CSS-ohjelmistokehysten vertailu

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, Tietojenkäsittely

Kevät 2018

Antti Kopperi

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki

Tekijä Antti Kopperi **Vuosi** 2018

Työn nimi CSS-ohjelmistokehysten vertailu

Työn ohjaaja Lauri Salminen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli etsiä vaihtoehtoa Bootstrapille, joka on suosituin front-endissä käytettävä CSS-ohjelmistokehys. Työhön valittiin kahdeksan suosituinta CSS-ohjelmistokehystä, jotka olivat Bootstrapin ohella Semantic UI, Materialize, Foundation, Bulma, Pure CSS, Skeleton ja UIkit. Niiden ominaisuuksia vertailtiin keskenään sekä kehittäjien omien kokemusten kautta että kehyksillä luomieni mallisivujen avulla, joissa esiteltiin kehysten lähinnä visuaalisia ominaisuuksia.

Tärkeimmät syyt CSS-ohjelmistokehysten käyttämiseen olivat sivuston luomisen nopeus, responsiivisuus ja selainyhteinsopivuus. Kuitenkin, mikäli toteutettava sivusto on teknisesti vaativa, ohjelmistokehystä ei välttämättä kannata käyttää. Lisäksi ohjelmistokehyksiä kohtaan on esitetty kritiikkiä siitä, että monet sivustot päätyvät näyttämään samanlaisilta.

Vertailussa kävi esille, että laajat ohjelmistokehykset Bootstrap, Semantic, Materialize, Foundation, Bulma ja UIkit ovat tarjoamiltaan ominaisuuksiltaan lähes yhtä kattavia. Pure CSS ja Skeleton ovat ominaisuuksiltaan niukimpia, mutta toisaalta keveimpiä. Näitä käyttäessä kuitenkin tulisi huomioida, että omat tyylimäärittelyt tulisi tehdä näiden päälle. Bootstrapia jo osaavalle luonteva askel osaamisen laajentamiseksi olisi Bulma, joka on ominaisuuksiltaan kattava paketti ja sen luokkanimet muistuttavat paljon Bootstrapia ja on siten helppoa oppia.

Tutkimuksessa vertailin myös ohjelmistokehysten latausnopeuksia. Nopeiten latautuivat keveimmät Pure CSS ja Skeleton. Laajoista ohjelmistokehyksistä nopeiten latautui Bootstrap. Latausnopeus oli pääasiassa parempi, kun CSS-tiedosto sijaitsi ulkoisella palvelimella, kuitenkin Pure CSS:n ja Skeletonin kohdalla tilanne oli päinvastainen.

Avainsanat Bootstrap, CSS, ohjelmistokehys, responsiivisuus

Sivut 37 sivua, joista liitteitä 1 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Business Information Technology
Visamäki

Author Antti Kopperi **Year** 2018

Subject CSS frameworks comparison

Supervisor Lauri Salminen

ABSTRACT

The aim of this study was to find an alternative for Bootstrap that is the most popular CSS framework used in the front-end development. The author of the thesis chose eight most popular CSS frameworks in this study that were Bootstrap, Semantic UI, Materialize, Foundation, Bulma, Pure CSS, Skeleton and UIkit. Their features were compared both by the expiraments of other developers and by the framework templates that were created along with this study.

The most important reasons to use CSS frameworks was the fast development of the page, responsivity and browser compatibility. However, if the website is technically demanding, it might be better not to use frameworks. Besides, the frameworks have faced critical comments on websites ending up looking like the same.

The comparison revealed that the features of the complete frameworks Bootstrap, Semantic, Materialize, Foundation, Bulma and UIkit were quite similar. There were much less features in Pure CSS and Skeleton. A developer that is going to use either of these two frameworks should create a separate style sheet. A developer that already knows Bootstrap the next easy step to expand knowledge would be to learn Bulma that is a complete CSS package and the class names of which remind Bootstrap and is easy to learn for that reason.

In this study the loading speed of the frameworks was also compared. The fastest frameworks to load were the lightest Pure CSS and Skeleton. Bootstrap was the fastest framework to load from the complete frameworks. The loading speed was mainly faster if the CSS file was in the third-party server and not in your own. However, with Pure CSS and Skeleton the result was the opposite.

Keywords CSS, Bootstrap, framework, responsivity

Pages 37 pages including appendices 1 page

SISÄLLYS

1 JOHDANTO

Web-sivujen rakentaminen on muuttunut viime vuosina paljon itse rakennettujen CSS-sivujen käyttämisestä valmiiden avointen CSS-ohjelmistokehysten käyttöön, jolla on saatu nopeutettua ja helpotettua sivujen rakentamista. Samalla sivuston käyttöliittymältä ja graafiselta ilmeeltä vaaditaan enemmän, eikä kehittämistä helpota se, että erilaisia selaimia ja eri kokoisia päätelaitteita tulee markkinoille yhä enemmän. Vaikka webkehittäjällä yleensä onkin teknistä ja graafista osaamista tyylikkään sivuston luomiseen, kaikkien näiden tekijöiden huomiointi on tuonut haastetta ja hidastanut sivustojen rakentamista, jolloin järkevämpää voisi olla käyttää valmista ohjelmistokehystä.

Olen kiinnostunut web-kehittämisestä ja minulla on jonkin verran kokemusta CSS-ohjelmistokehysten käytöstä Bootstrapin kautta. Olen kuitenkin kiinnostunut tietämään, mitä muita ohjelmistokehyksiä on saatavilla ja mitä etuja niistä voisi olla verrattuna Bootstrapiin. Halusin myös nähdä käytännössä, kuinka eri kehyksillä tehtävät sivut teknisesti toteutetaan ja minkälaisia visuaalisia ratkaisuja niissä on käytetty. Yhtenä tärkeänä ominaisuutena pidän myös sivuston latausnopeutta, jonka halusin ottaa tutkimuksessa myös huomioon.

