CSS-ESIPROSESSORIEN VERTAILU

Sass, Less ja Stylus

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, Tietojenkäsittelyn koulutusohjelma

kevät, 2017

Ville Valtonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki

Tekijä Ville Valtonen **Vuosi** 2017

Työn nimi CSS-esiprosessorien vertailu: Sass, Less ja Stylus

Työn ohjaaja /t Lauri Salminen

TIIVISTELMÄ

Opinnäytetyön päätavoitteina oli tutkia, mitä CSS-esiprosessointi tarkoittaa ja mitä lisäarvoa CSS-esiprosessorit voivat tuottaa CSS-tyyliohjeiden käsittelyyn. Lisäksi siinä vertailtiin keskenään kolmea suosittua CSS-esiprosessoria sekä otettiin selvää, millaisia eri vaihtoehtoja www-sisällönhallintajärjestelmä WordPress tarjoaa niiden hyödyntämiseen. Opinnäytetyön toimeksiantaja oli hämeenlinnalainen Aatos Media.

Teoriaosuudessa esitellään aluksi web-kehityksen perustekniikoita ns. front-endin näkökulmasta pääpainon ollessa CSS-tyyliohjeissa. Lisäksi siinä käsitellään www-sisällönhallintajärjestelmiä ja käydään läpi, mitä CSS:n esiprosessointi tarkoittaa yleisellä tasolla.

Käytännön osuudessa tutkitaan CSS-esiprosessoreja ja niiden toimintaa muun muassa käytännön esimerkein sekä vertaillaan Sass-, Less ja Stylus-esiprosessorien eri ominaisuuksia keskenään. Lopuksi tarkastellaan, miten kyseisiä CSS-esiprosessoreja voidaan hyödyntää WordPress-kehityksessä.

Työn päätuloksena saatiin selville, että oikein käytettyinä CSS-esiprosessorit voivat tuottaa todellista lisäarvoa jo melko pienissä projekteissa. Lisäksi vertailun perusteella todettiin, että ainakin markkinoiden suosituimpiin kuuluvien CSS-esiprosessorien ominaisuuksien erot ovat kokonaisuudessaan melko pieniä, mutta Sassia voidaan pitää monipuolisimpana vaihtoehtona.

Avainsanat CSS, esiprosessori, web-kehitys**,** vertailu

Sivut 28 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Ville Valtonen

Author Ville Valtonen **Year** 2017

Subject Comparison of CSS Preprocessors: Sass, Less and Stylus

Supervisors Lauri Salminen

ABSTRACT

The main goals of this thesis were to examine what CSS preprocessing means and what type of value CSS preprocessors could add to CSS. Three different popular CSS preprocessors were compared and it was also investigated what kind of options web content management system WordPress provides for utilizing the three preprocessors. The thesis was commissioned by Hämeenlinna-based Aatos Media.

The theory section begins with an introduction to basic front-end web development techniques with main emphasis being on CSS. Web content management systems are also studied. Finally, the section examines what CSS preprocessing means in general.

The practical section begins with examining CSS preprocessors with practical examples, as well as comparing different characteristics of three preprocessors, namely Sass, Less and Stylus. In the last part, it is scrutinized how CSS preprocessors could be used in WordPress development.

Main result of the thesis was that when used correctly CSS preprocessors could add value even to relatively small projects. In addition, based on the comparison it was found that differences in the most popular CSS preprocessors are rather small, but Sass can be considered the most versatile option.

Keywords CSS, preprocessor, web development, comparison

Pages 28 pages

SISÄLLYS

KÄSITTEITÄ

Olio-ohjelmointikieli	Ohjelmointikieli, joka koostuu tietyistä samaa toiminnallisuutta tai tietoa sisältäviä joukoista, olioista.
Ruby	Olio-ohjelmointikieli
Syntaksi	Lauseoppi, joka kuvaa ohjelmointikielissä koodin oikean rakenteen.

1 JOHDANTO

Verkkosivujen rakenteiden ja niihin käytettävien tekniikoiden monipuolistuessa ja monimutkaistuessa on pyritty kehittämään erilaisia apuvälineitä helpottamaan ja nopeuttamaan web-kehityksen työnkulkua. Myös CSS-tyyliohjeiden käyttöä on yritetty sujuvoittaa muun muassa erilaisilla CSS-esiprosessoreilla. Tämän opinnäytetyön päätarkoituksena on selvittää, mitä CSS-esiprosessorit ovat sekä mitä lisäarvoa niiden käyttäminen voi tuottaa pelkästään tavallisten CSS-tyyliohjeiden käsittelyyn verrattuna. Työssä myös vertaillaan keskenään kolmea suosittua esiprosessoria. Opinnäytetyön toimeksiantajana toimii Aatos Media.

Työ alkaa teoriaosuudella, jossa esitellään front-end-web-kehityksen yleisimpiä tekniikoita. Pääpainotus on kuitenkin CSS-tyyliohjeiden käsittelyssä. Työssä käytetään näistä uusimpia tekniikoita, eli HTML5-merkintäkieltä ja CSS3:a. Kuitenkin esimerkiksi ns. back-end-tekniikat, kuten PHP on rajattu kokonaan aiheen ulkopuolelle.

Käytännön osuudessa vertaillaan kolmea CSS-esiprosessoria valittujen kriteereiden mukaan. Lisäksi siinä tutkitaan, miten esiprosessoreja voidaan käyttää hyödyksi www-sisällönhallintajärjestelmä WordPressin kanssa.

Opinnäytetyössä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:
Mitä CSS-esiprosessorit ovat?
Miten ja missä tilanteissa niitä voi ja kannattaa hyödyntää?
Miten CSS-esiprosessorit toimivat sisällönhallintajärjestelmän kanssa?

2 FRONT-END-KEHITYKSEN TEKNIIKOITA

Web-kehityksessä termillä front-end tarkoitetaan esimerkiksi verkkosivuilla tavalliselle käyttäjälle ulospäin näkyvää osaa, kuten käyttöliittymää ja ulkoasua. Nykyään käyttäjän on myös yleensä mahdollista toimia sivulla responsiivisesti sekä perinteisillä tietokoneilla että mobiililaitteilla. Front-endin alle kuuluvat kolme toisiinsa melko tiiviisti liittyvää tekniikkaa: HTML, CSS ja JavaScript. Näiden tekniikoiden ja niiden yhteiskäytön lisäksi käyttöliittymäsuunnittelu on osa front-end-kehitystä. (Wodehouse n.d.)

Front-endin vastakohta on termi back-end. Se tarkoittaa web-kehityksessä palvelin- ja tietokantapuolta, joka ei yleensä ole suoraan näkyvissä tai käytettävissä tavalliselle käyttäjälle. Back-end vastaa esimerkiksi tietojen tai vaihtoehtojen tuomisesta verkkosivuilla olevalle lomakkeelle. Käytettäviä tekniikoita ja ohjelmointikieliä ovat muun muassa PHP, Ruby tai Python. (Wodehouse n.d.)

2.1 HTML-kieli

Web-kehityksen näkökulmasta HTML (HyperText Markup Language) -merkkauskieltä käytetään määrittelemään verkkosivujen sisällön rakenne sekä mitä siellä sijaitsevat eri elementit (esimerkiksi otsikot, leipäteksti, kuvat, lomakkeet) tarkoittavat. Se ei kuitenkaan ota kantaa siihen, miltä kyseiset asiat näyttävät. (Harris 2014, 11.)

HTML:n historia ulottuu 90-luvun alkuun. Alkuvaiheessa kehitys oli melko sekavaa eri verkkoselainten tekijöiden lisäillessä kieleen omia laajennuksiaan ja ominaisuuksiaan, jotka kuitenkin yleensä myöhemmin virallistettiin varsinaisiin versioihin mukaan. (Korpela 2014, 28.)

Vuodesta 1997 lähtien HTML:ää on kehittänyt W3C (World Wide Web Consortium) -organisaatio, joka koostuu suurimmaksi osaksi web-kehittämiseen keskittyneistä yrityksistä. Vuonna 1999 julkaistiin HTML 4.01, joka oli yli kymmenen vuoden ajan kielen viimeisin virallinen suositus. Kielestä julkaistiin myös XML-kieleen pohjautuva XHTML-versio vuonna 2000. (Korpela 2014, 26, 28.) Tällä hetkellä uusin HTML-versio on HTML5, josta W3C teki virallisen suosituksensa loppuvuodesta 2014. Lisäksi kehityksen alla on HTML5.1, josta on annettu kandidaattisuositus vuonna 2016. (W3C, n.d.d.)

Alla on esimerkki yksinkertaisen HTML-verkkosivun merkinnästä. HTML-tiedosto koostuu erilaisista sisäkkäin merkittävistä tageista, joiden sisälle verkkosivuston rakenteet (elementit) tuotetaan. Aivan ensimmäisellä rivillä määritellään verkkoselaimen käytettäväksi HTML:ää. Koko sivu tuotetaan Html-tagien väliin. Head-tagin sisällä määritellään muun muassa sivun otsikko ja viittaukset mahdollisiin eri tiedostoihin. Body-tagien väliin tulevat taas sivun varsinaiset sisältöelementit.

