamahulisi faile kuna annab hästi edasi HTML-dokumendi struktuuri. (Catlin, Eppstein, & Weizenbaum, File: SASS_REFERENCE, 2012)

```
#main article {
  border-bottom: 1px #CCC dotted; }
  #main article p {
    color: #333333; }
```

Koodinäide 2.4. Kompileerimisel saadud tulemus kasutades pesastatud stiili.

2.7.2 Laiendatud

Laiendatud stiil on kõige sarnasem sellele, kuidas enamik CSS'i arendajad seda teevad. Laiendatud stiili on inimsilmal kõige mugavam lugeda. Kuna selline formaat muudab faili küllaltki mahukaks, siis on soovitav seda kasutada ainult arendustsükli vältel. (Catlin, Eppstein, & Weizenbaum, File: SASS_REFERENCE, 2012)

```
#main article {
  border-bottom: 1px #CCC dotted;
}
#main article p {
  color: #333333;
}
```

Koodinäide 2.5. Kompileerimisel saadud tulemus kasutades laiendatud stiili.

2.7.3 Kompaktne

Kompaktne formaat võtab tunduvalt vähem ruumi kui pesastatud ja laiendatud. Iga reegel asub failis eraldi real ning tänu sellele on tähelepanu koondunud pigem selektoritele kui deklaratsioonidele selle sees.

```
#main article { border-bottom: 1px #CCC dotted; }
#main article p { color: #333333; }
```

Koodinäide 2.6. Kompileerimisel saadud tulemus kasutades kompaktset stiili.

2.7.4 Kokkupakitud

Kokkupakitud formaat on inimsilmale kõige kehvemini loetav kuna eesmärgiks on ruumi kokkuhoid. Kõik tühikud ning teised tühimärgid on eemaldatud kõikidest võimalikest. Tühikud on säilinud ainult selleks, et eraldada selektoreid. Kogu väljastatav CSS asub ühel real. Samuti on värvid kirjutatud võimalikult lühidalt. Seda formaati tuleks kasutada projekti lansseerimisel. (Catlin, Eppstein, & Weizenbaum, File: SASS_REFERENCE, 2012)

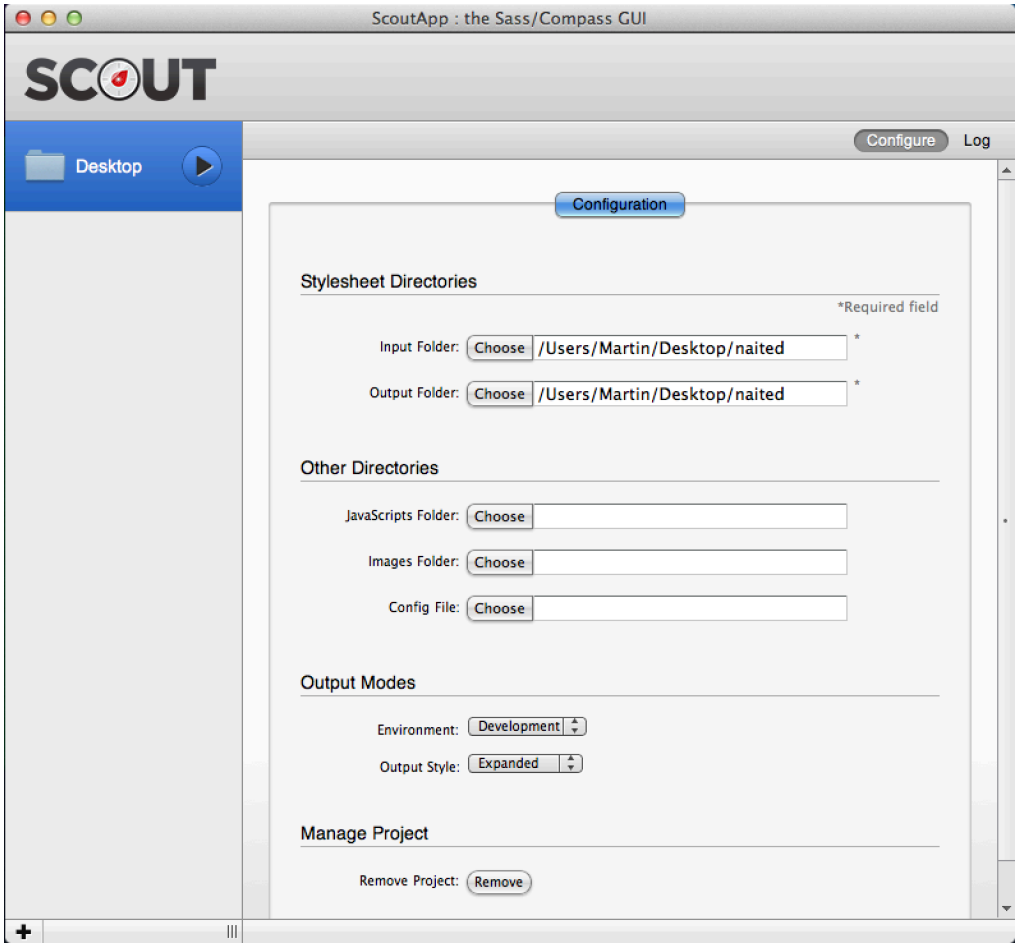
```
#main article{border-bottom:1px #CCC dotted}#main article
p{color:#333333}
```

Koodinäide 2.7. Kompileerimisel saadud tulemus kasutades kokkupakitud stiili.

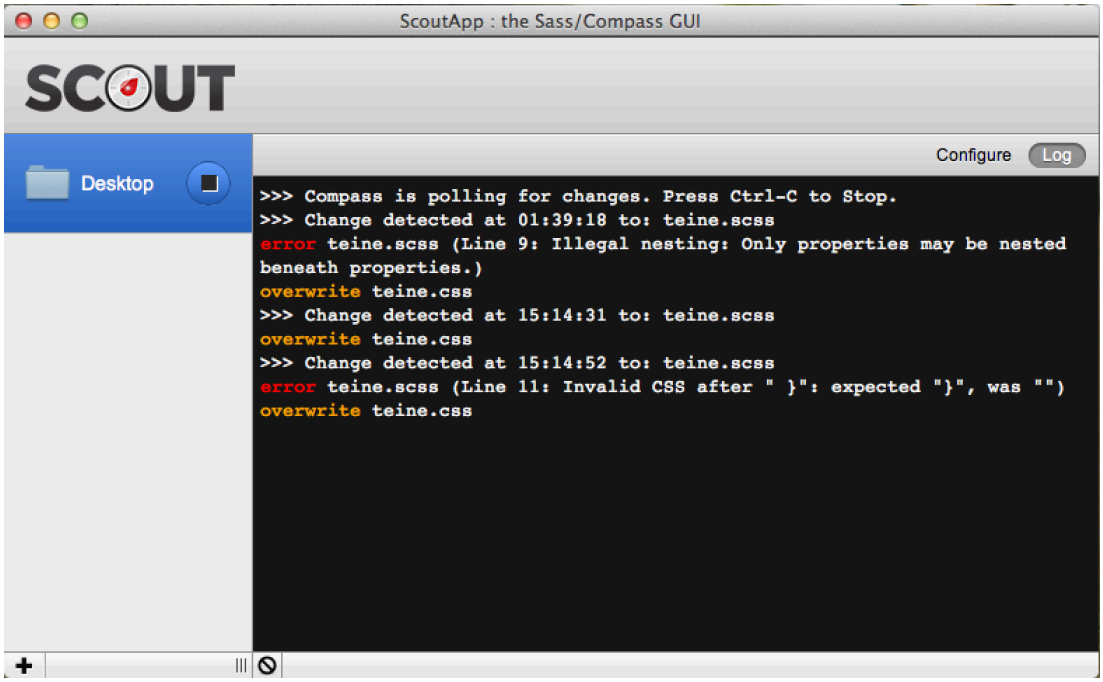
2.8 Sass'i kasutamine läbi rakenduse Scout

Kuna käsurida võib osades kasutajates tekitada vastumeelsust, siis on kolmandate osapoolte poolt loodud graafilise kasutajaliidesega vabavaraline rakendus Scout. Soovi korral saab programmi alla laadida aadressilt: http://mhs.github.com/scout-app/. See on saadaval nii OS X'i kui ka *Windows* keskonna jaoks. Üheks plussiks rakenduse Scout'i kasutamise juures on asjaolu, et Sass'i kasutamiseks ei ole vaja arvutisse paigaldada eraldi Ruby programmeerimiskeelt.

Scout on lihtne programm ning ei nõua kasutajalt käsurea kasutamise oskust. Scout töötab sarnaselt nagu käsk `sass --watch . .`, mis tähendab, et lähtekoodi kompileerimiseks ei ole vaja teha täiendavat tööd. Piisab sellest kui määrata töö alguses kaust, kus asub Sass'i lähtekoodi ning kaust, kuhu kompileeritakse tulemusfailid (vt ekraanipilt 2.4). Peale kaustade määramist võib jätta Scout'i taustale jooksma (vt joonis 2.5). (Mutually Human)



Ekraanipilt 2.4. Rakenduse *Scout* seadistamine.



Ekraanipilt 2.5. *Scout*'i teated kompileerimisel.

2.9 Sass'i kasutamise varjuküljed

Liigne HTML-struktuuri järgimine pesastamisel

Pesastamise kasutamisel tuleks püüda vältida lehe HTML-struktuuri järgimist. Võimalusel peaks hoidma kinni reeglist, mille järgi maksimaalne DOM'i elementide arv selektorina on neli. Seega selektor `body div.container article p` on juba natuke liiga pikk. Liiga pikad selektorid on mitmes mõttes halvad. Esiteks muudavad need faili mahu suureks, mis omakorda teeb aeglasemaks lehe laadimise. Teiseks probleemiks on see, et kui peaks olema tarvis muuta HTML'i struktuuri, siis võib olemasolev CSS lakata töötamast uue struktuuri saanud lehega.

Peale Sass'i kompileerimist tasuks alati pilk heita tulemusfailile, mis aitab selliseid vigu vältida. Võimaluse korral tuleks pikki selektoreid vältida kasutadades lehe HTML-koodi sees elementide juures ID'sid ning märgist *class*, mis võimaldaks elemendi poole pöörduda otse. (Ricalde, 2011)

Programmeerimine

CSS on oma olemuselt väga lihtne ning ei nõua selle kasutajalt programmeerimisalaseid teadmisi. Sass teeb selles osas aga märgatavaid korrektuure. Muutujad, funktsioonid, loogikaavaldised, sisseehitatud funktsioonid erinevate andmetüüpidega töötamiseks ja teised Sass'i poolt lisatavad laiendused nõuavad kasutajalt elementaarseid teadmisi programmeerimisest. Sellest tingituna nõuab keerulisemate probleemide lahendamine Sass'i abil CSS'iga võrreldes märgatavalt rohkem teadmisi. Samuti võib kuluda küllaltki palju aega selleks, et harjuda ära uue süntaksi ning Sass'i poolt pakutavate täiendustega kui kasutajal puudub kokkupuude programmeerimisega.