Opinnäyte on tutkimuspainotteinen, mutta siinä on myös toiminnallista otetta. Latausnopeuksien mittaamiset suoritetaan kvantitatiivisella tutkimuksella. Tärkeänä osana työtä toteutetaan 8 suosituimmalla ohjelmistokehyksellä sisällöltään samanlaiset esimerkkisivustot, joilla voidaan esitellä paremmin niiden eroavaisuuksia käytännössä. Sivustoilta otetaan myös kuvakaappauksia työhön. Esimerkkisivustot annetaan vapaaseen tutkimuskäyttöön tämän työn ohella.

Opinnäytetyön tietoperustassa esitetään yleisestikin web-kehityksen tekniikoita, kerrotaan sivustojen responsiivisuudesta, joka myös on edellytyksenä työhön otettavien ohjelmistokehysten valinnassa, sekä avoimeen kehitykseen ohjelmistokehysten luomisessa. Tehtyjä tutkimuksia aiheesta on vähän ja aihe uudistuu jatkuvasti, sillä aikaisempien tutkimustenkin jälkeen on syntynyt uusia ohjelmistokehyksiä.

Laajoja vertailuja eri ohjelmistokehysten välillä ei juurikaan ole tehty, vaikka ohjelmistokehysten käyttö on yleistynyt voimakkaasti, osittain ehkäpä siksi, että ohjelmistokehysten käyttö on keskittynyt hyvin vahvasti Bootstrapiin ja vienyt huomiota muilta. Tämä työ esittää kuitenkin myös vähemmän käytettyjä ohjelmistokehyksiä, joista tutkimustietoa on vähän tai ainakaan suomeksi ei lainkaan.

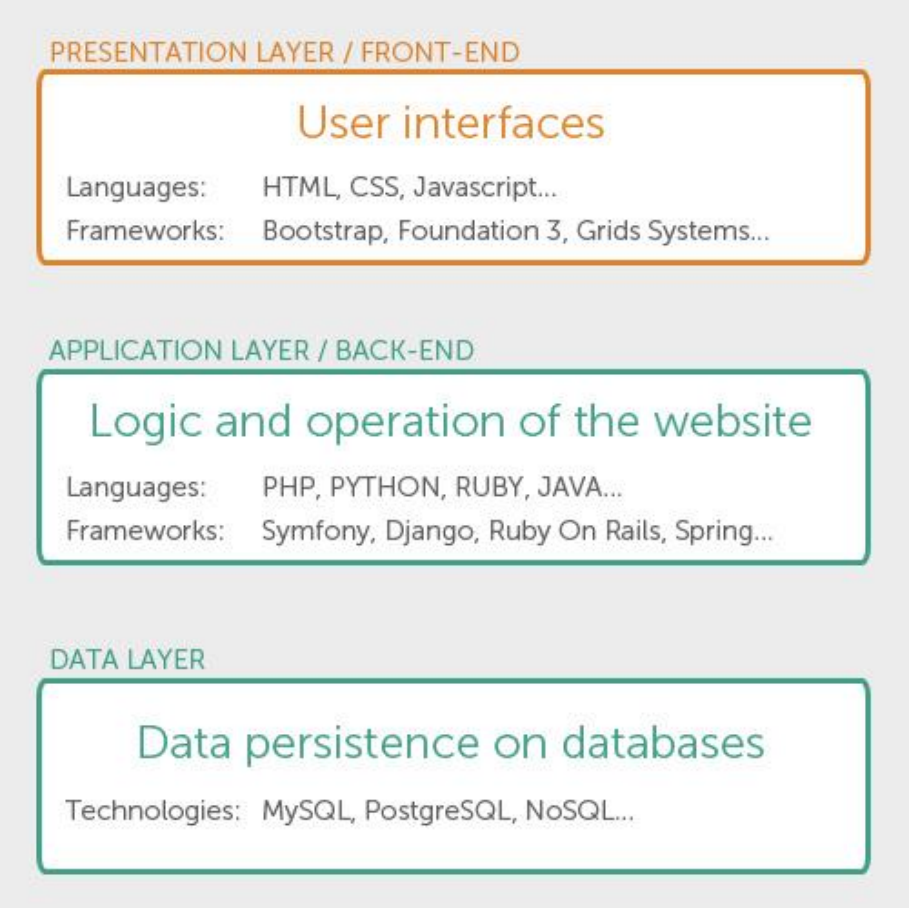
Työn tavoitteena onkin vertailla keskenään eri ohjelmistokehysten ominaisuuksia ja esitellä niitä käytännössä kuvien sekä luotavan

esimerkkisivuston avulla. Siten työ voisi toimia samalla hyvänä oppaana webkehittäjälle, joka pohtii verkkosivujen ulkoasua ja toiminnallisuutta ja sitä, millä ohjelmistokehyksellä sivusto kannattaisi toteuttaa. Toisaalta työssä pohditaan myös sitä, kannattaako webkehittäjän käyttää valmista ohjelmistokehystä vai rakentaa sivusto alusta alkaen itse.

Työ pyrkii vastaamaan seuraaviin tutkimuskysymyksiin: Miten eri ohjelmistokehykset eroavat toisistaan? Mikä ohjelmistokehys webkehittäjän tulisi valita omaan projektiinsa? Mitkä ovat ohjelmistokehysten käytön hyvät ja huonot puolet? Kannattaako ohjelmistokehystä käyttää? Mitkä ovat suosituimmat ohjelmistokehykset?

2 WEB-KEHITYKSESSÄ KÄYTETTÄVÄT TEKNIIKAT

Ohjelmistokehykset jaetaan front-endiin ja back-endiin sen perusteella, onko ohjelmistokehys näkyvissä loppukäyttäjälle vai ei. CSS-ohjelmistokehykset kuuluvat siten front-endiin ja vaikuttavat ratkaisevasti sivuston näkymään ja loppukäyttäjän kokemukseen. (Awwwards 2017.) Tämä työ keskittyy front-end-puoleen.