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>Esimerkkisivu</title>
<link href="sivusto.css" rel="stylesheet" type="text/css">
</head>
<body>

<h1>Esimerkkisivun pääotsikko</h1>
<p>Esimerkkisivun leipätekstiä.</p>
<img src="kuva.jpg" alt="esimerkkikuva">

</body>
</html>
```

2.1.1 HTML5

Suosituksensa mukaisesti HTML5 pyrkii korjaamaan aiemmissa versioissa olleita isoja puutteita, jotka liittyvät suurimmaksi osaksi erilaisten web-sovellusten alueelle. HTML5 onkin merkintäkielen aiempiin versioihin verrattuna huomattavasti enemmän niihin suuntautunut. (Korpela 2014, 26.)

Varsinaisen HTML-kielen näkökulmasta uusia ominaisuuksia on tullut muun muassa uusien multimediaelementtien, lomakkeiden ja graafisten ominaisuuksien alueelle. Lisäksi joitakin elementtejä on poistettu kokonaan tai siirretty ne määriteltäväksi tyyliohjeissa. (W3C, n.d.d.)

Riippuen siitä missä yhteydessä HTML5:stä puhutaan, sillä voidaan tarkoittaa eri asioita. Tiukimman määritelmän mukaan sillä tarkoitetaan HTML-merkintäkielen uusinta suositusversiota. Toisen määrittelyn mukaan sitä voidaan käyttää yleisesti nimityksenä suurelle joukolle moderneja webtekniikoita. Kolmannen määritelmän mukaan sillä voidaan tarkoittaa yleisesti sovellusten toteuttamista kaikkia näitä tekniikoita käyttäen (HTML, CSS, JavaScript.) (Korpela 2014, 2.)

2.2 **JavaScript**

Javascript (viralliselta nimeltään ECMAScript) on dynaaminen ohjelmointikieli, jota käytetään suurimmaksi osaksi osana verkkosivuja erilaisten selainskriptien tekemiseen. JavaScript sisältää useita olio-ohjelmointikielien ominaisuuksia. Sen on alun perin kehittänyt Netscape vuonna 1995. Ensimmäisen virallisen standardisointinsa se sai vuonna 1997. (Tutorialspoint 2017.)

Selainskriptillä tarkoitetaan koodia, joka ajetaan selaimen toimesta samalla, kun verkkosivuja näytetään. JavaScriptistä on tullut käytännössä ainoa selainohjelmoinnissa käytettävä kieli lähes jokaisen nykyaikaisen webselaimen tukiessa sen ominaisuuksia. Selainohjelmoinnin (client-side scripting) lisäksi JavaScriptiä voidaan käyttää myös palvelinohjelmointiin (server-side scripting). (Korpela 2014, 55−56.)

Selainohjelmoinnin näkökulmasta JavaScriptin hyötyjä ovat muun muassa mahdollisuus toteuttaa erilaisia interaktiivisia ominaisuuksia ilman, että käyttäjän tarvitsee päivittää selainta, tai että tietoja tarvitsee lähettää palvelimelle. Tällainen ominaisuus voi olla esimerkiksi reaaliaikainen ilmoitus salasanasta, joka ei vastaa määriteltyjä salasanojen vaatimuksia. HTML-dokumentissa JavaScript merkitään <script>-tagien väliin. (Tutorialspoint 2017).

3 WWW-SISÄLLÖNHALLINTAJÄRJESTELMÄT

1990-luvun loppupuolelta lähtien on alettu kehittää erilaisia WWW-sisällönhallintajärjestelmiä (myös CMS tai julkaisujärjestelmä) verkkosivujen luomisen ja ylläpitämisen avuksi. Ne ovat lisänneet tasaisesti suosiotaan vuosien varrella, ja nykyään maailman suosituimmat sisällönhallintajärjestelmät ovat WordPress, Joomla ja Drupal. (Linode 2017.)

3.1 Sisällönhallintajärjestelmien ominaisuuksia

Yhteistä lähes kaikille sisällönhallintajärjestelmille on, että ne tarjoavat erityisesti kokemattomille käyttäjille mahdollisuuden verkkosivujen luomiseen ja ylläpitämiseen parhaimmillaan ilman minkäänlaista eri web-tekniikoiden tai ohjelmoinnin osaamista. Tästä huolimatta monet ammattilaisina työskentelevät web-kehittäjätkin käyttävät näitä järjestelmiä apunaan töidensä pohjana muun muassa saadakseen julkaistua työnsä nopeammin. (Linode 2017.)

Valtaosassa sisällönhallintajärjestelmistä on mahdollisuus muokata verkkosivuja suoraan visuaalisen käyttöliittymän kautta. Lisäksi verkkosivuston luomisessa voidaan käyttää apuna erilaisia teemoja tai pohjia, joita voidaan myös muokata melko monipuolisesti. (Linode 2017.)

3.2 WordPress

Tämän opinnäytetyön käytännön osuudessa käytettävä sisällönhallintajärjestelmä on WordPress. Se on kehitetty alun perin pääasiassa erilaisten blogien alustaksi, mutta varsinkin nykyisin sitä käytetään monipuolisesti kaikenlaisten verkkosivustojen luomiseen ja kehittämiseen. Sitä pidetään hyvänä varsinkin staattiselle sisällölle, mutta laajan alustaa kehittävän yhteisön ansiosta myös sen dynaamiset ominaisuudet ovat parantuneet. (Linode 2017.) ”WordPress on moderni henkilökohtainen julkaisualusta. Sen painopisteinä ovat esteettisyys, web-standardit ja käytettävyys. WordPress on sekä ilmainen että samalla korvaamaton.” (WordPress n.d.a.)

4 CSS-TYYLIOHJEET

Verkkoselaimet pystyvät näyttämään pelkkää HTML-merkintäkieltä käyttäviä verkkosivuja, minkä lisäksi niiden ulkoasun muokkaaminen eri HTML-elementtien avulla on hyvin rajallisesti mahdollista. Kuitenkin verkkosivut ovat oletusarvoisesti ilman niille annettavia erilaisia muotoiluja hyvin yksinkertaisia ja visuaaliselta ilmeeltään karuja. Pelkän HTML:n avulla luoduissa verkkosivujen ulkoasuissa ei ole myös yleensä juurikaan yksilöllisiä ominaisuuksia ja ne ovat usein käytettävyydeltään melko huonoja. Näitä ongelmia ratkomaan on kehitetty CSS (Cascading Style Sheets) -tyyliohjeet. (Korpela 2008, XII.)

4.1 Historia

Verkkosivujen tyylimääritysten historia ulottuu 90-luvun alkuun. Aluksi dokumenttien rakenteen ja tyylin määrittelyiden erottamista yritettiin kehittää HTML:n pohjalta. Lisäksi useiden eri verkkoselainten kehittäjät ja muut toimijat suunnittelivat omia yksinkertaisia tyyliohjeitaan. Näiden yritysten korvaajaksi suunnitellun CSS:n kehittäminen aloitettiin vuonna 1994 CERNin tutkimuslaitoksessa Håkon Wium Lien ja Bert Bosin toimesta. Parin vuoden työskentelyn jälkeen W3C julkaisi siitä vuoden 1996 lopussa ensimmäisen suosituksensa, CSS1:n. Eri selainten tuki sille oli kuitenkin vielä hyvin vajavaista Internet Explorer 3.0:aa ja suurilta osin Netscape Navigator 4.0:aa lukuun ottamatta. CSS:stä oli kuitenkin käytännössä onnistuttu tekemään uusi tyyliohjeiden standardi. (W3C 2016.)

Kehittäminen jatkui ja virallinen CSS2-suositus annettiin vuonna 1998. Se korjasi runsaasti ensimmäisen version ongelmia sekä lisäsi useita uusia ominaisuuksia. Lisäksi sillä oli jo laajempi tuki Internet Explorerille, Netscape Navigatorille ja Operalle. (W3C 2016.)

CSS 2.1:stä alettiin kehittämään vuonna 1998 lähes heti sen jälkeen, kun CSS 2:lle oli annettu suositus. Nimestään huolimatta siinä enemmänkin pyrittiin karsimaan ja tarkentamaan ominaisuuksia, joita ei ikinä ollut otettu käyttöön edellisessä versiossa. (Korpela 2008, 56) Kehitys oli melko hidasta ja virallinen W3C:n suositus CSS 2.1:sta annettiin vasta vuonna 2011. Suosituksensa CSS3 sai lopulta vuonna 2014. (W3C 2016.)

4.2 Perustietoa CSS-tyyliohjeista

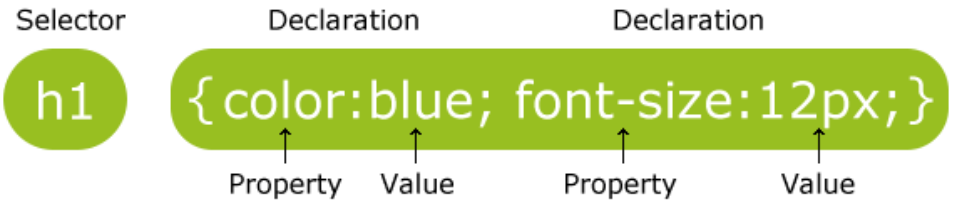
Yleisimmin CSS-tyyliohjeilla muotoillaan HTML-verkkosivuja, joiden muotoilun ja ulkoasun ne määrittävät. Tavallisimpia asioita, joihin CSS:llä halutaan ja voidaan vaikuttaa, ovat muun muassa tekstien fontit, koot ja muu

muotoilu, sivun eri elementtien asemointi ja suhde toisiinsa sekä eri elementtien väritys. HTML-tiedostojen lisäksi CSS:llä on kuitenkin myös mahdollista muotoilla esimerkiksi XML-tiedostoja. (Korpela 2008, 2–3, 49–50.)

Vaikka CSS:ää voidaankin tuottaa myös suoraan HTML-koodin sekaan käyttämällä style-tagia, on varsinkin isompien sivukokonaisuuksien tapauksessa huomattavasti suositeltavampaa linkittää HTML-tiedostoon erillinen tyylitiedosto. Alempana on kuvattu, miten sivusto.css-niminen CSS-tyylitiedosto linkitetään HTML-tiedoston head-osaan.

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

Kuvasta 1 nähdään, että tavallisimmin CSS:n syntaksi koostuu kolmesta eri osasta: selektorista (selector), ominaisuudesta (property) ja arvosta (value). Ominaisuus ja arvo muodostavat deklaraation. Selektorin ja ominaisuuden välissä sekä määrityksen lopussa käytetään aaltosulkua. Puolipisteen eroteltuna on mahdollista määritellä useita eri deklaraatioita samalle selektorille yhdellä kertaa. (Tutorialspoint n.d.a.)



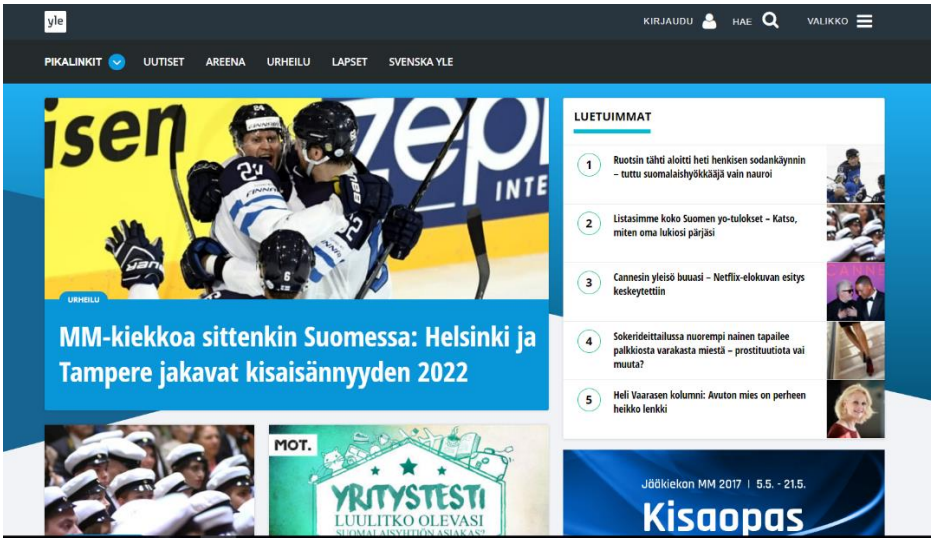
Kuva 1. CSS:n perussyntaksi (W3C n.d.c).

Alla on esitelty yksinkertainen CSS-määritys, joka sijaitsee edellä mainitussa sivusto.css-tiedostossa. Määrityksessä HTML-sivun body-elementin sisällä olevan tekstin väriksi määritellään punainen ja koko sivun taustaväriksi musta.