Võimalik väljundkoodi liigne mahukus

Kui kirjutada koodi Sass'is, siis võib esialgne lähtekood näida küllaltki lühike ning optimeeritud. Peale kompileerimist tuleks kindlasti heita pilk genereeritud CSS'ile veendumaks, et peale kompileerimist ei ole tekkinud põhjendamatult palju lisakoodi. Olles Sass'i võimalusi valesti kasutanud võib tulemusfaili maht olla muutunud lausa mitme suurusjärgu võrra, millest tingituna kannatab lehe või veebirakenduse laadimise kiirus.

Rohkem infot selle kohta, kuidas peaks lõplikus CSS'is selektorid olema kirjutatud saab siit: https://developers.google.com/speed/docs/best-practices/rendering?hl=et - UseEfficientCSSSelectors

3 CSS'i laiendusvõimalused läbi Sass'i

3.1 Muutujad

Tihtipeale tuleb ette, et projektis on mingisugused suurused (elementide laiused, kõrgused), värvid või muud elemendid, mis on samad läbi terve projekti. Kindlasti mitte harvem ei ole juhus, kus tellija soovib juba valmimisjärgus veebilehel või muul HTML'is kirjutatud dokumendis muuta mõnda värvi, elemendi laiust või muud parameetrit, mis on sama üle kõigi kujundusfailide. Selliste muudatuste tegemine võib aga osutuda aeganõudvaks kuna deklaratsioonide väärtusi on vaja muuta väga paljudel erinevatel ridadel. Sellisel juhul ei jää üldiselt muud üle kui tuleb muuta kasutades otsingut ja selle muutmise funktsiooni. See on aga küllaltki tüütu ning aeganõudev kui tegu on natuke suurema projektiga.

Teiseks probleemiks, mis võib tavalist CSS'i kirjutades ette tulla, on see, kui läbi terve projekti on kujunduses kasutusel kindel värv, mis kirjutatakse kujul #D699D6. Üldjuhul selline kirjutusviis ei ole inimesele hästi meeldejääv ning seetõttu peab ta hakkama seda eraldi kusagilt otsima. Sellest tulenevalt on tegemist jälle ajakaoga.

Selleks, et olukorda parandada on Sass'is kasutusele võetud muutujad. Muutuja deklareerimine näeb välja kui tavalise CSS deklaratsiooni kirjutamine. Vahe on selles, et omaduse asemele kirjutatakse muutuja nimi. Muutuja deklareerimiseks kasutab Sass, sarnaselt levinud skriptimiskeelele PHP, dollarimärki ($).

```
$width: 1000px;
$warning-color: #ff0000;
```

Koodinäide 3.1. Muutujate deklareerimine.

Muutuja skoop

Muutujaid on Sass'is kahte tüüpi. Esimesed on globaalsed (*global*) ehk need, mida saab kasutada terve faili ulatuses. Sellised muutujad defineeritakse alati eraldi real (vt. Koodinäide 3.2).

Tavaliselt defineeritakse kõik globaalsed muutujad kohe faili alguses või isegi eraldi failis juhul kui tegemist on suurema projektiga.

Teine grupp muutujaid on piiratud nähtavusega neid saab kasutada selektori sees ning tema sees olevate selektorite sees (vt. Koodinäide 3.2).
(Catlin & Catlin, Pragmatic Guide to Sass, 2011)

```
$width: 1000px;

header {
    $content: 800px;
    width: $width;
    div {
        width: $content;
    }
}

div {
    // width: $content; ei saa kasutada
}
```

Koodinäide 3.2. Globaalsed ja lokaalsed muutujad.

Eelnevas koodinäites on `$width` globaalne ning `$content` lokaalne muutuja. Juhul kui selektori "div" juures oleks muutujat `$content` kasutatud omadusele "width" väärtuse andmiseks, siis oleks Sass'i kompilaator selle peale veateate andnud.

Muutujaid on Sass'is viis tükki:

- numbrid: 2, 4.5, 10%, 20px;
- stringid: underline, dotted, none, left;
- värvid: #000000, rgba(255, 255, 255, 0.5);
- tõeväärtusmuutujad: true, false;
- listid: 5px 10px 5px, Arial, Helvetica, Courier.

Muutujate interpolatsioon

Muutujaid saab kasutada lisaks tavalisele omaduste väärtustamisele ka selleks, et moodustada nende abil omaduste või selektorite nimesid.

```
$side: top;
$radius: 10px;

.rounded-#{$side} {
  border-#{$side}-radius: $radius;
  -moz-border-radius-#{$side}: $radius;
  -webkit-border-#{$side}-radius: $radius;
}
```

Koodinäide 3.3. Muutujate interpolatsioon.

(Catlin, Weizenbaum, & Eppstein, Sass)

3.2 Matemaatilised avaldised

Elementide laiuse, kõrguse või lehel paiknemise määramiseks teeb CSS'i kirjutav disainer või arendaja peast ja vajadusel kalkulaatoriga arvutusi. See toob kaasa arendamiseks kuluva aja pikenemise ning võimalike vigade arvu suurenemise. Sass lisab võimaluse kasutada matemaatilisi avaldisi nii erinevate CSS'is olemasolevate suuruste, string'i tüüpi väärtuste kui ka värvide vahel. Teostatavad on kõik levinumad matemaatiilised tehted nagu liitmine, lahutamine, korrutamine kui ka jagamine. (Catlin & Catlin, Pragmatic Guide to Sass, 2011)

```
$border: 2px;
$width: 800px;
$fontSize: 4px;

div {
    width: $width - $border * 2;
    height: (800px/2);
    padding: $border;
    border: 1px solid black;
    font-size: $fontSize + 5px;
    color: #222222 + #333333;
    background-color: rgba(255, 0, 0, 0.75) + rgba(0, 255, 0, 0.75);
    margin: 5px + 5px auto 5px + 5px auto;
}
```

```
p:before {
  content: "Sass is superset of CSS#{1+2} syntax!";
}
```

Koodinäide 3.4. Matemaatiliste avaldiste kasutamine.

Eelneva Sass'i tulemuseks on järgnev CSS:

```
div {
  width: 796px;
  height: 400px;
  padding: 2px;
  border: 1px solid black;
  font-size: 7.75pt;
  color: #555555;
  background-color: rgba(255, 255, 0, 0.75);
  margin: 10px auto 10px auto;
}

p:before {
  content: "Sass is superset of CSS3 syntax!";
}
```

Koodinäide 3.5. Matemaatiliste avaldiste tulemus peale kompileerimist.

3.3 Pesastamine (Nesting)

Teine peamine lisafunktsionaalsus on pesastamine (*Nesting*). Need, kes on suuremate projektide juures kirjutanud CSS'i teavad, et tihtipeale tekivad lehel paiknevate korduvate elementide kujundamisel pikad read, mille kirjutamine on küllaltki tüütu. Samuti aitab pesastamine kaasa koodi paremale organiseeritusele kuna hoiab sarnaste DOM objektidega seotud koodi kenasti ühes kohas koos. (Catlin, Weizenbaum, & Eppstein, Sass)

Vaatame näitena alljärgnevat koodi:

```
#main article { border-bottom: 1px #CCCCCC dotted; }
#main article p { color: #333333; }
```

```
#main article p a { text-decoration: underline; }
#main article p a:hover { text-decoration: none; }
#main article p figure { border: 2px #F1F1F1 solid; }
#main article p strong { font-weight: 800; }
```

Koodinäide 3.6. CSS'i astmelised selektorid koos deklaratsioonidega.

Selle sama saaks Sass'i abil kirjeldada palju mugavamalt ning kiiremini. Kood näeks välja järgmine:

```
#main {
  article {
    border-bottom: 1px #CCC dotted;
    p {
      color: #333333;
      a {
        text-decoration: underline;
        &:hover {
          text-decoration: none;
        }
      }
      figure {
        border: 2px #F1F1F1 solid;
      }
      strong {
        font-weight: 800;
      }
    }
  }
}
```

Koodinäide 3.7. Pesastamine.

Väljastatav tulemus on järgmine:

```
#main article {
  border-bottom: 1px #CCC dotted; }
  #main article p {
    color: #333333; }
```

```
#main article p a {
  text-decoration: underline; }
  #main article p a:hover {
    text-decoration: none; }
#main article p figure {
  border: 2px #F1F1F1 solid; }
#main article p strong  {
  font-weight: 800; }
```

Koodinäide 3.8. Pesastamist sisaldava lähtekoodi tulemus peale kompileerimist.

Peale selektorite saab pesastamist kasutada ka omaduste juures.

```
h1 {
    text: {
        decoration: underline;
        transform: capitalize;
    }
}
```

Koodinäide 3.9. Pesastamise kasutamine omaduste juures.

Kompileerimisel saadav tulemus on järgmine:

```
h1 {
  text-decoration: underline;
  text-transform: capitalize; }
```

Koodinäide 3.10. Pesastamise kasutamine omaduste juures - kompileerimisel saadud tulemus.

3.4 @mixin direktiiv

Mixin on veel üks viis, mis vähendab korduvate koodiridade kirjutamist. Mixin on osa Sass'is kirjutatud koodist, mida saab rakendada teisele selektorile. Nendele, kes on varem kokku puutunud programmeerimisega, paistavad mixin'id nagu kasutaja poolt defineeritud funktsioonid PHP's, JavaScript'is, Ruby's või mõnes teises analoogses keeles. Ainuke erinevus seisneb selles, et mixin ei tagasta kunagi väärtuseid vaid väljastab selle asemel koodi.

Selleks, et defineerida mixin, tuleb kirjutada `@mixin`, mille järgi kirjutatakse tema nimi ning seejärel tema kujunduse defineeriv kood.

Mixin'i kasutamiseks tuleb kirjutada `@include [mixini-nimi]`. Näiteks `@include kollane-vaike`.

CSS3 tõi ühena uuendustest võimaluse luua kumeraid nurkasid. Kuna kumerate nurkade spetsifikatsioon ei ole veel täielikult valmis, siis kasutavad erinevad veebilehitsejate tootjad eesliiteid. Ehk siis selleks, et saaksime ümarate servadega kasti, mis töötaks nii Firefoxi, Internet Explorer'I kui ka Google Chrome'I peal, tuleb CSS'is deklareedida ümarad servad iga veebilehitseja jaoks eraldi. See on aga ideaalne koht, kus kasutada mixin'it.

```
@mixin rounded-corners {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  -ms-border-radius: 5px;
  -o-border-radius: 5px;
  border-radius: 5px;
}

article {
   @include rounded-corners;
}
```

Koodinäide 3.11. Mixin'i deklareerimine ning selle kasutamine.