Kuva 1. Sivuston datakerrosten kielet, ohjelmistokehykset ja teknologiat (Awwwards 2017.)

2.1 HTML

HTML (HyperText Markup Language) ja CSS (Cascade Style Sheets) ovat verkkosivujen rakentamisen ydinteknologioita. HTML-dokumentti tarjoaa sivun rakenteen ja CSS visuaalisen ulkoasun. Näistä koostuu perusta sivustojen ja verkkosovellusten tekemiseen. HTML rakentuu useasta eri elementeistä ja sillä määritellään sivustolle esimerkiksi otsikoita, tekstiä, taulukoita, listoja ja kuvia. HTML-dokumentti kertoo selaimelle, mitä elementtejä web-sivun tulee sisältää ja mitä tyyliä sen tulee käyttää. (W3C 2018)

HTML on avoimesti standardisoitu kieli, jonka ensimmäinen versiota alettiin suunnitella vuonna 1990. Vuonna 1991 julkaistiin ensimmäinen kuvaus

HTML:stä. Vuosien aikana kieltä on kehitetty ja siihen on luotu uusia elementtejä. HTML 2.0 julkaistiin vuonna 1995 sisältäen muun muassa taulukot. 1997 alkuvuodesta julkaistiin HTML 3.2, joka oli ensimmäinen kansainvälisen W3C-yhteisön (The World Wide Web Consortium) vahvasti kehittämä ja suosittelema julkaisu. Vuoden 1997 lopulla julkaistiin vielä HTML4-versio. Vuoden 1999 lopussa laadittiin päivitys 4.01, minkä jälkeen päivityksiä ei tehty useampaan vuoteen, sillä W3C:n organisaatio keskittyi kehittämään 2000-luvun alkupuolen XML-pohjaista XHTML-kieltä. (Korpela 2011, 24.)

2000-luvulla uutena asiana nousi esille mobiililaitteiden voimakas käytön kasvu ja internetin selaaminen mobiilissa. Myös internetin käyttö muuttui nopeasti. Video- ja äänentoistopalveluiden käyttö lisääntyi voimakkaasti. Vuosina 2004-2006 Applen, Mozillan ja Operan perustama WHATWG-yhteisö (Web Hypertext Application Technology Working Group) alkoi pohtia uudempaa, aikaisempien versioiden pohjalle rakentuvaa HTML-versiota, jossa suunnitelmina oli mm. lomakkeiden toiminnallisuuden laajentaminen (Web Forms 2.0) sekä Web Applications 1.0. W3C:n ja WHATWG:n suunnitelmat yhdistettiin ja vuonna 2008 julkistettiin ensimmäinen luonnos HTML5:stä, joka toi kyseiset elementit mukaan. Järjestöt kuitenkin kehittävät osin itsenäisesti omia standardejaan HTML:n kehittämiseksi, WHATWG lähes päivittäin, joskin muutokset ovat pieniä. W3C:n työnimenä oli ensin HTML5.1, ja tällä hetkellä käytössä on versio HTML5.2. (W3C 2018; Korpela 2011, 24; Korpela 2014, 29-30.)

HTML5:n monet ominaisuudet toivat myös uutta verkkosivujen mediantoisto-ominaisuuksiin. Uusia elementtejä olivat mm. canvas-, video- ja audio-elementit. HTML5:ssä on tuotu myös paljon rakenteellisia uusia elementtejä, joilla on helpotettu kehittäjän työtä. Kun aikaisemmin header-, nav- ja footer-osiot kuvattiin usein div-elementin sisällä, ovat kyseiset osiot HTML5:ssä omina elementteinään, samaten on luotu main-, aside-, article- ja section-elementit. (W3Schools 2018; Korpela 2014, 227-234.)

Taulukko 1. Kaavakuva HTML5:n rakenne-elementtien tyypillisestä käytöstä (Korpela 2014, 249.)

<table>
<tr><td colspan="3">header
esim. logo, sivuston nimi, tunnuskuva, otsikko</td></tr>
<tr><td>nav
sivuston
sisäinen
navigointi</td><td>(main tai div tai article)
sivun varsinainen sisältö</td><td>aside
esim.
linkkejä
lisätietoihin</td></tr>
<tr><td colspan="3">footer
esim. päiväys, linkkejä</td></tr>
</table>

Yllä olevan kaavakuvan rakenne-elementeillä header, nav, main, footer ja aside sivusto voidaan jakaa yksinkertaisemmin kuin div-elementeillä. Näiden elementtien käytöstä voi toki aiheutua ongelmia vanhimmilla

selaimilla, mutta nämä ovat kuitenkin HTML5:ssä käytössä ja mukana selainten tuoreimmissa versioissa. (Korpela 2014, 249.)

2.2 **CSS**

CSS (Cascading Style Sheets), eli tyyliohje, kuuluu HTML:n ohella verkkosivun perusrakenteisiin. Se koostuu säännöistä ja sillä määritellään HTML-sivun elementteihin ominaisuuksia hyvin yksityiskohtaisestikin, kuten tekstien typografeja, värejä, ulkoasua, fontteja, elementtien kokoja ja sivuston skaalautuvuutta eri kokoisille näytöille. CSS on HTML:stä riippumaton, joten sitä voidaan käyttää myös minkä tahansa xml-pohjaisen kielen kanssa. CSS on helposti opittavissa ja ymmärrettävissä. (Korpela 2014, 71; W3C 2018; Tutorialspoint 2018.)