```
body{

color: red;
background-color: black;
}
```

Kuvista 2 ja 3 nähdään, miten suuri vaikutus CSS- tyyliohjeilla on käytännössä.



Kuva 2. Yle.fi-verkkosivuston etusivu normaalissa näkymässä CSS- ja JavaScript -määritysten ollessa käytössä. (kuvakaappaus.)

Kuva 3. Kuvassa 2 oleva verkkosivusto pelkän HTML:n ollessa käytössä ilman muita määrityksiä. (kuvakaappaus.)

4.3 **CSS3**

CSS3:a ei pidetä määritelmiltään kovinkaan yhdenmukaisena, vaan se sisältää kokoelman vielä monissa erilaisissa kehitysvaiheissa olevia ominaisuuksia. Version kehitys on tapahtunut hiljalleen vuodesta 2001 lähtien, eikä kovinkaan suuria muutoksia ole tapahtunut kerralla. CSS3 pohjautuu CSS 2.1 -versioon, joten se on aiempien suositusten kanssa täysin yhteensopiva. (Korpela 2013, 13−14.)

Itse CSS:n perussyntaksiin ei ole tullut juurikaan muutoksia, mutta täysin uutta on vanhojen ja uusien ominaisuuksien sekoittaminen keskenään ja niiden jako erilaisiin moduuleihin. Uusia ominaisuuksia ja arvoja on tosin tullut melko paljon. Koska CSS3:n ominaisuuksien kehitys on niin erilaisissa vaiheissa, mikään verkkoselain ei kuitenkaan anna täyttä tukea aivan kaikille ominaisuuksille. (Korpela 2013, 13−14.)

W3C (n.d.b) mainitsee muiden muassa seuraavat, lähes kaikilla nykyaikaisilla verkkoselaimilla toimivat moduulit ja ominaisuudet tärkeimmiksi CSS3:ssa:

- Selektorit-moduulin avulla voidaan valita HTML-elementit, joille halutaan antaa tyylimäärityksiä. Esimerkiksi elementti <p> muokkaa kaikkia HTML-elementin leipätekstejä.

- Laatikkomalli-moduuli muodostuu sisällöstä, täytealueesta, reunoista ja marginaalista. Tätä mallia käytetään yleisesti verkkosivun ulkoasun muodostamiseen HTML-elementtien ympärille.

- Taustat ja reunat -moduulin uudet ominaisuudet mahdollistavat erilaisten kuvien käytön elementtien reunoina.

- Kuvien arvot -moduuli, johon on tullut muun muassa kuvien kääntämiseen liittyviä sekä kuvien skaalaamista ja resoluution määrittelemistä parantavia ominaisuuksia.

- Tekstiefektit-moduuli, johon on tullut uusia ominaisuuksia muun muassa määrittelemään, miten sanojen rivitys tapahtuu tietyillä alueilla.

- 2D/3D-muunnokset-moduuli, joka mahdollistaa esimerkiksi eri elementtien pyörittämisen kaksi- tai kolmiulotteisesti.

- Erilaiset animaatiot ovat kehittyneet. Yhä useammat niistä on tehtävissä täysin ilman JavaScriptin tai Flashin käyttöä.

- Monipalstainen sivurakenne mahdollistaa helpommin sanomalehteä muistuttavan palstoista koostuvan sivurakenteen luomisen.

- Käyttöliittymä-moduuli, jonka uusia ominaisuuksia ovat muun muassa monipuolisemmat mahdollisuudet antaa loppukäyttäjän muuttaa itse verkkosivujen elementtien kokoja.

Lisäksi media queryjä voidaan pitää lähes välttämättöminä modernien responsiivisten verkkosivujen luonnissa. Niiden ominaisuudet ovat kehittyneet runsaasti CSS2:n ajoista.

4.4 **CSS-frameworkit**

Myös erilaiset CSS-frameworkit (myös front-end framework) ovat suosittuja web-kehityksen parissa. Ne ovat käytännössä valmiita peruspaketteja frameworkista riippuen sisältäen HTML-, JavaScript- sekä CSS-tyyliohjetiedostoja muun muassa sivuston ulkoasua, käyttöliittymää sekä typografiaa varten. Niiden tarkoituksena on osaltaan nopeuttaa web-kehityksen prosessia tarjoamalla hyviä peruspohjia, joista lähteä liikkeelle verkkosivujen luomisessa. Suosittuja CSS-frameworkeja ovat muun muassa Bootstrap, Materialize ja Semantic-UI. (CSSauthor 2017.)

Yhteistä lähes kaikille frameworkeille on gridien käyttö, joka tarkoittaa verkkosivujen pohjan perustumista ja muodostamista eri kokoisille ruuduille. Tämän tekniikan käyttö osaltaan mahdollistaa ja helpottaa selkeiden ja yksinkertaisten, mutta samalla monipuolisten verkkosivujen kehittämistä. Lisäksi suurimmalle osalle frameworkeista eräs tärkeimmistä ominaisuuksista on verkkosivujen suunnittelu responsiivisiksi. Se tarkoittaa verkkosivujen suunnittelemista toimimaan samalla lailla kaikilla suosituimmilla verkkoselaimilla sekä myös mobiililaitteilla. (Awwwards 2017.)

5 CSS:N ESIPROSESSOINTI

CSS:ää ei ole koskaan tietoisesti haluttu kehittää oikeaksi ohjelmointikieleksi. Siitä puuttuvatkin käytännössä kaikki olio-ohjelmointikielien keskeisimmät ja hyödyllisimpinä pidetyt ominaisuudet. Joitakin hyvin alkeellisilla tasoilla olevia ehdotuksia muun muassa muuttujien lisäämisestä tavalliseen CSS:ään on tosin olemassa. Näiden lisäksi osa puuttuvista ominaisuuksista on joka tapauksessa tälläkin hetkellä toteutettavissa helpommin esimerkiksi joko JavaScriptin, tai PHP:n avulla. Suurimpia edellä mainittuja puutteita korjaamaan on CSS:lle kehitetty erilaisia esiprosessoreja. Niitä ei ole kuitenkaan tehty korvaamaan kokonaan CSS:ää, vaan laajentamaan, tukemaan ja nopeuttamaan sen käyttöä. (Korpela 2013, 21, 35.)

Prabhun (2015, 1) mukaan varsinkin viimeistään CSS3:n mukana tulleet uudet ominaisuudet voivat tehdä laajojen ja monimutkaisten CSS-tyyliohjetiedostojen ylläpitämisestä aikaisempaa huomattavasti vaikeampaa ja enemmän aikaa vievää. Tälloin CSS-esiprosessoreista saatava hyöty korostuu.

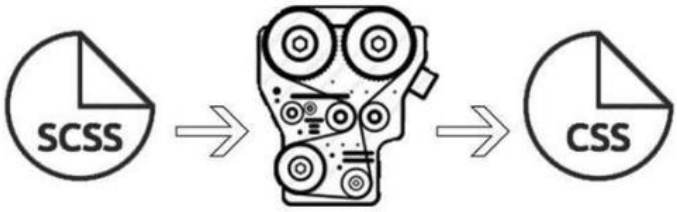
5.1 Esiprosessori

Tietotekniikan kontekstissa esiprosessorilla (myös esikääntäjä) tarkoitetaan ohjelmaa, joka käsittelee sille lähetettyä tiettyä tietoa ja kääntää sen haluttuun muotoon. Perinteisesti niitä on käytetty eri ohjelmointikielissä, kuten C, PHP tai JavaScript. Lisäksi esimerkiksi HTML:lle on saatavilla esiprosessoreita, kuten Pug ja Haml. (Prabhu 2015, 1–2.)

5.2 CSS-esiprosessorien perustoiminta

Prabhun (2015, 1) mukaan CSS-esiprosessorit laajentavat CSS-tyyliohjeita lisäämällä niihin mukaan monipuolisesti nykyaikaisten olio-ohjelmointikielien ominaisuuksia. Esiprosessorien käytön ansiosta on mahdollista noudattaa paremmin niin sanottua DRY (Don’t Repeat Yourself) -oppia, jonka perusperiaatteena on minimoida usein toistettavien tehtävien määrä ylläpidettäessä CSS-tyyliohjeita. CSS-esiprosessorien perustoimintaa voidaan pitää kaikilla esiprosessoreilla samankaltaisena. Niillä on omat tiedostomuotonsa, jonka esiprosessori kääntää tavalliseksi CSS-tyyliohjetiedostoksi. Tyypilliseen työnkulkuun kuuluu, että käyttäjä asettaa joko komentorivin, tai graafisen käyttöliittymän kautta tietyn ohjelman valvomaan käytettävän esiprosessorin omassa tiedostomuodossa olevaa tiedostoa. Varsinainen tiedoston muokkaus suoritetaan esimerkiksi jollakin tekstieditorilla. Kun tiedosto tallennetaan, suorittaa ohjelma muunnoksen CSS:ksi parhaimmillaan täysin automaattisesti.

Kuvassa 4 esiprosessorin omalla syntaksilla luotua tietoa (esimerkissä SCSS) syötetään keskellä olevalle esiprosessorille, joka muuntaa sen tavalliseksi CSS-tyyliohjeeksi. Se on tavallisen verkkoselaimen ymmärrettävissä ja sen avulla selain pystyy määrittelemään verkkosivun muotoilun.



Kuva 4. Havainnekuva CSS-esiprosessorin toiminnasta (Prabhu 2015, 1.)

6 VERTAILTAVIEN CSS-ESIPROSESSORIEN ESITTELY

Tarkempaan esittelyyn ja keskinäiseen vertailuun on valittu kolme suosittua CSS-esiprosessoria: Sass, Less ja Stylus.

6.1 Sass

Sass (Syntactically Awesome Stylesheets) on Hampton Catlinin ja Natalie Weizenbaumin Ruby-ohjelmointikielellä vuonna 2007 kehittämä ja julkaisema avoimen lähdekoodin CSS-esiprosessori. Sassia pidetään maailman ensimmäisenä CSS-esiprosessorina, josta myöhemmät kilpailijat ovat ottaneet vaikutteita. (Prabhu 2015, 5–6.)

6.1.1 Syntaksit

Sassin varsinaisesta kielestä käytetään nimitystä SassScript, josta on kehitetty kaksi jonkin verran toisistaan eroavaa syntaksia. Alun perin vuonna 2010 julkaistu uudempi, nykyään hyvin laajassa käytössä oleva syntaksi on nimeltään SCSS (Sassy CSS). Käytännössä sitä voidaan pitää vain eräänlaisena laajennuksena CSS:n syntaksille ja tavalliset CSS-tyylitiedostot ovatkin täysin yhteensopivia sen kanssa. Lisäksi se osaa tunnistaa automaattisesti eri verkkoselainten selainkohtaisia CSS-sääntöjä. SCSS-tiedostot ovat .scss-päätteisiä. Sassin kehittäjät itse suosittelevat nykyisin käyttämään ensisijaisesti tätä syntaksia. (Sass-lang 2016.)

Alla oleva esimerkki on luotu käyttäen SCSS-syntaksia. Siinä on määritelty muuttujien avulla body-elementille ja alapalkki-diville omat fonttinsa sekä niiden värit ja koot. Muuttujat ja niiden toiminta on esitelty tarkemmin luvussa 7.