Kompileeritud kujul on tulemuseks.

```
article {
      -webkit-border-radius: 5px;
      -moz-border-radius: 5px;
      -ms-border-radius: 5px;
      -o-border-radius: 5px;
      border-radius: 5px; }
```

Koodinäide 3.12. Peale kompileerimist saadud tulemus.

Eelnevalt kirjeldatud koodinäidetes on mixin'i sees olevate deklaratsioonide väärtused staatilised. See tähendab, et ükskõik, kus me kasutame mixinit `rounded-corners`, saame tulemuseks elemendi, mille kõik neli serva on ümarad raadiusega 5 pikslit.

Sellega mixin'i võimalused veel ei piirdu. Näiteks ei pruugi kõikide ümarate servade raadiuseks olla 5 pikslit. saab tegelikult aga veel laiendada määrates mixini definitsiooni juures ära argumendid. Argumentidele võib anda ka vaikimisi väärtuse.

```
@mixin rounded-corners($radius: 10px) {
      -webkit-border-radius: $radius;
      -moz-border-radius: $radius;
      -ms-border-radius: $radius;
      -o-border-radius: $radius;
      border-radius: $radius;
}

nav {
      @include rounded-corners;
}

article {
      @include rounded-corners(20px);
}
```

Koodinäide 3.13. Mixin'i defineerimine koos argumendiga ja selle kasutamine.

Kompileerimisel saadav tulemus on järgmine:

```
nav {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  -o-border-radius: 10px;
  border-radius: 10px; }
```

```
article {
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  -ms-border-radius: 20px;
  -o-border-radius: 20px;
  border-radius: 20px; }
```

Koodinäide 3.14. Mixin'i koos argumendiga kompileerimisel saadud tulemus.

3.5 Pärilus (Selector Inheritance)

Pärilust peetakse üheks kõige võimsamaks ning kasulikumaks lisaks, mida Sass pakub tavalise CSS'i täienduseks. Päriluse idee on oma tee Sass'i leidnud läbi OOCSS vaate. Sass'i poolt on seda ideed küll natuke laiendatud ning lisaks klassidele saab selektor pärida deklaratsioone ka teistelt selektori tüüpidelt, milleks on DOM'i element ise, klass või identifikaator (ID). (Eppstein, 2011)

Rääkides sellest, miks selektorite pärilus on kasulik, siis esimese põhjusena on selleks fakt, et see teeb märgatavalt lihtsamaks nii HTML-i kui ka kujundusfailide haldamise. Tänu sellele võidab arendaja jällegi aja kokkuhoius.

Selleks, et näidata, milliselt selektorilt tema omadused pärida on Sass'is kasutusel `@extend` direktiiv.

Et täpsemalt päriluse olemusest aru saada, vaatame lihtsat koodinäidet.

```
.message {
    border: 1px solid #ccc;
}

.important-message {
    @extend .message;
    color: #ff0000;
}
```

Koodinäide 3.15. Pärilus.

Peale kompileerimist näeb saadav tulemus välja järgmine:

```
.message, .important-message {
  border: 1px solid #ccc;
}

.important-message {
  color: #ff0000;
  font-weight: bold;
}
```

Koodinäide 3.16. Peale kompileerimist saadav tulemus.

Nagu tulemusest näeme, on kõik omadused, mis on defineeritud selektori .message puhul automaatselt rakendatud ka selektori .important-message puhul. Omaduste pärimine ei ole piiratud ainult ühe selektoriga. Soovides pärida omadusi mitmelt selektorilt tuleb need eraldada üksteisest komadega. Näiteks: `@extend .message, .warning`.

Lisaks saab päriluse kohta lugeda järgnevast Chris Eppsteini artiklist:
http://chriseppstein.github.com/blog/2009/10/12/css-class-inheritance/

3.6 @import direktiiv

Kuna suuremate projektide juures võivad CSS'is kirjeldatud kujundust sisaldavad failid venida väga pikaks, siis eksisteerib selleks CSS'is direktiiv `@import`, mis võimaldab ühel HTML-lehel kasutada koos mitut erinevat stiililehte. Suuremate projektide stiilifailide halduse teeb see lihtsaks kuid selline importimine toob kaasa märgatava kaotuse lehe laadimise kiiruses kuna iga uue imporditava stiililehe laadimiseks tuleb teha eraldi päring.

Erinevalt tavalisest CSS'is kasutatavast `@import` direktiivist tõmbab Sass kõik välised failid kokku ühte faili, ning lõpuks seotakse lehega ikkagi ainult üks CSS fail. Seega toimub importimine kompileerimise ajal mitte kliendi poolel. Lisaks sellele on kõik *mixin*'id ning muutujad, mis on defineeritud imporditavates failides kättesaadaval failis,

millesse nad imporditakse. Siinkohal võib tõmmata paralleeli PHP skriptimiskeeles kasutatava `import()` funktsiooniga.

Sass'is imporditavad failid sisaldavad üldjuhul nime esimese sümbolina alakriipsu "_" ning neid kutsutakse nimega *partial* (eesti. k. osa). Näiteks võime koodi parema organiseerituse eesmärgil luua ühe faili, kus on kirjeldatud ära kõik projektis kasutatavad muutujad (vt koodinäide 3.17). Peale seda võib loodud faili importida teise (vt koodinäide 3.18).

Selleks, et importida ühte faili teise, tarvitseb kirjutada `@import "variables";`.
(Catlin, Weizenbaum, & Eppstein, Sass)

Juhul kui jätta imporditava faili nimest ära alakriips "_", siis loob Sass lisaks veel ka tavalise CSS faili. (Catlin & Catlin, Pragmatic Guide to Sass, 2011)

Näites loodakse fail "_variables.scss", kuhu salvestatakse kõik projektis kasutatavad muutujad.

```
$titleColor: #333;
$backgroundColor: #f1f1f1;
```

Koodinäide 3.17. Muutujaid sisaldav fail "_variables.scss".

Peale muutujaid sisaldava faili loodakse fail, kuhu sisse imporditakse deklareeritud muutujatega fail. Alljärgnev koodinäide näitab, kuidas toimub importimine ning

```
@import "variables";

body {
  background-color: $backgroundColor;
}

h1 {
  color: $titleColor;
```

```
}
```

Koodinäide 3.18. Importimine faili.

Peale koodinäidete 3.17 ja 3.18 on tulemuseks järgmine CSS kood:

```
body {
  background-color: #f1f1f1; }

h1 {
  color: #333333; }
```

Koodinäide 3.19. Imporditud ja kompileeritud Sass faili tulemus.

3.7 Tingimuslaused

SassScript'is on kasutusel mitmesugused võimalused võimaldamaks luua paindlikumaid funktsioone. Üldjuhul neid vaja ei lähe kuid kui on plaanis luua mingisugune oma enda väike teek (*library*), siis see on võimalik just tänu nendele avaldistele, mis võimaldavad kirjutada paindlikumat koodi. Enamasti leiavad loogikaavaldised kasutamist `@mixin` ning `@function` direktiivis selleks, et kirjutada koodijuppe, mis oleksid võimalikud laialdaselt kasutatavad erinevate projektide juures. (Catlin & Catlin, Pragmatic Guide to Sass, 2011)

3.8 Sisseehitatud funktsioonid

Sass võimaldab kasutada sisemisi funktsioone, mis on loodud toimetamiseks erinevate andmetüüpidega. Enamus sisseehitatud funktsioone on loodud värvidega toimetamiseks. Järgnevalt heidetakse pilk funktsioonidele, mida kõige enam võiks kujundamisel vaja minna. Iga koodinäite juurde on lisatud pilt, visualiseerimaks saadavat tulemust.

Kellel on soovi teha lähemalt tutvust SassScript'i funktsioonidega, võib tutvuda järgmise lehega: http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html

Värvide muutmine heledamaks ja tumedamaks

Küllaltki tihti tuleb ette, et kujunduses on vaja kasutada ühe põhivärvi erinevaid variatsioone. Tavaliselt pöördutakse sellisel juhul mõne värvitabeli poole näiteks nagu siin: http://www.w3schools.com/html/html_colors.asp.

Sass võimaldab jätta selle käigu tegemata ning pakub värvitooni heledamaks tegemiseks välja funktsiooni `lighten($color, 20%)`. Funktsioon võtab esimese argumendina sisse baasvärvi, mille põhjal värvi heledamaks muudetakse. Teise argumendina antakse ette protsent kui palju antud värvi muuta.

Selleks, et teha värvi tumedamaks, on kasutusel sarnane funktsioon `darken(#ff0000, 20%)`, mille argumendid on analoogsed funktsioonile `lighten`. (Catlin, Eppstein, & Weizenbaum, Module: Sass::Script::Functions, 2012)

```
$baseColor: blue;

#lighter10 { background-color: lighten($baseColor, 10%); }
#lighter20 { background-color: lighten($baseColor, 20%); }
#lighter30 { background-color: lighten($baseColor, 30%); }
#lighter40 { background-color: lighten($baseColor, 40%); }
#base { background-color: $baseColor; }
#darker10 { background-color: darken($baseColor, 10%); }
#darker20 { background-color: darken($baseColor, 20%); }
#darker30 { background-color: darken($baseColor, 30%); }
#darker40 { background-color: darken($baseColor, 40%); }
```

Koodinäide 3.20. värvifunktsioonid lighten() ja darken()

Kompileerimisel saadud kood on järgmine:

```
#lighter10 {
  background-color: #3333ff; }
#lighter20 {
  background-color: #6666ff; }
#lighter30 {
  background-color: #9999ff; }
#lighter40 {
  background-color: #ccccff; }
#base {
```

```
  background-color: blue; }
#darker10 {
  background-color: #0000cc; }
#darker20 {
  background-color: #000099; }
#darker30 {
  background-color: #000066; }
#darker40 {
  background-color: #000033; }
```

Koodinäide 3.21. Värvifunktsioonide lighten() ja darken() tulemus CSS'ina.

Eelneva koodi tulemuseks on alljärgnev tabel:

Ekraanipilt 3.1. Funktsiooni "lighten" kasutamisel saadav tulemus.

Läbipaistvus

Funktsioone, mis võimaldab teha värve läbipaistvaks on Sass'is mitmeid. Üheks kõige levinumaks neist on `rgba(blue, .2)`. Sarnaselt eelenevalt vaadeldud funktsioonidele antakse ka siin ette esimesena värv ning seejärel arv, mille võrra suurendada läbipaistvust.