CSS:n kehitti Håkon Wium Lie lokakuussa 1994. Sitä ylläpitää W3C-yhteisö, joka antaa aika ajoin suosituksia uusista lisäyksistä CSS:ään. Kieli on kehittynyt vuosien aikana. Joulukuussa 1996 W3C julkaisi CSS1:n, jossa oli yksinkertaisia visuaalisia muokkausmalleja kaikille HTML-elementeille. Toukokuussa 1998 julkaistiin CSS2, joka rakentui CSS1:n pohjalle, ja jossa oli mediatuki mm. tulostimille ja audiolaitteille, ladattavia fontteja, elementtien sijoittelulle ja taulukoille. Kesäkuussa 1999 julkaistiin edelleen käytössä oleva versio CSS3, joka pohjautuu CSS1:een ja CSS2:een. (Tutorialspoint 2018.)

”CSS:n sääntö koostuu yhdestä tai useammasta selektorista ja aaltosulkeissa olevista deklaraatiosta. Selektori ilmoittaa, mitä elementtejä deklaraatio koskee, ja se on yksinkertaisimmillaan vain elementin nimi. Deklaraatio koostuu ominaisuuden nimestä kaksoispisteestä ja ominaisuuden arvosta.” (Korpela 2014, 71-72.)

Esimerkiksi body-elementin font-family-ominaisuudelle voidaan antaa arvoksi Verdana ja asettaa sivun taustakuvaksi kuva back.gif seuraavasti:

```
body {
  font-family: Verdana;
  background-image: url("back.gif");
}
```

CSS:n sisältämät tiedot voitaisi sisällyttää myös HTML-dokumenttiin, mutta järkevä ja HTML5:n standardin mukainen ratkaisu on eriyttää CSS-tiedot omaan tiedostoon, jolloin sivuston ylläpito on helpompaa, kun sama CSS-tiedosto voidaan linkittää kaikille sivuston sivuille. Sivustolle voidaan myös helposti määrittää useita erilaisia tyylejä ulkoisista CSS-tiedostoista, ja sivu voidaan rakentaa niin, että myös sivuston käyttäjä voi vaihtaa tyyliä, vaikka HTML-dokumentin rakenne pysyykin samana. Tällä tavoin sivun ulkoasua voidaan yksilöidä käyttäjän mieleiseksi. (W3C 2018.) Tässä työssä myöhemmin esitettävät ohjelmistokehykset sisältävät omat CSS-tiedostonsa, jotka tuovat omat tyylikirjastonsa sivun käytettäväksi.

Tutorialspoint (2018) listaa CSS:n käytön edut seuraavasti:

- **CSS säästää aikaa:** Kun tiedosto on kirjoitettu kerran, sitä voi hyödyntää usealla HTML-sivulla. HTML-elementeille voi kuvailla eri tyylejä ja käyttää niitä sivustoilla
- **Sivu latautuu nopeammin:** HTML-koodia ei tarvitse kirjoittaa niin paljon, vaan käyttää elementtien tyylimäärittelyyn CSS:ää. Vähemmän koodia tarkoittaa nopeampaa latautumista.
- **Helppo ylläpito:** Jos haluaa tehdä muutoksen, joka koskee sivuston kaikkia sivuja, tarvitsee muuttaa vain CSS-tiedostoa, ja muutos tulee automaattisesti kaikille sivuille.
- Laajemmat tyylimuokkaukset kuin HTML:n attribuuteissa: Sivu näyttää siten paljon paremmalta.
- **Usean laitteen yhteensopivuus:** CSS mahdollistaa sisällön optimoinnin usealle eri laitteelle. Samaa HTML-dokumenttia voidaan näyttää eri tavalla eri laitteille, mikä mahdollistaa paremman näkymän esimerkiksi tulostukseen ja mobiililaitteille.
- **Maailmanlaajuiset web-standardit:** HTML-attribuutteja ei enää suositella, vaan suositellaan käyttämään tyylimäärittelyissä CSS:ää. Tulevaisuuden selaimia ajatellen HTML-sivut tulisi siten standardin mukaisiksi.
- **Offline-selaaminen:** CSS pystyy tallentamaan web-sovelluksia paikallisesti välimuistiin. Näin voidaan katsoa sivustoa offline-tilassa. Välimuistin ansiosta sivusto myös latautuu nopeammin.
- **Alustariippumattomuus:** Toimii myös uusimmilla selaimilla.

2.3 **JavaScript**

JavaScript on korkean tason ohjelmointikieli, joka mahdollistaa web-sivuston staattisen sisällön muuttamisen sisältörikkaammaksi. Se on pääasiassa selainskriptien tekemiseen käytetty kieli ja sillä sivustolle saadaan mm. säännöllisesti päivittyvää sisältöä, interaktiivisia karttoja, animaatioita, videoita, pelejä ym. Se on HTML:n ja CSS:n lisäksi esittävä front-end-puoleen kuuluva teknologia. Mikään vaatimus se ei kuitenkaan sivustolle ole ja jos sivu on tarkoitettu vain sisällön esittämiseen ilman vuorovaikutusta käyttäjän kanssa, ei JavaScriptiä tarvita välttämättä ollenkaan. Selainohjelmoinnin (client-side scripting) lisäksi JavaScriptiä voidaan käyttää myös palvelimessa toimivana eli palvelinohjelmointiin (server-side scripting). (Korpela 2014, 55; Mozilla 2018.)