```
$pääfontti: Calibri, Arial;
$pääfonttiväri: #036;
$pääfonttikoko: 12px;
$sivufontti: Times New Roman, Arial;
$sivufonttiväri: #333;
$sivufonttikoko: 10px;

body {
  font: $pääfontti;
  color: $pääfonttiväri;
  p: $pääfonttikoko;
}

#alapalkki {
  font: $sivufontti;
  color: $sivufonttiväri;
  p: $sivufonttikoko;
}
```

Toinen käytössä oleva syntaksi on alkuperäinen Sass-syntaksi, josta käytetään yleensä nimeä ”sisennetty syntaksi”, tai vaihtoehtoisesti esiprosessorin omaa nimeä. Se eroaa suoraviivaisuudessaan ulkoasultaan tavallisesta CSS:stä hieman enemmän kuin SCSS. Puolipisteet eivät ole lainkaan käytössä, vaan ominaisuudet tulee joka tilanteessa jaotella omille riveilleen. Lisäksi aaltosulut on korvattu sisennyksillä. Syntaksin mukaista koodia sisältävät tiedostot nimetään .sass-päätteisesti. (Sass-lang 2016.)

Alla on edellä mainittu SCSS-esimerkki muutettuna sisennetyn syntaksin muotoiseksi:

```
$pääfontti: Calibri, Arial
$pääfonttiväri: #036
$pääfonttikoko: 12px
$sivufontti: Times New Roman, Arial
$sivufonttiväri: #333
$sivufonttikoko: 10px

body
  font: $pääfontti
  color: $pääfonttiväri
  p: $pääfonttikoko


#alapalkki
  font: $sivufontti
  color: $sivufonttiväri
  p: $sivufonttikoko
```

Esimerkistä nähdään, että sisennetty syntaksi näyttää puuttuvia aaltosulkuja ja puolipisteitä lukuun ottamatta esimerkissä täysin samalta, kuin SCSS.

Lopulta tavalliseksi CSS:ksi käännettynä edellä mainittu koodinpätkä näyttää seuraavalta:

```
/* line 8, tyylit.sass */
body {
  font: Calibri, Arial;
  color: #003366;
  p: 12px;
}

/* line 14, tyylit.sass */
#alapalkki {
  font: Times New Roman, Arial;
  color: #333333;
  p: 10px;
}
```

Esiprosessoinnin tekemä ohjelma on luonut koodia kääntäessään automaattisesti kommentit, joissa on kerrottu millä rivillä kyseinen määritys sijaitsee .sass-tiedostossa.

6.1.2 Sass-frameworkit

Hieman CSS-frameworkien tapaan myös itseään Sass-frameworkeiksi kutsuvia ohjelmia on tarjolla runsaasti. Yleisesti ottaen niitä voidaan pitää eräänlaisina työkalupakkeina tai kirjastoina, jotka sisältävät erilaisia valmiita koodipaketteja käytettäväksi Sassin kanssa. Tunnetuimpina Sass-frameworkeina voidaan pitää Compassia ja Bourbonia. (Giraudel 2014.)

6.2 **Less**

Less.js (Leaner Style Sheets) on Alexis Sellierin vuonna 2009 kehittämä CSS-esiprosessori, joka on alun perin luotu Ruby-ohjelmointikielellä, kuten Sass. Uudemmat versiot on kuitenkin luotu JavaScriptillä. Syntaksiltaan Less muistuttaa hyvin paljon SCSS:ää, joten puolipisteiden ja aaltosulkujen käyttö on pakollista joka tilanteessa. Myös Lessin syntaksi on automaattisesti puhdasta CSS:ää. (Lesscss n.d.)

Alla olevassa yksinkertaisessa esimerkissä on luotu kaksi muuttujaa, jotka määrittelevät verkkosivun taustavärin sekä tekstivärin.