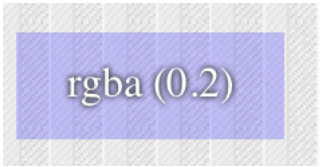
```
$baseColor: blue;
#rgba { background-color: rgba($baseColor, .2); }
```

Koodinäide 3.22. Värvifunktsiooni rgba() kasutamine.

Kompileerimisel saadav tulemus:

```
#rgba {
  background-color: rgba(0, 0, 255, 0.2); }
```

Koodinäide 3.23. Funktsiooni rgba() väljund CSS'ina.



Ekraanipilt 3.2. Funktsiooni "rgba()" kasutamisel saadav tulemus.

3.9 Lähtekoodi kommenteerimine

Kommentaare on Sass'is kahte tüüpi. Esimene tüüp on samad kommentaarid mida kasutab klassikaline CSS. Kõik, mis jääb märkide `/*` ja `*/` vahele on kommentaar. Selliselt kirjutatud kommentaarid võivad paikneda mitmel real.

```
/* This is a typical multiline comment in CSS and Sass */
```

Koodinäide 3.24. Nii Sass'is kui ka CSS'is kasutatav kommentaar.

Selliselt kirjutatud kommentaarid ignoreeritakse Sass'i poolt ning täpselt samasugune tulemus väljastatakse ka kompileeritud CSS'i faili.

Teine kommentaari tüüp on kasutusel ainult Sass'i failides ning peale kompileerimist selliselt kirjutatud kommentaare CSS failis näha ei ole. Sellised kommentaarid on ainult üherealised. Kui tahta paigutada kommentaari üle mitme rea, siis peab iga järgmine rida algama samuti märkidega `//`.

```
// This is how we define comment in Sass which will not be show in
compiled CSS
```

Koodinäide 3.25. Ainult Sass'is kasutatav kommentaar.

(Catlin & Catlin, Pragmatic Guide to Sass, 2011)

4 Näidislehestik

4.1 Eesmärk

Näidislehestiku eesmärgiks on näidata kõikide töös käsitletud Sass'i võimaluste kasutamist praktikas. Samuti püütakse näidislehestiku abil näidata, kuidas valmib väike teek (library) koodijuppidest, mille saab võtta kasutusele iga uue projekti juures.

Ekraanipilt 4.1. Loodava lehe näidis.

4.2 Lehe struktuur

Loodava näidislehe failide ja kaustade struktuur on järgmine:

- **naidislehestik/**
  - index.htm
  - forms.htm
  - **css/**
    - main.css
  - **images/**
    - stripes.png
    - sass_logo.jpeg
  - **sass/**
    - _css3.scss
    - _forms.scss
    - _variables.scss
    - main.scss

Projekti kaustas asuvad failid "index.htm" ja "forms.htm" on näidislehestiku ainukesed HTML-dokumendid. Kaustas "css" on Sass'i lähtekoodi sisaldavate failide kompileerimisel saadud tavaline CSS-fail, milles on koos kogu lehe kujunduse määravad reeglid. Kaust "sass" sisaldab Sass'i lähtekoodifaile. Järgnevalt on ära toodud kõik lähtekoodifailid lühikeste kirjeldustega, mis konkreetses failis asub.

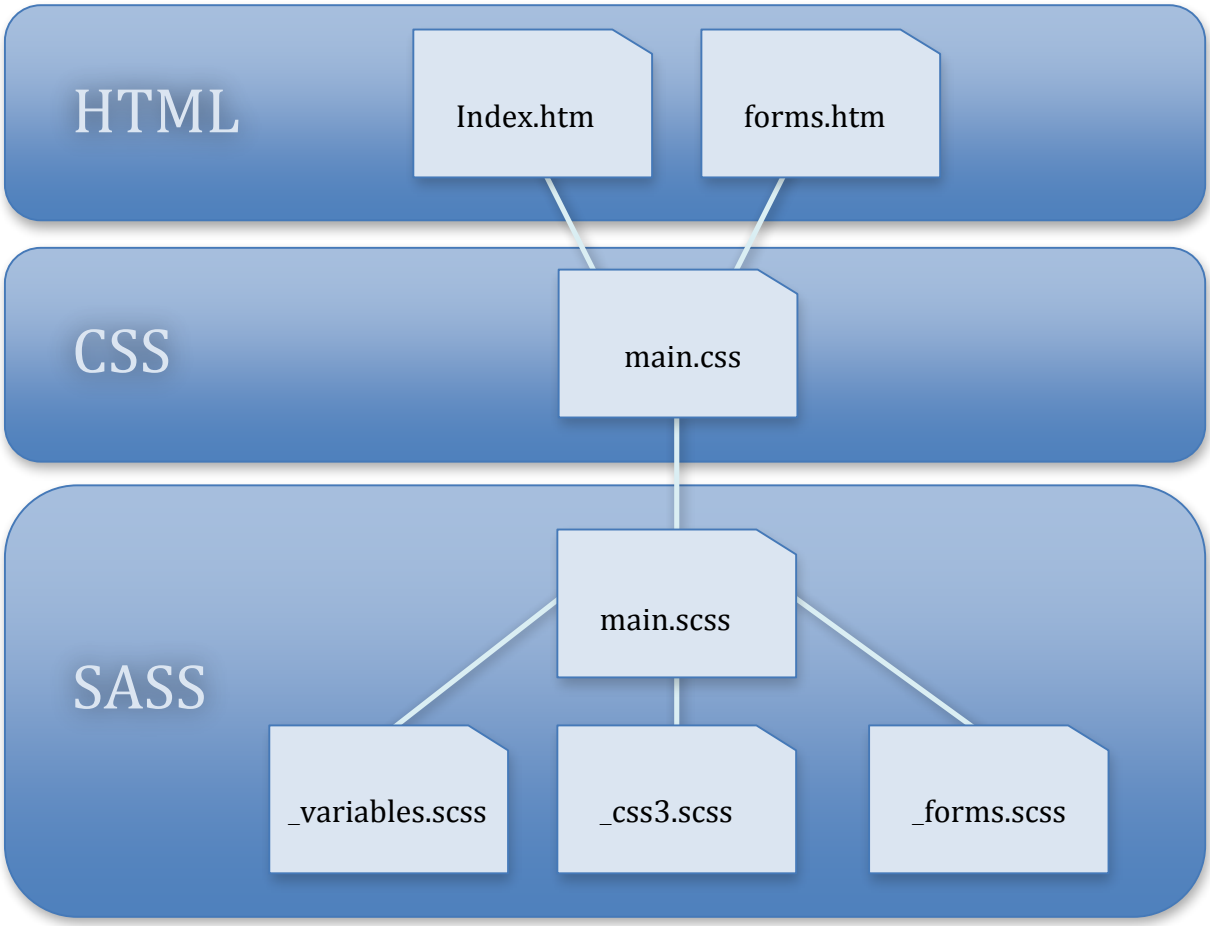
_css3.scss – sisaldab eranditult defineeritud *mixin*-direktiive, mis tegelevad CSS3-ga lisandunud uute võimalustega.

_forms.scss – sisaldab kõikide lehestikus olevate vormide kujunduse deklaratsioone.

_variables.scss – fail, milles on defineeritud kõik näidislehestikus kasutatavad muutujad.

main.scss – lehestiku kõige tähtsam fail, kuhu on imporditud kõik teised Sass'i failid. Samuti on selles failis deklareeritud kõik ülejäänud CSS reeglid, mis on vajalikud lehe kujundamiseks. See on ainuke fail, millest luuakse väljundfail kausta "css".

Loogilisel kujul iseloomustab loodavat lehestikku järgmine skeem:



Joonis 4.1. Näidislehestiku loogiline struktuur

Eelnevalt jooniselt on näha, et failid _variables.scss, _css3.scss ja _forms.scss on *partial*'id, mis imporditakse (vt peatükk 3.6) peafaili "main.scss". Seejärel luuakse Sass'is kirjutatud lähtekoodi sisaldavate failide põhjal tavaline CSS-fail (vt lisa 3), "main.css", mis omakorda on seotud näidislehestikus olevate HTML-failidega.

4.3 Teostus

***Partial* "_css3.scss"**

Failis "_css3.scss" on loodud hulk *mixin*-direktiive (vt peatükk 3.4), mis lihtsustavad projektis CSS3-ega lisandunud võimaluste kasutamist.

Lisaks sellele on autor püüdnud koondada siia *partial*'isse kõik koodijupid, mis on iseseisvad ning kasutatavad teistes projektides. Koodi taaskasutus on üks peamiseid viise, mis aitab vähendada töömahtu ning samuti lühendada projekti valmimiseks kuluvat aega. Oma väikese SassScript'i teegi (library) loomine on väga hea võimalus.

Järgnevalt on ära kirjeldatud kõik failis "_css3.scss" olevad mixin'id autoripoolsete kommentaaridega.

```
@mixin text3d($color) {
  color: $color;
  text-shadow: 1px 1px 0px darken($color, 5%),
               2px 2px 0px darken($color, 10%);
}
```

Koodinäide 4.1. @mixin text3D

Mixin'i ülesandeks on immiteerida 3D-teksti kasutades selleks omadust "text-shadow", mis võimaldab tekstile tekitada erinevat tüüpi varje.

```
@mixin rounded($radius: 10) {
    -webkit-border-radius: #{$radius}px;
    -moz-border-radius: #{$radius}px;
    -ms-border-radius: #{$radius}px;
    -o-border-radius: #{$radius}px;
    border-radius: #{$radius}px;
}
```

Koodinäide 4.2. @mixin rounded

Mixin, mille ülesandeks on lihtsustada ümarate nurkade loomist võimaldades hoida arendaja aega kokku kuna ümarate servadega elemendi saamiseks ei ole vaja kirjutada sisuliselt sama deklaratsiooni mitu korda. *Mixin* võtab argumendina sisse raadiuse (`$radius`) ning vastavalt sellele väljastab kasutamisel deklaratsioonid ümarate nurkade saamiseks erinevatele veebilehitsejatele.

```
@mixin rounded-special($topLeft, $topRight, $bottomRight,
$bottomLeft) {
    border-radius: #{$topLeft}px #{$topRight}px #{$bottomRight}px
#{$bottomLeft}px;
    -moz-border-radius: #{$topLeft}px #{$topRight}px
```

```
#{$bottomRight}px #{$bottomLeft}px;
    -webkit-border-radius: #{$topLeft}px #{$topRight}px
#{$bottomRight}px #{$bottomLeft}px;
}
```