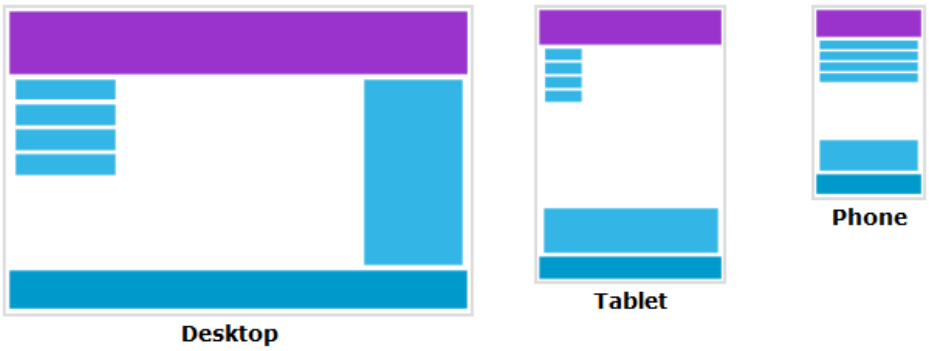
Nimestään huolimatta JavaScript-kielellä ei ole tekemistä Java-ohjelmointikielen kanssa. Nimi juontuu JavaScriptin oikeudet ostaneen Sun Microsystemin ilmeisesti markkinointisyistä sille antamasta nimestä. JavaScript on muodostunut käytännössä lähes ainoaksi selainohjelmoinnin kieleksi ja sillä on erittäin laaja tuki. Yksityiskohdissa on eroja ja kielestä on eri murteita, mutta toteutusten erot ovat JavaScriptin alkuajoista pienentyneet. Käytön aloittaminen on erittäin helppoa, mutta kokonaisuuden

omaksuminen vie kehittäjältä aikaa ja kielessä on joukko erikoisuuksia. (Korpela 2014, 56.)

Kuten ohjelmointikielillä yleensäkin, myös JavaScriptissä on käytössä mm. oliot, muuttujat ja funktiot ja rajapintojen avulla sen toiminnallisuutta voidaan laajentaa. Kuitenkaan siinä ei ole sellaista luokkarakennetta, kuten monessa ohjelmointikielessä. JavaScriptiä käytetään monessa asiassa apuvälineenä esimerkiksi paikkaamaan HTML5-tuen aukkoja selaimissa. JavaScriptillä voidaan muokata HTML-sivua, ei vain sen ulkoasua, vaan myös rakennetta ja sisältöä. Sillä voidaan esimerkiksi ohjelmallisesti luoda sivulle taulukko esimerkiksi käyttäjän antamien käskyjen mukaan. Kielellä yleisestikin lisätään sivustolle vuorovaikutteisuutta ja toiminnallisuutta. HTML5-dokumentti voi myös ääritapauksissa koostua lähes yksinomaan JavaScriptistä, esimerkiksi sitä käyttävässä sovelluksessa. (Korpela 2014, 56-59.)

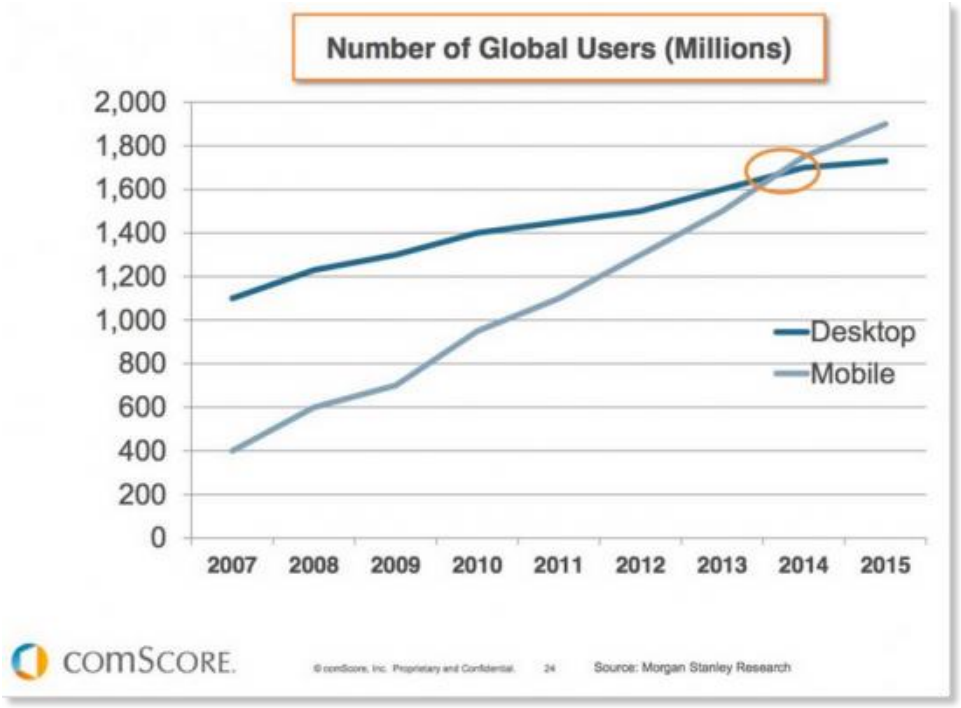
3 RESPONSIIVISUUS WEB-SIVUSTOILLA

Responsiivinen web-suunnittelu tarkoittaa web-sivuston muotoilua HTML:llä ja CSS:llä niin, että sivusto näyttää hyvältä kaikilla alustoilla. Web-sivustoja selataan pöytäkoneilla, tableteilla tai puhelimilla, ja sivuston tulisi toimia kaikilla laitteilla yhtä helposti. Sivusto ei saisi jättää informaatiota pois pienemmillä laitteilla, vaan muuntaa sisältö mahtumaan mille tahansa laitteelle. Käytettävä laite vaikuttaa myös näkymään: esimerkiksi puhelimella selattaessa sivu saattaa näyttää yhden sarakkeen kerralla ja tabletilla kaksi. (Strickler 2013; W3Schools 2018.) Marcotte (2011, 3) kuvaa ongelmaa näin: ”taiteilija voi valita, minkä kokoiselle kanvaasille aikoo maalata maalauksen, verkkosivulla kanvaasin kokoa ei voida tietää”.