```
@taustavari: #grey;
@tekstivari: #111111;

body{
    background-color: @taustavari;
}

p {
    color: @tekstivari;
}
```

Esimerkin perusteella eroa SCSS-syntaksiin ei käytännössä ole eri muuttujan julistavan merkin (@) lisäksi.

6.3 **Stylus**

Stylus on TJ Holowaychukin vuonna 2011 kehittämä CSS-esiprosessori. Kuten Less, se on luotu JavaScriptiä käyttäen. Kuten SCSS ja Less, sen tuottama koodi on suoraan täysin puhdasta CSS:ää. (Giraudel 2015.)

Stylusin syntaksi on todella salliva. Käytännössä on esimerkiksi mahdollista käyttää saman tyyliohjeen sisällä sekä sisennettyä syntaksia ilman päätteitä että päättää määritykset puolipisteillä. Stylusin tiedostot ovat .styl-päätteisiä. (Giraudel 2015.)

Alla olevassa esimerkissä on demonstroitu Stylusin syntaksin sallivuutta. Siitä voidaan havaita, että jopa yhden selektorin sisällä olevien määritysten tapoja voidaan vaihdella. Aaltosulkuja käytettäessä on muistettava myös sulkeva sulku.

```
fonttikoko = 11px
$fontti = Calibri, Arial;

body {
  font fontti
  font-size = fonttikoko;
}
```

6.4 **Esiprosessorien asennus- ja käyttötavat**

Perusperiaatteiltaan eri CSS-esiprosessorien asennusprosessit ja käyttötavat muistuttavat suurilta osin toisiaan lukuun ottamatta pieniä yksityiskohtia.

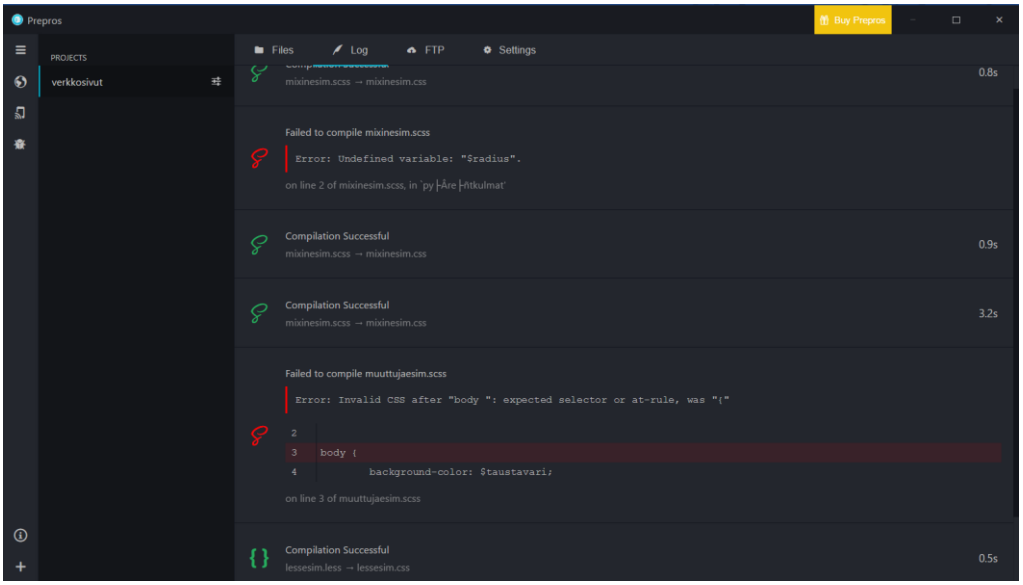
6.4.1 Prepros-käyttöliittymä

Jokaista tässä opinnäytetyössä käsiteltävää CSS-esiprosessoria voidaan käyttää lukuisilla erilaisilla kolmannen osapuolen graafisilla kääntäjätyökaluilla. Kaikkein kattavimmilla ominaisuuksilla varustetut ohjelmat ovat pääosin maksullisia, mutta yleisesti ottaen kaikilla niistä on mahdollista vahtia reaaliaikaisesti muutoksia tiedostoissa ja suorittaa koodin kääntäminen lähes reaaliaikaisesti. Vaatimuksena oli löytää ilmainen ohjelma, jossa olisi ominaisuutena esiprosessori Sassille, Lessille ja Stylusille.

Usean ilmaisen ohjelman kokeilemisen jälkeen päädyttiin käyttämään ohjelmaa nimeltään Prepros. Kyseessä on periaatteessa maksullinen ohjelma, vaikka sen käyttäminen ominaisuuksiltaan täysin vastaavana trial-versiona onkin mahdollista rajoittamattoman ajan. Prepros on Windows- ja Linux-käyttäjille kehitetty versio Macin ominaisuuksiltaan vastaavasta ohjelmasta CodeKit. Sillä on mahdollista käsitellä myös lukuisia ohjelmointikieliä, kuten JavaScriptiä. Lisäksi siinä on runsaasti muita ominaisuuksia, kuten verkkoselaimen automaattinen päivittäminen muutosten teon jälkeen. (Prepros 2017.)

Preprosin peruskäyttö on yksinkertaista. Aluksi määritellään tietokoneelta kansio, josta sen halutaan valvovan tiedostojen muutoksia. Tämän jälkeen käynnistetään valvonta, jonka jälkeen voidaan aloittaa esimerkiksi .sass-tiedoston työstäminen jollakin tekstieditorilla.

Kuvassa 5 on luotu projekti nimeltään Verkkosivut, jolle määriteltyä tiedostokansiota Prepros valvoo. Se havaitsee automaattisesti kansion tiedostoihin tehdyt muutokset ja suorittaa niiden perusteella kääntämisen CSS:ksi. Prepros myös ilmoittaa lokissaan mahdollisista virheistä sekä niiden syistä ja sijainnista.



Kuva 5. Kuvakaappaus Prepros-ohjelmasta käynnissä ja valvomassa muutoksia tiedostoissa.

6.4.2 Sassin asennus ja käyttö komentorivillä

Erillisten kolmannen osapuolen graafisten käyttöliittymien lisäksi Sass voidaan ottaa käyttöön myös suoraan komentoriviltä. Sitä varten vaaditaan Windows- ja Linux-käyttöjärjestelmille erikseen Ruby-ohjelmointikielen ympäristön asentaminen tietokoneelle. Mac-koneilta se taas löytyy yleensä jo valmiiksi asennettuna. Rubyn asennus Windows 10 -tietokoneelle on yksinkertaista, ja se tapahtuu oman asennusohjelmansa kautta, jonka voi ladata ilmaiseksi internetistä. Kun asennusohjelma on suorittanut asennuksen loppuun, avataan komentorivi ja annetaan komento ”gem install sass”, jolloin Ruby asentaa Sassin käyttöön tarvittavat tiedostot. (Sass-lang n.d.a.) Kuvassa 6 Sass on juuri asennettu.

```
C:\Users\v_val>gem install sass
Fetching: sass-3.4.23.gem (100%)
Successfully installed sass-3.4.23
Parsing documentation for sass-3.4.23
Installing ri documentation for sass-3.4.23
Done installing documentation for sass after 16 seconds
1 gem installed

C:\Users\v_val>
```

Kuva 6. Sass asennettu onnistuneesti komentorivin kautta. (kuvakaappaus)

Tämän jälkeen voidaan asettaa Sass valvomaan muutoksia halutuissa tiedostoissa tai koko kansiossa.

```
sass --watch app/sass:c:\verkkosivut
```

6.4.3 Lessin asennus ja käyttö

Useimpien graafisten käyttöliittymien lisäksi Lessin asentamiseen on kaksi eri vaihtoehtoa: komentoriviä käyttävä palvelinversio, joka on toteutettu Node.js:llä sekä suoraan uusimmissa verkkoselaimissa toimiva versio.

Node.js on JavaScriptilla tehty palvelinpuolen avoimen lähdekoodin kehitysalusta. Sen avulla voidaan ajaa JavaScriptia palvelimen päässä sekä toteuttaa erilaisia JavaScript-sovelluksia ja verkkosivuja. Node voidaan ladata ja asentaa tietokoneelle normaalisti lataamalla internetistä ja asentamalla se asennusohjelman avulla. Nodeen liittyy tiiviisti myös NPM, joka on sen eräänlainen pakettienhallintajärjestelmä. Komentorivin avulla voidaan asentaa näitä paketteja/moduuleja käyttöön. (Tutorialspoint n.d.b.) Lessin käyttöä varten asennetaan sen paketti seuraavalla komennolla:

```
$ npm install less -g
```

Asennuksen jälkeen on mahdollista asettaa Less valvomaan haluttua kansiota komennolla:

```
lessc tyylit.less tyylit.css
```

Lessia on mahdollista kääntää myös suoraan verkkoselaimen avulla. Koska Less on tehty JavaScriptilla, käyttö onnistuu yksinkertaisesti vain viittaamalla HTML:llä JavaScript- sekä .less -tiedostoihin. Esiprosessori yrittää kääntää koodin joka kerta, kun verkkoselainta päivitetään uudelleen. Tästä syystä tapaa kuitenkin suositellaan vain nopeaan testaukseen, ja tuotantokäytössä sitä ei tulisi koskaan käyttää sen hitauden ja mahdollisten toimintaongelmien vuoksi. (Lesscss n.d.)

6.4.4 Stylusin asennus ja käyttö

Kuten muitakin CSS-esiprosessoreja, myös Stylusta voidaan käyttää useilla graafisilla käyttöliittymillä. Stylusta tukevien ohjelmien määrä on kuitenkin hieman pienempi, kuin kahdella muulla esitellyllä. Niiden lisäksi sitä voidaan käyttää myös Noden komentorivin avulla, jossa sille asennetaan oma NPM-paketti. Muutoin sen peruskäyttö tapahtuu komentorivillä täysin Lessiä vastaavasti.

7 ESIPROSESSORIEN YLEISET OMINAISUUDET

Alla olevissa kappaleissa tarkastellaan CSS-esiprosessorien yleisimpiä ominaisuuksia valtaosin vertailtavien esiprosessorien näkökulmasta sekä miten niitä voisi hyödyntää CSS:n käsittelyssä. Esimerkeissä on käytetty pääasiassa Sassin SCSS:ää, joka on esitelty tarkemmin luvussa 6.

7.1 Muuttujat

Muuttujien (variables) avulla voidaan tallentaa yhteen paikkaan tietoa, jota halutaan käyttää myöhemmin uudelleen saman tyyliohjeen sisällä. Tyypillisiä käyttökohteita ovat eri CSS:n ominaisuudet, kuten eri värit ja fontit, mutta tallentaa voidaan käytännössä kaikenlaisia CSS:n arvoja myöhempään käyttöön. Esimerkiksi Sassin molemmissa syntakseissa muuttuja julistetaan $-merkillä. (Sass-lang n.d.b). Lessissä taas käytetään @-merkkiä. Stylusissa tunnisteen käyttö ei ole pakollista, mutta siinä voidaan käyttää $-merkkiä.

Alla olevassa esimerkissä luodaan SCSS:n avulla muuttuja $taustavari, jossa määrätään verkkosivun taustaväriksi punainen. Tämä muuttuja on kutsuttavissa ja käytettävissä nimellään missä tahansa tyyliohjeen sisällä.

```
$taustavari: red;

body {

  background-color: $taustavari;
}
```

7.2 Mixinit ja funktiot

Mixinien avulla voidaan luoda ja ryhmitellä mitä tahansa CSS-sääntöjä uudelleen käytettäviksi ryhmiksi. Tämä voi olla kätevä ominaisuus esimerkiksi ryhmiteltäessä eri verkkoselainten selainkohtaisia CSS-asetuksia yhteisiksi ryhmiksi. SCSS:ssä mixin julistetaan @mixin-komennolla ja kutsutaan @include-komennolla. (Sass-lang n.d.b). Mixinien käytössä tulee kuitenkin olla tarkkana, sillä niiden ylikäyttö voi pahimmillaan tehdä tyylitiedostoista sekavia ja kasvattaa niiden kokoa huomattavasti. (Prabhu 2015, 11).

Alla olevassa esimerkissä luodaan SCSS:n avulla mixin @pyöreätkulmat, jossa määritellään CSS:n border-radius-ominaisuus käyttöön body-elementin sisällä. Kyseisellä ominaisuudella voidaan luoda elementille pyöreät reunat. Mixinissä on mukana omat määrityksensä eri verkkoselaimia ja niiden asetuksia varten.

```
@mixin pyöreätkulmat($säde:5px) {
-webkit-border-radius: $säde;
-moz-border-radius: $säde;
-ms-border-radius: $säde;
border-radius: $säde;
}

 body {@include pyöreätkulmat(12px);}
```

Funktiot muistuttavat melko paljon mixineitä, ja suurin ero toiminnallisuudessa tulee siitä, että funktioilla voidaan suorittaa erilaisia laskutoimituksia, joiden arvoja voidaan käyttää missä tahansa kohtaa tyyliohjetta. Ne voivat olla mitä tahansa datatyyppiä. (Wendell 2011).

Esimerkissä luodaan kaksi eri funktiota, lasku- ja suhdefunktio.

```
@function laskufunktio($numero1, $numero2){
@return $numero1 + $numero2
}

@function suhde($vasen, $oikea) {
@return ($vasen / $oikea) * 100%;
}

body {
padding: laskufunktio(20px, 8px);
margin: suhde(25px, 60px);
}
```

CSS:ksi käännettynä funktio on suorittanut laskutoimitukset:

```
body {
  padding: 28px;
  margin: 41,66667%;

}
```

7.3 **Nesting**

Nesting mahdollistaa CSS:n selektorien sisäyttämisen hieman HTML:n elementtien tapaan. Esimerkiksi kaikki header-elementille tehdyt säädöt voidaan laittaa luokan #header sisään. Liiallista selektorien sisäyttämistä toisiinsa tulee kuitenkin välttää, koska koodista saattaa hyvin helposti tulla sekavaa ja vaikeasti ylläpidettävää. (Sass-lang n.d.b).

Alla olevassa esimerkissä body-elementin sisään on luotu määritys fonttikoolle. Lisäksi sen sisään on tehty määritys leipätekstin (p) värille. Aaltosulkeet tulevat vasta loppuun.

```
body {
    font-size: 11px;
    p {
       color: black;
    }
}
```

Normaaliksi CSS:ksi käännettynä yllä oleva esimerkki on seuraavanlainen. Esiprosessori on erottanut leipätekstin värin omaksi selektorikseen.

```
body {
  font-size: 11px;
}

body p {
  color: black;
}
```

7.4 **Extend**

Komento @extend mahdollistaa CSS:n selektorien ominaisuuksien jakamisen muiden kesken. Kun halutaan samat ominaisuudet useammalle selektorille, tämä tekniikka on hyödyllinen. Alla olevassa esimerkissä luodaan container-selektorista extend-määritys. (sass-lang n.d.b).

```
.container {
color: black;
border: 1px solid black;
}

.alapalkki {
@extend .container;
background-color: red;
}
```

CSS:ksi käännettynä containerin ja alapalkin yhteiset ominaisuudet ovat yhdistetty.

```
.container, .alapalkki {
color: black;
border: 1px solid black;
}

.alapalkki {
background-color: red;
}
```

7.5 **Operaatiot**

Operaatioiden avulla voidaan muun muassa suorittaa erilaisia laskutoimituksia. Nämä voivat olla käteviä esimerkiksi grid-pohjaisten verkkosivujen pohjien koon laskemisessa. Omia erilaisia operaatioita on muun muassa numeroille, väreille ja listauksille. (Sass-lang n.d.b).

Alla olevassa esimerkissä käytetään vertailu-operaattoria <=. Esimerkki sisältää myös @if-funktion käyttöä; kun muuttujan $padding arvo on 20 px tai sen alle, arvo pysyy muuttumattomana. Jos sen arvo taas on yli 20 px, se puolitetaan.

```
$padding: 22px;

h1 {
@if($padding <= 20px) {
padding: $padding;

} @else {
 padding: $padding / 2;
}
}
```

CSS:ksi käännettynä esimerkki näyttää seuraavalta.