Koodinäide 4.3. @mixin rounded-special

Sarnaselt *mixin*'ile "rounded" (vt koodinäide 4.2) võimaldab ka see luua ümaraid nurki. Erinevuseks on, et sellel *mixin*'il on võimalik argumentidena ette anda iga nurga jaoks eraldi raadiused.

```
@mixin columns($count, $gap, $rule: null, $alignment: default) {
    @include column-properties($count, $gap);

    @if $rule != null {
        -moz-column-rule: $rule;
        -webkit-column-rule: $rule;
        column-rule: $rule;
    }
    @if $alignment != default {
        text-align: $alignment;
    }

    @include respond-to(medium-screens) {
        @include column-properties($count - 1, $gap);
    }
    @include respond-to(handhelds) {
        @include column-properties($count - 2, $gap);
    }
}
```

Koodinäide 4.4. @mixin columns

Üks uuendus, mis CSS3'ega kaasa tuli, oli mitmeveeruline asetus (multi-column layout). Kuna see moodul ei ole muutunud W3C poolt ametlikult soovituseks, siis tuleb kasutada erinevate veebilehitsejate jaoks eraldi prefiksit. Selleks, et ei peaks iga kord kui on soov kasutada mitmeveerulist asetust kirjutama välja tasub ka siinkohal kasutada *mixin*-direktiivi. *Mixin* (vt koodinäide 4.4) võtab sisse neli argumenti, millest kaks esimest on kohustuslikud ning kaks viimast valikulised. Kohustuslikud argumendid on veergude arv (`$count`) ja veergude vaheline kaugus üksteisest (`$gap`). Valikulised on veergude

eraldaja (`$rule`) ning veerudena kuvatava teksti paigutus (`$alignment`), milleks vaikimisi kasutatakse vasakule joondust.

```
@mixin column-properties($count, $gap) {
    -moz-column-count: $count;
    -webkit-column-count: $count;
    column-count: $count;

    -moz-column-gap: #{$gap}px;
    -webkit-column-gap: #{$gap}px;
    column-gap: #{$gap}px;
}
```

Koodinäide 4.5. @mixin column-properties

Mixin, mis on mõeldud kasutamiseks ainult *mixin*'i "columns" sees (vt koodinäide 4.4) selleks, et mitte korrata juba kirjutatud koodi.

```
@mixin respond-to($media) {
    @if $media == handhelds {
        @media only screen and (max-width: 480px) {
            @content;
        }
    }
    @else if $media == medium-screens {
        @media only screen and (min-width: 481px) and (max-width:
1023px) {
            @content;
        }
    }
    @else if $media == wide-screens {
        @media only screen and (min-width: 1024px) {
            @content;
        }
    }
}
```

Koodinäide 4.6. @mixin respond-to

CSS3-e tulekuga laienesid võimalused kasutada meediapäringuid. Meediapäringud võimaldavad kasutada erinevaid stiile vastavalt sellele, millise ekraanilaiuse või seadmega lehekülge vaadatakse. Rohkem infot meediapäringute kohta saab soovi korral siit: http://webdesignerwall.com/tutorials/css3-media-queries

Koodinäites 4.6 on *mixin*, mille eesmärgiks on lihtsustada meediapäringute kasutamist.

Seda, kuidas näidislehestik toimib erinevate ekraanisuurustega saab testida siin: http://responsive.is/www.tlu.ee/~strin/seminaritoo/naidislehestik

Partial "_form.scss"

Partial "_form.scss" sisaldab eranditult failis "forms.htm" (vt lisa 2) kirjeldatud vormielementide paigutust ning kujundust.

```
#contact {
    p {
        border-bottom: $fieldSepStyle;
        &:last-child {
            border: 0;
        }
        clear: left;
    }

    label {
        display: block;
        font-weight: bold;
        text-align: right;
        width: $labelWidth;
        float:left;
    }

    .small {
        color: #666666;
        display: block;
        font-size: 11px;
        font-weight: normal;
        text-align: right;
        width: $labelWidth;
    }

    input {
        float: left;
        padding: $inputPad;
        border: $inputBorderStyle;
        margin: $inputMargin;
```

```
            width: $inputWidth;
            @include rounded(5);
            background: {
                color: lighten($mainColGreen, 30%);
            }
            @include respond-to(medium-screens) {
                width: 260px;
            }
            @include respond-to(handhelds) {
                width: 150px;
            }
        }

        textarea {
            @extend input;
            height: $textareaHeight;
        }

        button {
            margin-left: $labelWidth;
        }
}
```

Koodinäide 4.7. Fail "_forms.scss" sisu.

Vormide juures on kasutatud pesastamist (vt peatükk 3.3) eesmärgiga luua võimalikult täpne selektor. Tekstiväljade puhul on kasutatud *mixin*'it "rounded" ümarate nurkade loomiseks ning *mixin*'it "respond-to" selleks, et sobitada kujundust erinevate ekraanisuurustega. Tekstivälja (*textarea*) puhul on rakendatud *extend*-direktiivi, mis tähendab seda, et kõik omadused võetakse selektorilt "input". Tekstivälja kõrguse väärtus on ainukesena erinev.

Partial "_variables.scss"

Failis "_variables.scss" on deklareeritud kõik näidislehestikus kasutatavad muutujad (vt. ka peatükk 3.1). Hoides muutujaid ühes failis teeb see väga lihtsaks lehekülje kujunduse elementide ning värvide muutmise.

Võib juhtuda, et peale kujunduse valmimist tekib vajadus muuta kogu lehte kas laiemaks või kitsamaks. Tänu kasutatud muutujatele ning matemaatilistele arvutustele failis

“main.scss”, pole vaja soovitud tulemuse saavutamiseks teha muud kui anda muutujale `$contWidth` teine väärtus.

Niisamuti käib lehekülje värvide, taustapildi ning HTML-struktuuris olevate elementide paigutuse muutmine osaliselt läbi muutujate.

Alljärgnevalt on ära toodud kogu näidislehestiku loomisel kasutatud muutjate fail:

```
// üldised lehe muutujad
$contBoxCol: #ffffff;
$mainColGreen: #A8CF34;
$linkCol: $mainColGreen;
$textCol: #333;
$pageTitleCol: #b738b8;
$bgImage: "../images/stripes.png";
$articleSepLine: 1px dotted #f1f1f1;
$siteMargin: 40px;

// teksti suurused
$normalTextSize: 14px;
$headerSize: 38px;
$lineHeight: 22px;
$fonts: 'Trebuchet MS', Helvetica, sans-serif;

// sisukast
$contWidth: 750px;
$contPad: 30px;
$contElemSep: 10px;

// külgriba
$asideWidth: 250px;
$asideGap: 10px;
$asidePad: 10px;
$asideBgCol: lighten($mainColGreen, 20%);
$articleSepStyle: 1px solid darken($mainColGreen, 20%);
```

```
// leht "vormid"
$fieldSepStyle: dotted 1px $mainColGreen;
$labelWidth: 140px;

// teksti sisestusväljad
$inputWidth: $contWidth - $labelWidth - $contPad * 4;
$inputMargin: 5px 10px 5px 10px;
$inputPad: 10px 4px;
$inputBorderStyle: solid 1px darken($mainColGreen, 20%);
$textareaHeight: 80px;
```

Koodinäide 4.8. Fail "_variables.scss"

main.scss

Fail main.scss on peamine fail, milles on deklareeritud üldine lehekülje kujundus.

```
@import "variables";
@import "css3";
@import "forms";
```

Koodinäide 4.9. *Partial*'ite importimine põhifaili.

Eelnevas koodinäites imporditakse kõik näidislehestikus olevad *partial*'id selleks, et kasutada nendes defineeritud muutujaid ja *mixin*-direktiive.

```
body {
    background-image: url(#{$bgImage});
    margin-top: $siteMargin;
    font-family: $fonts;
    line-height: $lineHeight;
    font-size: $normalTextSize;
    color: $textCol;
}
```

Koodinäide 4.10. Lehe üldise ilme kujundamine.

Selektori "body" sees defineeritakse hulk üldiseid leheküljega seotud omadusi (taustapilt, kaugus ülemisest servast, kirjastiilid, rea kõrgus, kirja suurus ning värv). Kõik omistatavad väärtused pärinevad failist "_variables.scss" (vt koodinäide 4.8).

```
h1 {
    font-size: $headerSize;
    @include text3d($pageTitleCol);
    @include respond-to(medium-screens) {
        font-size: $headerSize - 7;
    }
    @include respond-to(handhelds) {
        font-size: $headerSize - 10;
    }
}
```

Koodinäide 4.11. Esimese taseme pealkirja kujundamine.

Esimesena määratakse pealkirja teksti suurus. Seejärel lisatakse juurde mixin "text3d" (vt koodinäide 4.1), mis loob pealkirjale 3D-effekti. Peale seda lisatakse koodibloki mixin'id "respond-to", mille ülesanneteks on muuta erinevate seadmetega, millega külastajad lehete vaatavad.

```
h2 {
    font-size: $headerSize - 12;
    @include text3d(darken($linkCol, 10%));
}
```

Koodinäide 4.12. Teise taseme pealkirja kujundamine.

```
#container {
    $width: $contWidth + $asideWidth;
    width: $width;
    margin: 0 auto 0 auto;
    @include respond-to(medium-screens) {
        width: $width - 250;
    }
    @include respond-to(handhelds) {
        width: auto;
    }
}
```

Koodinäide 4.13. Kujunduse määramine elemendile "container".

"container" on element, mille sees paiknevad kõik teised lehekülje elemendid. Koodibloki alguses on defineeritud lokaalne muutuja `$width`, mille eesmärgiks on

talletada väärtust, mis sisukasti ning parempoolse uudiseid sisaldava kasti laiuste liitmisel.

```
aside {
    width: $asideWidth - $asideGap - 2 * $asidePad;
    background-color: $asideBgCol;
    padding: $asidePad;
    float: right;
    @include rounded(10);
    article {
        border-bottom: $articleSepStyle;
    }
    @include respond-to(handhelds) {
        width: auto;
        float: none;
        margin-bottom: $asidePad;
    }
}
```

Koodinäide 4.14. Parempoolse riba kujunduse määramine.