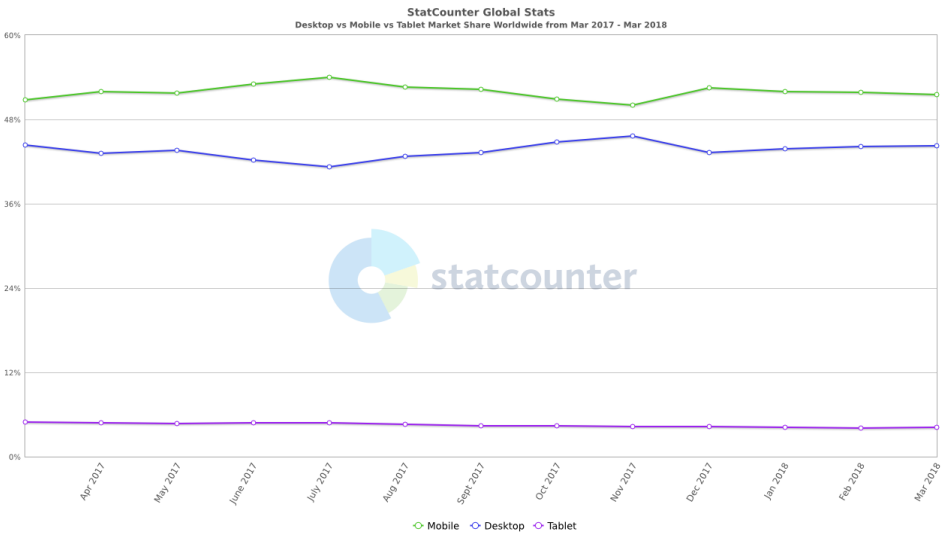
Kuva 2. Esimerkki sivuston responsiivisuudesta eri laitteille, kuvakaappaus w3schools.com-sivustolta (W3Schools 2018.)

Alla olevasta kaaviosta nähdään, kuinka mobiililaitteiden käyttö on kasvanut valtavasti suhteessa tietokoneiden käyttöön ja taitekohta tapahtui jo vuonna 2014, jolloin mobiililaitteiden käyttäjämäärä ohitti tietokoneiden käyttäjämäärän. Edelleen sivustot on kuitenkin suunniteltava niin, että niitä voidaan käyttää millä tahansa laitteella, sillä tuskin on sivustoa, jota käytetään vain tietyllä laitteella. (Chaffey 2018.)



Kuva 3. Tietokoneiden ja mobiililaitteiden maailmanlaajuiset käyttäjämäärät (Chaffey 2018.)

Syksyllä 2016 internetiä selattiin enemmän mobiililaitteiden kautta kuin tietokoneelta, joten sivustojen käytettävyydessä ei voida sivuuttaa mobiililaitteita. (Heisler 2016.) Trendi näyttää 2017-2018 pysyneen jokseenkin tasaisena, mutta mobiililaitteiden käyttö on runsaampaa kuin tietokoneiden. (StatCounter 2018.) Edelleen näkee silti sivustoja, joita ei ole optimoitu mobiililaitteille. Mobiililaitteet vaativat erilaista lähestymistapaa siinä, miten sivun rakenne asettuu näytölle. Kun näyttöjen koot jatkuvasti muuttuvat, sivuston tulisi voida muuntautua mihin tahansa kokoon, nyt ja tulevaisuudessa. (LePage, 2018.)



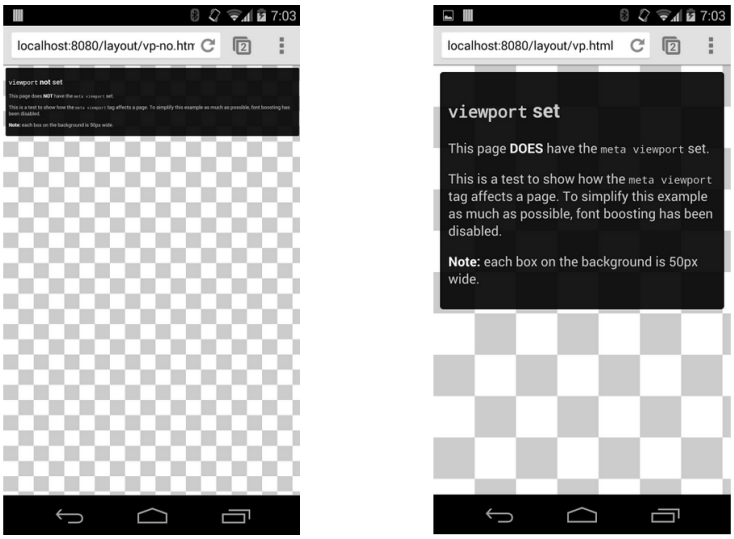
Kuva 4. Internetin käytön suhde maailmanlaajuisesti tietokoneiden ja mobiililaitteiden kesken (StatCounter 2018.)