```
h1 {
  padding: 11px;
}
```

7.6 **Import**

Import (tuonnin) avulla voidaan pienentää CSS-tiedostoa pienempiin osiin. Tämä mahdollisuus on olemassa jo tavallisessa CSS:kin, mutta esiprosessoreiden siihen tuoma lisäarvo on, ettei mukana lähetetä ylimääräistä pyyntöä palvelimelle. Sassissa käytetään komentoa @import. (Sass-lang n.d.b).

8 ESIPROSESSORIEN VERTAILU

Esiprosessorien keskinäisessä vertailussa tarkastellaan niiden ominaisuuksia, käyttöönoton helppoutta ja sen vaihtoehtoja, sekä käytettävyyttä ja syntaksin oppimiskäyrää.

8.1 Syntaksien tärkeimmät ominaisuudet

Taulukossa 1 on lueteltu esiprosessorien syntaksien tärkeimpiä ominaisuuksia sekä sitä, miten niitä voidaan käyttää kyseisellä esiprosessorilla. Nämä ominaisuudet on esitelty tarkemmin luvussa 5.

Taulukko 1. Yhteenveto esiprosessorien eri ominaisuuksista (CSS PRE 2016).

Ominaisuus	Sass	Less	Stylus
Syntaksissa mahdollisuus olla käyttämättä aaltosulkuja ja puolipisteitä	Kyllä (sisennetyssä syntaksissa)	Ei	Kyllä
Mahdollisuus asettaa muuttujia ja käyttää niitä myöhemmin tiedostossa	Kyllä, julistetaan $-merkillä	Kyllä, julistetaan @-merkillä	Kyllä, voidaan julistaa $-merkillä
Mahdollisuus käyttää muuttujia selektorin, ominaisuuden tai arvon osana	Kyllä	Kyllä	Kyllä
Muuttujaa ei tarvitse julistaa ennen käyttöä	Ei	Kyllä	Ei
Selektoreja voidaan sisäyttää nestingin avulla	Kyllä	Ei	Kyllä
Mixinit	Kyllä, julistetaan @-merkillä, kutsutaan @include-komennolla	Kyllä, kutsutaan halutun luokan nimeä	Kyllä, kutsutaan halutun luokan nimeä
Import-ominaisuus	Kyllä	Kyllä	Kyllä
Extend-ominaisuus	Kyllä	Kyllä	Kyllä

Taulukosta nähdään, että tärkeimmiltä ominaisuuksiltaan vertailussa olevat esiprosessorit ovat melko samankaltaisia. Pieniä eroja kuitenkin löytyy siinä, mitä merkkejä käytetään ominaisuuksien julistamiseen. Ainoastaan Less erottuu ominaisuuksiltaan hieman suppeampana.

8.2 Käyttöönotto ja käytettävyys

Taulukko 2:een on kerätty esiprosessorien erilaiset käyttötavat.

Taulukko 2. Yhteenveto esiprosessorien eri käyttötavoista.

Ominaisuus	**Sass**	**Less**	**Stylus**
Mahdollisuus esiprosessorin käyttöön komentorivillä	Kyllä	Kyllä	Kyllä
Mahdollisuus esiprosessorin käyttöön erilaisilla graafisilla käyttöliittymillä	Kyllä	Kyllä	Kyllä
Mahdollisuus esiprosessorin käyttöön suoraan verkkoselaimessa	Ei	Kyllä	Ei

8.3 Vertailun tulokset

Vertailtujen asioiden perusteella voidaan päätellä, etteivät erot esiprosessoreiden välillä ole valtavia. Tuloksista kuitenkin nähdään, että kokonaisuudessaan Sassia voidaan pitää verratuista ominaisuuksiltaan monipuolisimpana vaihtoehtona. Lisäksi sitä varten kehitetyt eri frameworkit ja muut apuvälineet tekevät siitä todella varteenotettavan vaihtoehdon.

Less on nimensäkin mukaisesti hieman yksinkertaisempi ja minimalistisempi vaihtoehto, jonka toisaalta voi uskoa houkuttelevan omaa käyttäjäkuntaansa. Toisaalta pääominaisuuksiltaan se jää hieman muita vertailtuja heikommaksi.

Stylusin pääominaisuudet vastaavat käytännössä Sassia. Stylus saattaa houkutella sellaisia ihmisiä, jotka eivät halua liikaa miettiä kirjoittavatko he oikeanlaista koodia. Vapaan syntaksin takia onkin tärkeää, että mahdollisesti useamman henkilön käsitellessä samoja Stylus-tiedostoja niissä säilyy yhtenäisyys ja selkeys.

Syntaksin oppimisen kannalta kaan eroja ei kovin paljoa ole. Perustaidoilla CSS:stä pääsee jo varsin nopeasti varsinkin SCSS:n ja Lessin sisälle. Lisäksi olio-ohjelmointikokemuksesta saattaa olla hyötyä.

9 ESIPROSESSORIT JA WORDPRESS

Luvussa tarkastellaan, millä tavoin CSS-esiprosessorit voisivat olla hyödynnettävissä osana WordPress-teeman muokkausta. Lisäksi tutkitaan, millaisia lisäosia WordPressissa on tarjolla aiheeseen liittyen. Esivalmisteluina WordPress-alusta on asennettu paikallisesti omalle tietokoneelle käyttäen XAMPP-palvelinohjelmistoa ja MySQL-tietokantoja. Lisäksi muut tarvittavat säädöt on tehty etukäteen valmiiksi alustan käyttöönottoa varten.

WordPress-järjestelmässä käyttäjän ei ole itse pakko käsitellä CSS:ää ollenkaan, vaan mahdollista on myös käyttää pelkästään muiden tekemiä ja jakamia teemoja. Niissä ulkoasumääritykset on tehty valmiiksi. CSS:ää on kuitenkin mahdollista muokata käsin käyttäjän niin halutessa. Lisäksi olemassa on lisäosia, joiden avulla voidaan tehdä halutut CSS-deklaraatiot käyttäen apuna esimerkiksi erilaisia liukusäätimiä ja valikoita. Annetuista tiedoista lisäosa muodostaa varsinaisen CSS-tyyliohjeen.

9.1 Lisäosat ja muut vaihtoehdot

WordPressille on olemassa runsaasti erilaisia, yleensä kolmansien osapuolten kehittämiä lisäosia, joiden avulla CSS:n esiprosessointi voidaan suorittaa suoraan WordPressin sisällä käyttämättä komentoriviä tai erillisiä ohjelmia. Niidenkin käyttö on tosin mahdollista. Esimerkiksi Prepros voidaan normaaliin tapaan määritellä valvomaan tiedostoja ja suorittamaan esiprosessointi automaattisesti joka kerta tallennettaessa.

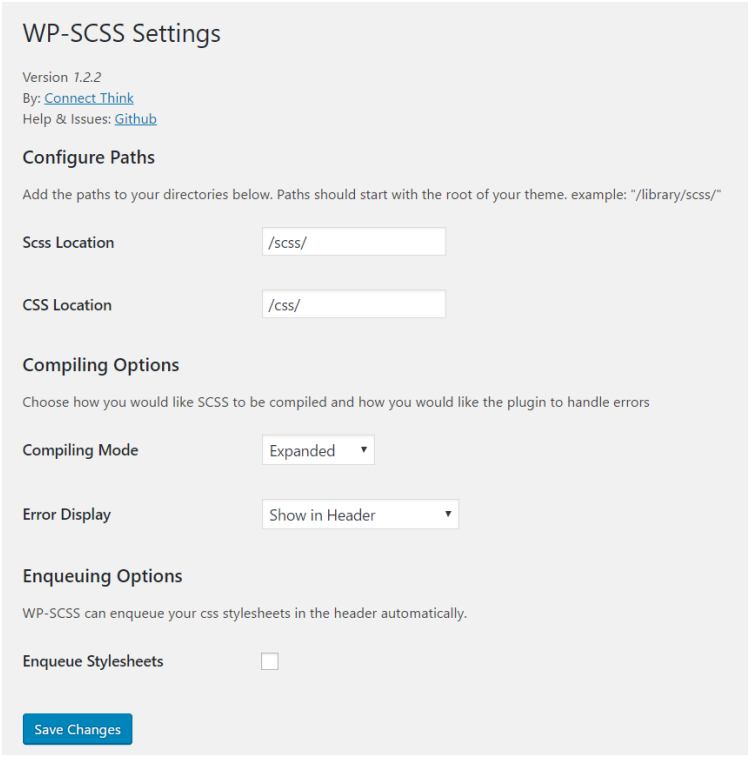
WordPressin lisäosakirjastoa tutkiessa löytyi joitakin potentiaalisia lisäosia testattavaksi. Varsinkin Sassille niitä oli saatavilla useampia. Eräs suosituimmista oli WP-SCSS, joka nimensä mukaisesti toimii SCSS-esiprosessorina. Myös Lessille löytyi oma lisäosa WP-LESS. Stylusille tarjonta oli heikompaa. Sille löytyi vain yksi lisäosa, jota ei ole päivitetty useampaan vuoteen. Sillä ei myös ole juurikaan aktiivisia käyttäjiä.

Lisäksi ohjemateriaalin (WordPress n.d.b.) mukaan suoraan verkkoselaimessa käytettävästä versiosta (wordpress.com) löytyvät sekä Sass (SCSS) että Less -tuki sisäänrakennettuina. Toisaalta kyseisessä versiossa ei ole ollenkaan mahdollisuutta lisäosien käyttöön. Ohjeen mukaan käyttäjän on mahdollista suorittaa esiprosessointi vain kirjoittamalla koodi oikeassa syntaksissa CSS:n muokkausikkunaan ja valitsemalla haluttu esiprosessori valikosta. WordPressin selainversiota ei kuitenkaan tässä opinnäytetyössä käsitellä tarkemmin.

9.2 Käyttö valmiin teeman kanssa

Lisäosien toimintaa tutkimista varten on valittu esimerkiksi WP-SCSS. Teemaksi on valittu WordPressin teemakirjastosta Twenty Sixteen -niminen valmis teema. Se on yksi järjestelmän oletusteemoista. Seuraavaksi lisäosa WP-SCSS asennetaan normaalisti ja aktivoidaan se käyttöön.

Lisäosan asetusikkunassa (kuva 7) asetetaan polut sekä SCSS-tiedoston että CSS:n sijainnille teeman juuressa. Jos CSS-tiedostoa ei ole luotu valmiiksi lisäosa luo sen automaattisesti, kun esiprosessoria käytetään ensimmäisen kerran. Lisäosa tarjoaa myös muita asetuksia. Esimerkiksi kohdassa Compiling Mode voidaan säätää, miten esiprosessori käsittelee syntyvän koodin. Valitaan Expanded, jolloin CSS-tiedostoon tuotettu koodi on kaikkein väljimmässä muodossa, eli muun muassa jokainen deklaraatio on omalla rivillään. Valita voisi myös esimerkiksi asetuksen Compressed, jolloin koko CSS-tiedoston sisältö puristetaan yhteen riviin ilman välilyöntejä. Tämä saattaa parantaa varsinkin isojen tiedostojen toimivuutta, tosin luettavuuden ja selkeyden kustannuksella. Lisäksi mahdolliset virheet voidaan asettaa näytettäväksi sivun yläreunan otsikko-osassa, sillä tässä tapauksessa käytetään vain paikallista testisivua.