Vaikimisi arvutatakse elemendi “aside” laius välja vastavalt muutuja väärtustele. Kui juhtub, et lehte külastatakse mobiilselt seadmelt, siis muudetakse elemendi asetust ning laiust.

```
section {
    $width: $contWidth - 2 * $contPad;
    width: $width;
    padding: $contPad;
    background-color: rgba($contBoxCol, .8);
    @include rounded(15);
    @include respond-to(medium-screens) {
        width: $width - 250px;
    }
    @include respond-to(handhelds) {
        width: auto;
    }
}
```

Koodinäide 4.15. Kujunduse määramine elemendile "section".

```
article {
    border-bottom: $articleSepLine;
```

```
    &:last-child {
        border: 0;
    }
}
```

Koodinäide 4.16. Kujunduse määramine elemendile "article"

Eelneva deklaratsiooniga määratakse elemendi “article” alumiseks piirjooneks failis “_variables.scss” defineeritud muutuja `$articleSepLine`. Kasutades viidet ülemise taseme selektorile (&) ning pseudoklassi `:last-child` eemaldatakse lehel olevalt viimaselt elemendilt “article” piirjoon.

```
p {
    &.three-columns {
        @include columns(3, 40, 1px dotted #777, justify);
    }
    &.two-columns {
        @include columns(2, 40, 1px dotted #777, justify);
    }
}
```

Koodinäide 4.17. Paragrahvid kujunduse määramine.

Eelneva koodilõiguga defineeritakse kahte eri tüüpi paragrahv. Esimene neist on kolme veeruga ning neljakümne piksli laiune. Teine on samade parameetritega kahe veeruga. Mõlemal juhul on paragrahvid eraldatud üksteisest joonega ning tekst nendes joondatud rööpselt. Veergude omaduste määramiseks kasutatakse *mixin*’it “column” (vt koodinäide 4.4). Kui kõrvutada koodinäites 4.17 kirjutatud kood lisas 3 toodud paragrahvide deklaratsiooniga, siis selgub, et peale kompileerimist on mõlema paragrahvi selektori kohta tekkinud ligi paarkümend rida koodi.

```
img {
    float: right;
    margin-left: 10px;
}
```

Koodinäide 4.18. Pildi kujundamine.

```
nav {
    margin-bottom: 62px;
    background-color: $mainColGreen;
    ul {
```

```
            list-style-type: none;
            padding: 0;
            li {
                background-color: lighten($linkCol, 5%);
                cursor: pointer;
                display: inline;
                float: left;
                text-transform: uppercase;
                &:first-child{
                    @include rounded-special(15, 0, 0, 15);
                }
                &:last-child{
                    @include rounded-special(0, 15, 15, 0);
                }
                &:hover {
                    background-color: darken($linkCol, 15%);
                }
                a, a:visited, a:hover {
                    text-decoration: none;
                    display: block;
                    width: 140px;
                    color: #FFFFFF;
                    font-weight: bold;
                    text-align: center;
                    padding: 4px;
                    @include respond-to(medium-screens) {
                        width: 120px;
                    }
                    @include respond-to(handhelds) {
                        width: 80px;
                    }
                }
            }
        }
}
```

Koodinäide 4.19. Menüüriba kujundamine.

Menüüriba kujundamisel on väga tähtis, et oleks täpselt paigas selektorid (st. lehekülje elemendid), mida kujundada soovitakse. Seda eelkõike sellepärast, et menüü loomisel on lehestikus kasutusel järjestamata list (ul). Kuna aga järjestamata listi kasutatakse peale menüü ka muu tekstilise sisu esitamisel, siis selleks, et ei tekiks konflikti menüüs ning sisus kasutatava listi vahel, on tähtis võimalikult täpne selektor.

```
a, a:visited {
    color: $linkCol;
    text-decoration: none;
    font-weight: bold;
    &:hover {
        text-decoration: underline;
        color: darken($linkCol, 20%);
    }
}
```

Koodinäide 4.20. Tavalise lingi kujundamine.

```
footer {
    $footerPad: $contPad / 2;
    $width: $contWidth - 2 * $footerPad;

    @extend section;
    @include respond-to(medium-screens) {
        width: $width - 250px;
    }
    @include respond-to(handhelds) {
        width: auto;
    }

    width: $width;
    padding: $footerPad;
    margin-top: $contElemSep;
    background-color: rgba($contBoxCol, .5);
    font-size: $normalTextSize;
}
```

Koodinäide 4.21. Lehekülje jaluse kujundamine.

Valminud näidislehestik on kättesaadaval aadressil:
http://www.tlu.ee/~strin/seminaritoo/naidislehestik/

4.4 Järeldus

Sass võimaldab kasutada koodi efektiivsemalt kui tavaline CSS. Seda väidet aitab põhjendada näidislehestikus loodud "_css3.scss" *partial*, mis toimib kui eraldi teek

(library) ning on piisavalt paindlik, et see võtta kasutusele tulevastes projektis. Iga projekti juures saab olemasolevat teeki laiendada ning muuta järjest paindlikumaks.

Vaatamata sellele, et üldjuhul võimaldab Sass arendada CSS'i kiiremini, peaks kasutaja pidevalt analüüsima, kas konkreetse probleemi või ülesande lahendamisel on otstarbekas kasutada Sass'i või saab seda teha kokkuvõttes paremini tavapärase CSS'iga.

Kindlasti tuleks analüüsida kompilaatori poolt väljastatavat CSS'i. Sass'iga algust tehes võib juhtuda, et väljastatav kood on palju pikem kui peaks. Seega peaks Sass'i kirjutades alati mõtlema, kuidas kirjutatud kood kompileerub. Samal teema jätkuks tasub lugeda Roy Tomeij'i poolt kirjutatud artiklit "Sass doesn't create bad code. Bad coders do.", mis asub järgmisel aadressil: http://thesassway.com/articles/sass-doesnt-create-bad-code-bad-coders-do

Kokkuvõtlik järeldus, milleni autor jõudis näidislehestikku koostades on see, et Sass'i kasutades ei pea ilmtingimata kasutama kõiki pakutavaid võimalusi. Olenevalt projektist piisab ka näiteks sellest kui kasutada lihtsalt muutujaid.

Kokkuvõte

Käesoleva seminaritöö ülesandeks oli anda lugejale ülevaade sellest, mida kujutab endast Sass ja milliseid võimalusi see pakub CSS'i efektiivsemaks kirjutamiseks. Samuti oli eesmärgiks hinnata Sass'i kasulikkust lehestiku loomisel.

Töös vaadati põgusalt, mis on CSS ning millised on selle tehnoloogia võimalused ja puudused. Seejärel tutvustati tehnoloogiat Sass ning selle võimalusi tavalise CSS'i laiendamiseks. Viimases peatükis kirjeldati Sass'i võimaluste katsetamiseks praktikas loodud näidislehestikku.

Seminaritöö andis autorile põhjalikud teadmised, mida kujutab endast Sass. Autor leiab, et iga lugeja, kelle igapäevane töö on veebilehekülgede või lihtsalt HTML-dokumentide kujundamine, millega seoses ta peab kirjutama CSS'i, peaks võtma korraks aja maha ning tutvuma Sass'i või mõne teise kompileeritava CSS'i metakeelega. Kuigi alguses võib uue süntaksi ja selle võimalustega harjumine võtta teataval määral aega, on suure tõenäosusega peale Sass'iga harjumist märgata tunduvalt produktiivsuse kasvu ning koodi organiseerimise paranemist. Vaatamata sellele tuleb püüda analüüsida igat ette sattuvat probleemi eraldi, sest alati ei pruugi olla otstarbekas kasutada lahenduse otsimisel Sass'i.

Kasutatud kirjanduse loetelu

Catlin, H., & Catlin, L. M. (2011). *Pragmatic Guide to Sass.* (K. Kay, Toim.) Raleigh, United States of America: Pragmatic Bookshelf.

Why use CSS? - CSS | MDN. (3. september 2012. a.). Kasutamise kuupäev: 13. oktoober 2012. a., allikas Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/CSS/Getting_Started/Why_use_CSS

Eppstein, C. (2011). Code smarter CSS with Sass. *.NET Magazine* (211).

Ricalde, M. (20. november 2011. a.). *Nested Selectors: The Inception Rule*. Kasutamise kuupäev: 12. oktoober 2012. a., allikas The Sass Way: http://thesassway.com/beginner/the-inception-rule

Mutually Human. (kuupäev puudub). *Compass and Sass without all the hassle*. Kasutamise kuupäev: 14. oktoober 2012. a., allikas Scout: http://mhs.github.com/scout-app/

Catlin, H., Weizenbaum, N., & Eppstein, C. (kuupäev puudub). *Sass*. Kasutamise kuupäev: 4. oktoober 2012. a., allikas Sass - Syntactically Awesome Stylesheets: http://sass-lang.com/tutorial.html

Netherland, W., Weizenbaum, L. N., Eppstein, M. C., & Mathis, B. (2012). *Sass and Compass in Action* (MEAP Edition tr.). Shelter Island, New York: Manning Publications.

Schmitt, C. (2010). *CSS Cookbook* (3rd Edition tr.). (S. St.Laurent, Toim.) Sebastopol, California, United States of America: O'Reilly Media.

Topfunky Corporation. (12. jaanuar 2009. a.). Haml & Sass. Seattle, Washington, United States of America.

York, R. (2005). *Beginning CSS: Cascading Style Sheets for Web Design.* Indianapolis: Wiley Publishing.

Lie, W. H., & Bos, B. (1999). *Cascading Style Sheets: Designing for the Web* (2nd Edition tr.). Boston: Addison-Wesley Professional.

Bos, B., Lie, W. H., Lilley, C., & Jacobs, I. (11. aprill 2008. a.). *Cascading Style Sheets, level 2*. Kasutamise kuupäev: 20. oktoober 1021. a., allikas World Wide Web Consortium (W3C): http://www.w3.org/TR/2008/REC-CSS2-20080411/

The World Wide Web Consortium. (7. juuni 2011. a.). *About the CSS 2.1 Specification*. (B. Bos, T. Çelik, I. Hickson, & W. H. Lie, Toimetajad) Kasutamise kuupäev: 20. oktoober 2012. a., allikas Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification: http://www.w3.org/TR/CSS2/about.html

Robbins, N. J. (2007). *Learning Web Design: A Beginner's Guide to (X)HTML, StyleSheets, and Web Graphics* (3rd Edition tr.). (A. Gustafson, Toim.) Sebastopol: O'Reilly Media.

Gillenwater, M. Z. (2011). *Stunning CSS3: A project-based guide to the latest in CSS .* Berkeley: New Riders.

Catlin, H., Eppstein, C., & Weizenbaum, N. (28. september 2012. a.). *Module: Sass::Script::Functions*. Kasutamise kuupäev: 28. oktoober 2012. a., allikas SASS: http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html

Demaree, D. (29. november 2011. a.). *Getting Started with Sass*. Kasutamise kuupäev: 24. oktoober 2012. a., allikas A List Apart: http://www.alistapart.com/articles/getting-started-with-sass/

Gallagher, N. (28. veebruar 2012. a.). *About normalize.css - Nicolas Gallagher*. Kasutamise kuupäev: 3. november 2012. a., allikas Nicolas Gallagher: http://nicolasgallagher.com/about-normalize-css/

Catlin, H., Eppstein, C., & Weizenbaum, N. (2. november 2012. a.). *File: SASS_REFERENCE*. Kasutamise kuupäev: 4. november 2012. a., allikas SASS: http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html

Võõrkeelsete lühendite loetelu

HTML – *HyperText Markup Language*, hüperteksti märgistuskeel, mida kasutavad pea kõik internetis olevad veebilehekülgedest.
HTML5 – tavalise *HTML*'I kõige viimane, veel arenduses olev version.
CSS - *Cascading Style Sheets*, astmelised stiililehed.
CSS3 – CSS'i viimane, veel arenduses olev spetsifikatsioon.
OOCSS – *Object Oriented Cascading Style Sheets*, objektorjenteeritud astmelised stiililehed.
SASS - *Syntactically Awesome Stylesheets*, metakeel, mille eesmärgiks on laiendada tavalise CSS'i võimalusi, luua eeldused koodi kiiremaks kirjutamiseks ning paremaks organiseerimiseks.
DOM - *Document Object Model*, dokumendi objektmudel, HTML ja XML dokumentidega suhtlemise liides.
HTTP - HyperText Transfer Protocol, andmevahetusprotokoll, mille ülesandeks on veebidokumentide vahetamine internetis.
PHP - *Hypertext Preprocessor*, laiandaselt levinud dünaamiliste veebilehekülgede loomiseks mõeldud skriptimiskeel.

LISAD

Lisa 1: näidislehestik – HTML-struktuur (index.html)

```
<!DOCTYPE HTML>
<html lang="et">
<head>
    <meta charset="UTF-8">
    <title>Näidislehestik | SASS</title>
    <link rel="stylesheet" href="css/main.css">
</head>
<body>
    <div id="container">
        <header>
            <h1>Sass'i kasutamine veebilehe loomisel</h1>
        </header>
        <nav>
            <ul>
                <li><a href="index.htm">Esileht</a></li>
                <li><a href="index.htm">link</a></li>
                <li><a href="forms.htm">Vormid</a></li>
            </ul>
        </nav>
        <aside>
            <h2>Uudised</h2>
            <article>
                <h3>Lorem ipsum dolor sit amet</h3>
                <p>
                    Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua.
                </p>
            </article>
            <article>
                <h3>Ut enim ad minim veniam</h3>
                <p>
                    Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur.
                </p>
            </article>
            <article>
                <h3>Duis aute irure dolor</h3>
                <p>
```

```
                    Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur.
                </p>
            </article>
        </aside>
        <section>
            <article>
                <p>
                    See leht on valminud seminaritöö "<strong>CSS-i
laiendusvõimalused läbi SASS-i. ning nende kasutamine
veebirakenduste loomisel</strong>" raames. Lehe kujunduse loomisel
on kasutatud Sass'i, mis hiljem on kompileeritud tavaliseks CSS'iks.
SCSS fail, mis on antud lehe kujunduse aluseks on soovikorral
kättesaadav <a href="sass/main.scss">siit</a>. Peale kompileerimist
saadud CSS fail asub <a href="css/main.css"
target="_blank">siin</a>.
                </p>
                <h2>Sass</h2>
                <p>
                    <img src="images/sass_logo.jpeg" alt="Sass"
width="100" height="110">Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.
                </p>
                <p>
                    Rohkem infot ametlikult kodulehelt: <a
href="http://www.sass-lang.com" target="_blank">http://www.sass-
lang.com</a>
                </p>
                    <strong>Sass'is teegid, mis võimaldavad koodi
veelgi väledamalt kirjutada:</strong>
                    <ul>
                        <li><a href="http://compass-style.org"
target="_blank">Compass</a></li>
                        <li><a href="http://thoughtbot.com/bourbon/"
target="_blank">Bourbon</a></li>
                    </ul>
```

```
            </article>
            <article>
                <h2>Pealkiri 2</h2>
                <p class="two-columns">
                    Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.
                </p>
                <p>
                    Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.
                </p>
            </article>
            <article>
                <h2>CSS3</h2>
                <p class="three-columns">
                    Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.
                </p>
            </article>
        </section>
        <footer>
            labore et dolore magna aliqua
        </footer>
    </div>
</body>
</html>
```

Lisa 2: näidislehestik – HTML-struktuur (forms.html)

```
<!DOCTYPE HTML>
<html lang="et">
<head>
    <meta charset="UTF-8">
    <title>Näidislehestik | SASS</title>
    <link rel="stylesheet" href="css/main.css">
</head>
<body>
    <div id="container">
        <header>
            <h1>Sass'i kasutamine veebilehe loomisel</h1>
        </header>
        <nav>
            <ul>
                <li><a href="index.htm">Esileht</a></li>
                <li><a href="index.htm">link</a></li>
                <li><a href="forms.htm">Vormid</a></li>
            </ul>
        </nav>
        <aside>
            <h2>Uudised</h2>
            <article>
                <h3>Lorem ipsum dolor sit amet</h3>
                <p>
                    Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua.
                </p>
            </article>
            <article>
                <h3>Ut enim ad minim veniam</h3>
                <p>
                    Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur.
                </p>
            </article>
            <article>
                <h3>Duis aute irure dolor</h3>
                <p>
```

```
                    Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur.
                </p>
            </article>
        </aside>
        <section>
            <article>
                <p>
                    See leht on valminud seminaritöö "<strong>CSS-i
laiendusvõimalused läbi SASS-i. ning nende kasutamine
veebirakenduste loomisel</strong>" raames. Lehe kujunduse loomisel
on kasutatud Sass'i, mis hiljem on kompileeritud tavaliseks CSS'iks.
SCSS fail, mis on antud lehe kujunduse aluseks on soovikorral
kättesaadav <a href="sass/main.scss">siit</a>. Peale kompileerimist
saadud CSS fail asub <a href="css/main.css"
target="_blank">siin</a>.
                </p>
                <h2>Vormid</h2>
                <div id="contact">
                    <form>
                    <p>
                        <label for="name">Nimi:
                            <span class="small">Sisesta oma
täisnimi</span>
                        </label>
                        <input type="text" name="name" id="name">
                    </p>
                    <p>
                        <label for="email">E-post:
                            <span class="small">Sisesta oma e-posti
aadress</span>
                        </label>
                        <input type="text" name="email" id="email">
                    </p>
                    <p>
                        <label for="message">Kirja sisu:
                            <span class="small">Kirjuta, mis on
mõttes</span>
                        </label>
                        <textarea name="message"
id="message"></textarea>
                    </p>
```

```
                    <p>
                        <button type="submit">Saada kiri</button>
                    </p>
                    </form>
                </div>
            </article>
        </section>
        <footer>
            labore et dolore magna aliqua
        </footer>
    </div>
</body>
</html>
```

Lisa 3: näidislehestik – kompileeritud kujul (main.css)

```
#contact p {
  border-bottom: dotted 1px #a8cf34;
  clear: left;
}
#contact p:last-child {
  border: 0;
}
#contact label {
  display: block;
  font-weight: bold;
  text-align: right;
  width: 140px;
  float: left;
}
#contact .small {
  color: #666666;
  display: block;
  font-size: 11px;
  font-weight: normal;
  text-align: right;
  width: 140px;
}
#contact input, #contact textarea {
  float: left;
  padding: 10px 4px;
  border: solid 1px #677f1e;
  margin: 5px 10px 5px 10px;
  width: 490px;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  -ms-border-radius: 5px;
  -o-border-radius: 5px;
  border-radius: 5px;
  background-color: #ddecb0;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  #contact input, #contact textarea {
    width: 260px;
  }
}
@media only screen and (max-width: 480px) {
```

```
  #contact input, #contact textarea {
    width: 150px;
  }
}
#contact textarea {
  height: 80px;
}
#contact button {
  margin-left: 140px;
}

body {
  background-image: url(../images/stripes.png);
  margin-top: 40px;
  font-family: 'Trebuchet MS', Helvetica, sans-serif;
  line-height: 22px;
  font-size: 14px;
  color: #333333;
}

h1 {
  font-size: 38px;
  color: #b738b8;
  text-shadow: 1px 1px 0px #a432a4, 2px 2px 0px #902c91;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  h1 {
    font-size: 31px;
  }
}
@media only screen and (max-width: 480px) {
  h1 {
    font-size: 28px;
  }
}

h2 {
  font-size: 26px;
  color: #88a828;
  text-shadow: 1px 1px 0px #779423, 2px 2px 0px #677f1e;
}
```

```
#container {
  width: 1000px;
  margin: 0 auto 0 auto;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  #container {
    width: 750px;
  }
}
@media only screen and (max-width: 480px) {
  #container {
    width: auto;
  }
}

aside {
  width: 220px;
  background-color: #cbe386;
  padding: 10px;
  float: right;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  -o-border-radius: 10px;
  border-radius: 10px;
}
aside article {
  border-bottom: 1px solid #677f1e;
}
@media only screen and (max-width: 480px) {
  aside {
    width: auto;
    float: none;
    margin-bottom: 10px;
  }
}

section, footer {
  width: 690px;
  padding: 30px;
  background-color: rgba(255, 255, 255, 0.8);
  -webkit-border-radius: 15px;
```

```
  -moz-border-radius: 15px;
  -ms-border-radius: 15px;
  -o-border-radius: 15px;
  border-radius: 15px;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  section, footer {
    width: 440px;
  }
}
@media only screen and (max-width: 480px) {
  section, footer {
    width: auto;
  }
}

article {
  border-bottom: 1px dotted #f1f1f1;
}
article:last-child {
  border: 0;
}

p.three-columns {
  -moz-column-count: 3;
  -webkit-column-count: 3;
  column-count: 3;
  -moz-column-gap: 40px;
  -webkit-column-gap: 40px;
  column-gap: 40px;
  -moz-column-rule: 1px dotted #777777;
  -webkit-column-rule: 1px dotted #777777;
  column-rule: 1px dotted #777777;
  text-align: justify;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  p.three-columns {
    -moz-column-count: 2;
    -webkit-column-count: 2;
    column-count: 2;
    -moz-column-gap: 40px;
    -webkit-column-gap: 40px;
```

```
    column-gap: 40px;
  }
}
@media only screen and (max-width: 480px) {
  p.three-columns {
    -moz-column-count: 1;
    -webkit-column-count: 1;
    column-count: 1;
    -moz-column-gap: 40px;
    -webkit-column-gap: 40px;
    column-gap: 40px;
  }
}
p.two-columns {
  -moz-column-count: 2;
  -webkit-column-count: 2;
  column-count: 2;
  -moz-column-gap: 40px;
  -webkit-column-gap: 40px;
  column-gap: 40px;
  -moz-column-rule: 1px dotted #777777;
  -webkit-column-rule: 1px dotted #777777;
  column-rule: 1px dotted #777777;
  text-align: justify;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  p.two-columns {
    -moz-column-count: 1;
    -webkit-column-count: 1;
    column-count: 1;
    -moz-column-gap: 40px;
    -webkit-column-gap: 40px;
    column-gap: 40px;
  }
}
@media only screen and (max-width: 480px) {
  p.two-columns {
    -moz-column-count: 0;
    -webkit-column-count: 0;
    column-count: 0;
    -moz-column-gap: 40px;
    -webkit-column-gap: 40px;
```

```
    column-gap: 40px;
  }
}

img {
  float: right;
  margin-left: 10px;
}

nav {
  margin-bottom: 62px;
  background-color: #a8cf34;
}
nav ul {
  list-style-type: none;
  padding: 0;
}
nav ul li {
  background-color: #b1d449;
  cursor: pointer;
  display: inline;
  float: left;
  text-transform: uppercase;
}
nav ul li:first-child {
  border-radius: 15px 0px 0px 15px;
  -moz-border-radius: 15px 0px 0px 15px;
  -webkit-border-radius: 15px 0px 0px 15px;
}
nav ul li:last-child {
  border-radius: 0px 15px 15px 0px;
  -moz-border-radius: 0px 15px 15px 0px;
  -webkit-border-radius: 0px 15px 15px 0px;
}
nav ul li:hover {
  background-color: #779423;
}
nav ul li a, nav ul li a:visited, nav ul li a:hover {
  text-decoration: none;
  display: block;
  width: 140px;
  color: #FFFFFF;
```

```
  font-weight: bold;
  text-align: center;
  padding: 4px;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  nav ul li a, nav ul li a:visited, nav ul li a:hover {
    width: 120px;
  }
}
@media only screen and (max-width: 480px) {
  nav ul li a, nav ul li a:visited, nav ul li a:hover {
    width: 80px;
  }
}

a, a:visited {
  color: #a8cf34;
  text-decoration: none;
  font-weight: bold;
}
a:hover, a:visited:hover {
  text-decoration: underline;
  color: #677f1e;
}

footer {
  width: 720px;
  padding: 15px;
  margin-top: 10px;
  background-color: rgba(255, 255, 255, 0.5);
  font-size: 14px;
}
@media only screen and (min-width: 480px) and (max-width: 1024px) {
  footer {
    width: 470px;
  }
}
@media only screen and (max-width: 480px) {
  footer {
    width: auto;
  }
}
```