Kuva 7. WP-SCSS-lisäosan asetusikkuna. (kuvakaappaus).

Seuraavaksi tehtiin tekstieditorilla teemaan SCSS:n avulla pieniä muutoksia, kuten muutettiin taustan väriä. Kuvassa 8 WP-SCSS on havainnut tallennuksessa, että yksi muuttujista on jäänyt määrittelemättä. Lisäosa ilmoittaa siitä sivun header-osiossa. Osa muutoksista, kuten taustavärin vaihtaminen on kuitenkin tehty.

Kuva 8. Muokattu WordPress-teema lisäosan virheilmoituksineen (kuvakaappaus.)

Kaiken kaikkiaan kokeiltu lisäosa tarjoaa melko monipuoliset ominaisuudet yksinkertaisessa paketissa. Toisaalta jos käyttäjällä on käytössä jo muutenkin jokin vähintään samanlaiset ominaisuudet tarjoava ohjelma, niin suurta tarvetta tämän lisäosan käytölle on vaikea nähdä. Näitä voivat olla esimerkiksi jokin graafinen käyttöliittymä tai WordPressin tukema Sass-framework (kuten Compass).

10 YHTEENVETO

Onnistuin mielestäni vastaamaan tyydyttävästi opinnäytetyölle asetettuihin tutkimuskysymyksiin. Toisaalta ainakin WordPressia tutkiessa aika alkoi käydä jo vähiin, ja siitä olisi voinut luultavasti saada hieman enemmänkin irti. Joka tapauksessa uskoisin, että lukija saa tämän työn perusteella hyvän yleiskuvan siitä, mitä CSS-esiprosessointi tarkoittaa ja mitä eri vaihtoehtoja se tarjoaa. Työstä käy ilmi, etteivät ainakaan suosituimpiin kuuluvien CSS-esiprosessorien erot ole kovinkaan merkittäviä. Ne syntyvät lähinnä eri ominaisuuksien ja toiminnallisuuden toteuttamisesta jonkin verran erilaisin vivahtein.

CSS-esiprosessorit tuovat mielestäni kohtalaisen pienen syntaksin ja ominaisuuksien opiskelun jälkeen varsinkin kokeneelle kehittäjälle selkeää lisäarvoa web-kehittämisen ja CSS:n käsittelyn prosessiin jo melko pienissäkin projekteissa. Lukuisten erilaisten vaihtoehtojen, ja varsinkin graafisten käyttöliittymien ansiosta niiden käyttöönottokin on todella sujuvaa. Ottamalla esiprosessori käyttöön heti verkkosivustokokonaisuuden luonnin alussa voidaan myöhemmin säästää aikaa ja vaivaa. Uskoisin, että hyöty korostuu edelleen varsinkin suurissa kehitysprojekteissa, jossa tavallista CSS-koodia voi helposti olla tuhansia rivejä eri tiedostoissa. Tämän opinnäytetyön rajallisissa puitteissa tällaista tilannetta ei kuitenkaan päästy kunnolla demonstroimaan. Toisaalta pelkän yksittäisen ja pienen, vain staattista sisältöä sisältävän verkkosivun tyyliohjeiden ylläpitoa varten esiprosessorin asennus, käyttöönotto ja syntaksin opettelu alusta alkaen ei välttämättä maksa vaivaa.

Olen jo pitkään ollut yleisesti kiinnostunut web-kehityksestä, mutta viime vuosina tietämyksen ja taitojen kunnolliseen ylläpitämiseen ei ole ollut aikaa. CSS-esiprosessoreista minulla ei ollut minkäänlaista aikaisempaa kokemusta. Valitsinkin tämän aiheen, koska se kuulosti teemansa perusteella kiinnostavalta. Opinnäytetyön tekemisen aikana opinkin paljon lisää käsitellyistä aiheista. Työn varsinaisen pääaiheen lisäksi sain monista muista tekniikoista paljon lisätietoa, hyvää kertausta ja päivitystä aiemmin oppimiini asioihin. Tulevaisuudessa pyrin varmasti soveltamaan työn tekemisen aikana oppimiani asioita käytännössä niin paljon kuin mahdollista.

Opinnäytetyön kehittämistä voisi jatkaa käytännön osalta esimerkiksi suunnittelemalla ja toteuttamalla jonkin oikean (isohkon) verkkosivuston muutoksen jotain CSS-esiprosessoria käyttäväksi. Lisäksi voisi käyttää vertailumatriisia oikean vaihtoehdon löytämiseksi yritykselle.

On täysin mahdollista, että tulevaisuudessa CSS-tyyliohjeiden uudet versiot tulevat lopulta tekemään erilliset CSS-esiprosessorit turhiksi. Toisaalta mitään kovinkaan vakavasti otettavia ja realistisia viitteitä tällaisesta kehityksestä ei tällä hetkellä ole.

LÄHDELUETTELO

Awwwards (2017). What Are Frameworks? 22 Best Responsive CSS Frameworks For Web Design. Haettu 19.4.2017 osoitteesta https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

CSSauthor (2017). CSS Frameworks. Haettu 19.4.2017 osoitteesta http://www.cssauthor.com/css-frameworks/

CSS PRE (2016). Comparison of Support For Selected Features. Haettu 4.5.2017 osoitteesta https://csspre.com/compare/

Giraudel, H. (2014). Sass Frameworks: Compass or Bourbon? Haettu 26.5.2017 osoitteesta https://www.sitepoint.com/compass-or-bourbon-sass-frameworks/

Giraudel, H. (2015). Getting To Know Stylus. Haettu 20.4.2017 osoitteesta https://www.sitepoint.com/getting-to-know-stylus/

Harris, A. (2014). *HTML5 & CSS3 All-in-One For Dummies.* Hoboken*:* John Wiley & Sons (e-kirjaversio).

Korpela, J. (2008). *CSS verkkosivujen muotoilussa.* Jyväskylä: Docendo.

Korpela, J. (2013). *CSS3: Uudet mahdollisuudet.* Jyväskylä: Docendo.

Korpela, J. (2014). *HTML5-käsikirja.* Jyväskylä: Docendo.

Lesscss (n.d.a). About Less. Haettu 6.4.2017 osoitteesta http://lesscss.org/about/

Lesscss (n.d.b). Usage. Haettu 26.5.2017 osoitteesta http://lesscss.org/usage/

Linode (2017). Content Management Systems: an Overview. Haettu 8.5.2017 osoitteesta https://www.linode.com/docs/websites/cms/cms-overview

Prabhu, A. (2015). *Beginning CSS Preprocessors: With SASS, Compass.js and Less.js*. New York City: Apress (e-kirjaversio).

Prepros (2017). Prepros. Haettu 24.5.2017 osoitteesta https://prepros.io/

Sass-lang (n.d.a). Install Sass. Haettu 3.5.2017 osoitteesta
http://sass-lang.com/install

Sass-lang (n.d.b). Sass Guide. Haettu 4.5.2017 osoitteesta
http://sass-lang.com/guide

Sass-lang (2016). Sass Syntax. Haettu 12.4.2017 osoitteesta
http://sass-lang.com/documentation/file.SASS_REFERENCE.html

Sitepoint (2015). Sass Basics: The Function Directive. Haettu 27.5.2017 osoitteesta
https://www.sitepoint.com/sass-basics-function-directive/

Tutorialspoint (n.d.a). CSS Syntax. Haettu 19.4.2017 osoitteesta
https://www.tutorialspoint.com/css/css_syntax.htm

Tutorialspoint (n.d.b). Node.js Tutorial. Haettu 27.5.2017 osoitteesta
https://www.tutorialspoint.com/nodejs/

Tutorialspoint (2017). JavaScript Overview. Haettu 3.4.2017 osoitteesta
https://www.tutorialspoint.com/javascript/javascript_overview.htm

W3C (n.d.a). CSS Overview. Haettu 28.3.2017 osoitteesta
https://www.w3schools.com/css/css_intro.asp

W3C (n.d.b). CSS3 Overview. Haettu 5.4.2017 osoitteesta
https://www.w3schools.com/css/css3_intro.asp

W3C (n.d.c). CSS Syntax. Haettu 19.4.2017 osoitteesta
https://www.w3schools.com/css/css_syntax.asp

W3C (n.d.d). HTML5 Overview. Haettu 30.3.2017 osoitteesta
https://www.w3schools.com/html/html5_intro.asp

W3C (2016). A Brief History of CSS Until 2016. Haettu 31.3.2017 osoitteesta
https://www.w3.org/Style/CSS20/history.html

Wendell, M. (2011). Using Pure Sass Functions To Make Reusable Logic More Useful. Haettu 23.5.2017 osoitteesta
http://thesassway.com/advanced/pure-sass-functions

Wodehouse, C. (n.d.). Front-End Web Development: Client-Side Scripting & User Experience. Haettu 4.4.2017 osoitteesta
https://www.upwork.com/hiring/development/how-scripting-languages-work/

WordPress (n.d.a). Haettu 13.4.2017 osoitteesta
https://fi.wordpress.org/

WordPress (n.d.b). Advanced CSS Controls. Haettu 31.5.2017 osoitteesta
https://en.support.wordpress.com/custom-design/editing-css/2/