Lisa 4: näidislehestik – Sass (main.scss)

```
@import "variables";
@import "css3";
@import "forms";

body {
    background-image: url(#{$bgImage});
    margin-top: $siteMargin;
    font-family: $fonts;
    line-height: $lineHeight;
    font-size: $normalTextSize;
    color: $textCol;
}

h1 {
    font-size: $headerSize;
    @include text3d($pageTitleCol);
    @include respond-to(medium-screens) {
        font-size: $headerSize - 7;
    }
    @include respond-to(handhelds) {
        font-size: $headerSize - 10;
    }
}

h2 {
    font-size: $headerSize - 12;
    @include text3d(darken($linkCol, 10%));
}

#container {
    $width: $contWidth + $asideWidth;
    width: $width;
    margin: 0 auto 0 auto;
    @include respond-to(medium-screens) {
        width: $width - 250;
    }
    @include respond-to(handhelds) {
        width: auto;
    }
}
```

```
aside {
    width: $asideWidth - $asideGap - 2 * $asidePad;
    background-color: $asideBgCol;
    padding: $asidePad;
    float: right;
    @include rounded(10);
    article {
        border-bottom: $articleSepStyle;
    }
    @include respond-to(handhelds) {
        width: auto;
        float: none;
        margin-bottom: $asidePad;
    }
}

section {
    $width: $contWidth - 2 * $contPad;
    width: $width;
    padding: $contPad;
    background-color: rgba($contBoxCol, .8);
    @include rounded(15);
    @include respond-to(medium-screens) {
        width: $width - 250px;
    }
    @include respond-to(handhelds) {
        width: auto;
    }
}

article {
    border-bottom: $articleSepLine;
    &:last-child {
        border: 0;
    }
}

p {
    &.three-columns {
        @include columns(3, 40, 1px dotted #777, justify);
    }
    &.two-columns {
```

```
        @include columns(2, 40, 1px dotted #777, justify);
    }
}

img {
    float: right;
    margin-left: 10px;
}

nav {
    margin-bottom: 62px;
    background-color: $mainColGreen;
    ul {
        list-style-type: none;
        padding: 0;
        li {
            background-color: lighten($linkCol, 5%);
            cursor: pointer;
            display: inline;
            float: left;
            text-transform: uppercase;
            &:first-child{
                @include rounded-special(15, 0, 0, 15);
            }
            &:last-child{
                @include rounded-special(0, 15, 15, 0);
            }
            &:hover {
                background-color: darken($linkCol, 15%);
            }
            a, a:visited, a:hover {
                text-decoration: none;
                display: block;
                width: 140px;
                color: #FFFFFF;
                font-weight: bold;
                text-align: center;
                padding: 4px;
                @include respond-to(medium-screens) {
                    width: 120px;
                }
                @include respond-to(handhelds) {
```

```
                    width: 80px;
                }
            }
        }
    }
}

a, a:visited {
    color: $linkCol;
    text-decoration: none;
    font-weight: bold;
    &:hover {
        text-decoration: underline;
        color: darken($linkCol, 20%);
    }
}

footer {
    $footerPad: $contPad / 2;
    $width: $contWidth - 2 * $footerPad;

    @extend section;
    @include respond-to(medium-screens) {
        width: $width - 250px;
    }
    @include respond-to(handhelds) {
        width: auto;
    }

    width: $width;
    padding: $footerPad;
    margin-top: $contElemSep;
    background-color: rgba($contBoxCol, .5);
    font-size: $normalTextSize;
}
```

Lisa 5: näidislehestik – Sass (_variables.scss)

```
// üldised lehe muutujad
$contBoxCol: #ffffff;
$mainColGreen: #A8CF34;
$linkCol: $mainColGreen;
$textCol: #333;
$pageTitleCol: #b738b8;
$bgImage: "../images/stripes.png";
$articleSepLine: 1px dotted #f1f1f1;
$siteMargin: 40px;

// teksti suurused
$normalTextSize: 14px;
$headerSize: 38px;
$lineHeight: 22px;
$fonts: 'Trebuchet MS', Helvetica, sans-serif;

// sisukast
$contWidth: 750px;
$contPad: 30px;
$contElemSep: 10px;

// külgriba
$asideWidth: 250px;
$asideGap: 10px;
$asidePad: 10px;
$asideBgCol: lighten($mainColGreen, 20%);
$articleSepStyle: 1px solid darken($mainColGreen, 20%);

// leht "vormid"
$fieldSepStyle: dotted 1px $mainColGreen;
$labelWidth: 140px;

// teksti sisestusväljad
$inputWidth: $contWidth - $labelWidth - $contPad * 4;
```

```
$inputMargin: 5px 10px 5px 10px;
$inputPad: 10px 4px;
$inputBorderStyle: solid 1px darken($mainColGreen, 20%);
$textareaHeight: 80px;
```

Lisa 6: näidislehestik – Sass (_css3.scss)

```
//
//  CSS3 juurde kuuluvad võimalused
//

@mixin text3d($color) {
  color: $color;
  text-shadow: 1px 1px 0px darken($color, 5%),
               2px 2px 0px darken($color, 10%);
}

//
//  Ümarad nurgad
//

@mixin rounded($radius: 10) {
    -webkit-border-radius: #{$radius}px;
    -moz-border-radius: #{$radius}px;
    -ms-border-radius: #{$radius}px;
    -o-border-radius: #{$radius}px;
    border-radius: #{$radius}px;
}

@mixin rounded-special($topLeft, $topRight, $bottomRight,
$bottomLeft) {
    border-radius: #{$topLeft}px #{$topRight}px #{$bottomRight}px
#{$bottomLeft}px;
    -moz-border-radius: #{$topLeft}px #{$topRight}px
#{$bottomRight}px #{$bottomLeft}px;
    -webkit-border-radius: #{$topLeft}px #{$topRight}px
#{$bottomRight}px #{$bottomLeft}px;
}

//
// mitmeveeruline asetus
//

@mixin columns($count, $gap, $rule: null, $alignment: default) {
    @include column-properties($count, $gap);

    @if $rule != null {
        -moz-column-rule: $rule;     // Firefox
```

```
        -webkit-column-rule: $rule;  // Safari, Chrome
        column-rule: $rule;
    }
    @if $alignment != default {
        text-align: $alignment;
    }

    @include respond-to(medium-screens) {
        @include column-properties($count - 1, $gap);
    }
    @include respond-to(handhelds) {
        @include column-properties($count - 2, $gap);
    }
}

// veergude omadused mitmeveerulise asetuse korral
@mixin column-properties($count, $gap) {
    -moz-column-count: $count;
    -webkit-column-count: $count;
    column-count: $count;

    -moz-column-gap: #{$gap}px;     // Firefox
    -webkit-column-gap: #{$gap}px;  // Safari, Chrome
    column-gap: #{$gap}px;
}

//
//  meediapäringud
//
@mixin respond-to($media) {
    @if $media == handhelds {
        @media only screen and (max-width: 480px) {
            @content;
        }
    }
    @else if $media == medium-screens {
        @media only screen and (min-width: 481px) and (max-width:
1023px) {
            @content;
        }
    }
    @else if $media == wide-screens {
```

```
        @media only screen and (min-width: 1024px) {
            @content;
        }
    }
}
```

Lisa 7: näidislehestik – Sass (_forms.scss)

```
#contact {
    p {
        border-bottom: $fieldSepStyle;
        &:last-child {
            border: 0;
        }
        clear: left;
    }

    label {
        display:block;
        font-weight:bold;
        text-align:right;
        width: $labelWidth;
        float:left;
    }

    .small {
        color: #666666;
        display: block;
        font-size: 11px;
        font-weight: normal;
        text-align: right;
        width: $labelWidth;
    }

    input {
        float:left;
        padding: $inputPad;
        border: $inputBorderStyle;
        margin: $inputMargin;
        width: $inputWidth;
        @include rounded(5);
        background: {
            color: lighten($mainColGreen, 30%);
        }
        @include respond-to(medium-screens) {
            width: 260px;
        }
        @include respond-to(handhelds) {
            width: 150px;
```

```
        }
    }

    textarea {
        @extend input;
        height: $textareaHeight;
    }

    button {
        margin-left: $labelWidth;
    }
}
```