Ennen kuin tabletit ja mobiililaitteet olivat yleistyneet, web-sivustoja saatettiin suunnitella vain tietokonenäytöille, jolloin web-sivustoilla saattoi olla paljon staattisia rakenteita ja esimerkiksi kuvilla määrättyjä kokoja (engl. fixed size). Tällä menetelmällä rakennetut sivut eivät näytä hyvältä pienemmillä näytöillä, joka joutuu pienentämään sivuston sopimaan näytöllensä. Yksi nopea korjaus tällaisen sivuston ongelmaan voi olla viewport-määritelmä head-tagien sisällä:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
...
```

(W3Schools 2018.)

Tämä elementti antaa selaimelle ohjeen siitä, kuinka käsitellä sivuston mittasuhteita käytettäessä suurennusta ja pienennystä. ”Width”-muuttuja asettaa leveydeksi käyttäjän päätelaitteen leveyden ja ”initial-scale”-muuttujaan asetetaan haluttu suurennuksen taso, kun sivu ladataan. Yllä olevassa esimerkissä oleva arvo 1.0 tarkoittaa, että 1 sivuston pikseli on 1 päätelaitteen pikseli ja arvo 0.5, että 1 sivuston pikseli on 0,5 päätelaitteen pikseliä. (W3Schools 2018.)



Kuva 5. Mobiilinäkymät esimerkkisivustoista, jossa vasemmalla olevalle ei ole asetettu viewport-määritelmää, ja oikean puoleiseen on. Kuvakaappaus sivustolta. (LePage 2018.)

Viewport-elementtiä käytettäessä on huomioitava, että HTML-elementeissä ei voi käyttää määrättyjä (”fixed”) pikselikokoja, vaan ne on asetettava suhteellisina prosenttikokoina. Tällä ratkaisulla päästään lähelle responsiivisen sivuston määritelmään, jossa sivusto skaalautuu näytön mukaisesti eri tavoin. Paremmat ratkaisut kuitenkin saadaan kehittyneemmillä ratkaisuilla (esimerkiksi grid system ja flexbox), joissa elementit

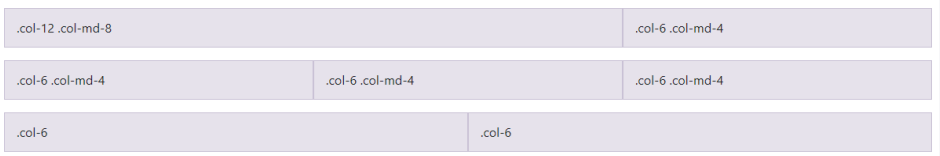
voivat vaihtaa sijoitteluaan vaakatasosta pystytasoon näytön koon muuttuessa. (W3Schools 2018.)

Responsiivisessa web-suunnittelussa on huomioitava monia asioita, jotta käyttökokemus olisi hyvä kaikilla laitteilla. Nämä tulisi määritellä CSS-tiedostojen media query -osioissa, joissa asetetaan tietyt tyylimäärittelyt sivustolle, kun sivuston leveys alittaa media query -osioon merkityn leveyden. (LePage 2018.) Tällä tavoin isolla näytöllä marginaalit, fontit ja elementit ovat isompia ja pienellä näytöllä pienempiä tai niiden keskinäinen sijainti voi vaihtua. Media queryn avulla saadaan sivuston responsiivisuus aivan uudelle tasolle. (Marcotte 2011, 79.)

Sivuston responsiivisuus on tärkeää, koska sen avulla web-kehittäjien ei tarvitse tehdä useita eri sivustoja samalla sisällöllä vain sen takia, että käyttäjillä on eri kokoisia laitteita. Myös Google optimoi hakutuloksiaan sen mukaan, onko sivusto responsiivinen. (Strickler 2013.)

3.1 Sivuston rakenteen luominen grid systemillä ja flexboxilla

Grid system on responsiivisen sivuston rakentamisen menetelmä, jolla luodaan sille selkeä pohjarakenne. Se on hyvä ja suositeltu tapa organisoida sivuston moduuleja ja koodin osia. Grid on käytännöllinen sekä web-kehittäjille että käyttäjille. Se selkeyttää sisältöä ja parantaa sivuston käyttäjän kokemusta. Grid ei tarkoita, että sivuston ulkoasusta tulisi tylsä tai laatikkomainen, sen estäminen jää edelleen web-kehittäjän tehtäväksi. (Shillcock 2013.)



Kuva 6. Kuvakaappaus, esimerkki Bootstrapin grid systemistä (Bootstrap 2018c.)

Grid system rakentuu periaatteessa horisontaalisista ja vertikaalisista viivoista, jotka risteämällä luovat ruudukon. Näin syntyviin ruutuihin luodaan sivun sisältöä. Tällä tavalla sisällöstä saadaan paljon luettavampaa ja hallittavampaa. (Shillcock 2013.)

Kun grid systemillä pystyy rakentamaan sivulle kaksiulotteista rakenteellisuutta, flexbox soveltuu vain yksiulotteiseen sisällön esittämiseen. Siinä elementtejä sijoitellaan horisontaalisesti peräkkäin, ja jos jokin elementti ei mahdu riville, se siirtyy seuraavalle riville. Mikään ei tietenkään estä yhdistämästä näitä kahta tapaa keskenään, ja se onkin paikoin järkevää, sillä flexboxilla on omat vahvuutensa tietyissä tilanteissa lyhyemmän rakenteellisuutensa ja joustavuutensa vuoksi. (Borgen 2017.)

3.2 **CSS-esiprosessorit**

CSS-esiprosessori on skriptikieli, joka laajentaa CSS:ää ja mahdollistaa web-kehittäjän kirjoittamaan koodia esiprosessorin kielellä ja muuntaa sen sitten CSS:ksi. Tunnetuimpia esiprosessoreita ovat SASS, LESS ja Stylus. (Coron 2017.) CSS-kieli on hyvin alkeellinen ja keskeneräinen kieli. Funktioiden, määrittelyjen uudelleen käyttäminen ja periytyvyys on hyvin vaikeaa toteuttaa siinä. Isommissa, monimutkaisissa sivustoissa, ylläpidosta voi tulla siitä syystä iso ongelma. Aluksi ongelmaa yritettiin korjata jakamalla CSS-tiedostoja pienempiin osiin, useampaan tiedostoon, mutta tämä johti koodin toistuvuuteen ja erilaisiin ylläpitohaasteisiin. (Cinarli 2014; Maheedharan 2017.) Tässä CSS-esiprosessorit tulevat apuun. Ne tekevät suunnittelusta paljon tehokkaampaa ja helpompaa, koska ne mm. käyttävät apunaan olio-ohjelmointia. (Bradford 2017.)

SASS ja LESS ovat tarkoitukseltaan samanlaisia, ne nopeuttavat CSS:n luomista ja ovat erityisen hyödyllisiä isommissa projekteissa. Ne perustuvat molemmat olio-ohjelmointiin. Ne mahdollistavat mm. muuttujien ja sisäkkäisten sääntöjen (engl. nested rules) käytön, helpottavat pitämään säännöt järjestyksessä ja nopeuttavat CSS:n kirjoittamista. Kuten muissakin ohjelmointikielissä, muuttujiin voi tallettaa tietoa, jota voi käyttää koko tyylitiedostossa. Esimerkiksi tietyn värin voi sijoittaa muuttujaan tiedoston alkuosassa ja käyttää sitten värimuuttujaa tiedoston osissa. Tämä nopeuttaa erityisesti muutosten tekemistä, sillä mikäli väriarvoa haluaa muuttaa, tarvitsee muuttaa vain muuttujan arvoa, ei kaikkia osia erikseen. (Bradford 2017; Coron 2017.)

SASS:lla ja LESS:llä on kuitenkin joitakin eroavaisuuksia. SASS käyttää ohjelmointikielenä Rubyä, LESS puolestaan JavaScriptiä. Muuttujien määrittelyssä SASS käyttää merkkiä $ ja LESS merkkiä @. LESS:llä on paremmat virheilmoitukset ja käyttäjäystävällisempi dokumentaatio kuin SASS:lla. Kuten monessa muussakin, keskinäistä paremmuutta on kuitenkin vaikea osoittaa, ja monesti onkin vain web-kehittäjän omista mieltymyksistä kiinni, kumpaa kannattaa käyttää. (Bradford 2017.)

Esimerkki SASS-tiedoston muuttujista:

```
$fontti: Helvetica, sans-serif;
$vari: #333;

body {
  font: 100% $fontti;
  color: $vari;
}
```

Yllä oleva esimerkki muodostaa seuraavanlaisen CSS-tiedoston:

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

4 CSS-OHJELMISTOKEHYKSET

Awwwards (2017) ja Roy (2018) kuvaavat CSS-ohjelmistokehyksiä (engl. CSS Framework) seuraavasti: Ohjelmistokehys on standardisoitu paketti HTML:ää, CSS:ää ja JavaScriptiä, jota voidaan käyttää nopeuttamaan HTML5-sivustojen rakentamista. Verkkosivujen kehittämisestä Awwwards nostaa esiin ongelman siinä, että eri sivustojen rakenteet ovat pohjimmiltaan hyvin samanlaisia, vaikkakaan eivät saman näköisiä ja saman sisältöisiä. Miksi siis käyttää aikaa samojen rakenteiden tekemiseen moneen kertaan, jos ne voitaisi hyödyntää valmiilla standardisoidulla mallilla? Ohjelmistokehysten tehtävänä onkin olla selkeänä pohjana sivustolle, jolloin web-kehittäjien ei tarvitse tehdä samaa työtä useaan kertaan, ja siten voidaan säästää todella paljon aikaa ja rahaa. (Awwwards 2017.)

Awwwardsin (2017) mukaan yleensä CSS-ohjelmistokehyksille tyypillisiä yhteisiä komponentteja ovat:

- CSS-lähdekoodi, jolla luodaan sivuston rakenne, johon sivuston ylläpitäjä voi sitten sijoittaa erilaisia elementtejä
- HTML-elementtien, typografian ja tyylien määrittelyt
- Ratkaisut, joilla saadaan sama näkymä eri selaimille
- CSS-luokkien luominen HTML-elementeille

Awwwards (2017.)

Eri ohjelmistokehyksissä on monia visuaalisia eroavaisuuksia, mutta niistä lähes kaikki käyttävät ruudukkoa (engl. grid system) tai flexboxia HTML-sivun rakenteen pohjana. Yhteistä niille on myös se, että niillä on oma standardisoitu konseptinsa HTML5:lle, CSS:lle ja JavaScriptille. Tärkein yhteinen ominaisuus niillä kuitenkin on keskittyminen responsiivisuuteen. (Roy 2018; Prechu 2018.)

4.1 CSS-ohjelmistokehysten hyödyt ja haitat

Awwwards (2017), Belén (2016) ja Prechu (2018) mainitsevat CSS-ohjelmistokehysten käytössä seuraavia hyötyjä ja haittoja web-kehittäjän näkökulmasta:

Taulukko 2. CSS-ohjelmistokehysten hyödyt ja haitat web-kehittäjän näkökulmasta

Hyödyt	Haitat
- sivuston tekemisen nopeus - koodin puhtaus - ratkaisut yleisiin CSS-ongelmiin - selainyhteensopivuudet	- osin turhaa koodia - sivuston tekijä ei opi itse tekemään ratkaisuja - tekninen velka isommissa sivuistoissa

- koodeissa hyviä esimerkkiratkaisuja ongelmiin - helppous yhteisprojekteissa - sivustojen responsiivisuus	

Kehysten käytön suurin hyöty, joka tulee kaikessa tekemisessä vastaan, on ehdottomasti sivuston tekemisen nopeutuminen. CSS:ää ei tarvitse itse tehdä alusta alkaen, vaan valmiilla pohjalla pääsee helposti alkuun. Responsiivisuusominaisuuksien luonti sivustolle on käytännössä jokaisella sivustolla lähes samanlaista, joten olisi järkevää joka tapauksessa käyttää ohjelmistokehystä vähintäänkin siltä osin. On myös olemassa minimaalisilla ominaisuuksilla olevia CSS-ohjelmistokehyksiä, jotka tarjoavat vain tuon responsiivisuuspohjan ja loput ominaisuudet voi määritellä itse. Nopeutta lisää se, ettei tarvitse säätää CSS:ää sopimaan eri selaimille, vaan ratkaisut tulevat valmiina. (Andrew 2011.)

Web-kehittäjän osaamisesta ollaan yleisesti sitä mieltä, että ohjelmistokehykset tarjoavat kehittäjälle hyviä esimerkkejä ratkaisuista, jolloin kehittäjä oppisi näkemään, kuinka vaativat tekniset ratkaisut toteutetaan. Toisaalta, samalla nostetaan esille ongelma siinä, ettei kehittäjä kuitenkaan oppisi itse tekemään näitä ratkaisuja, eikä oppimista siten tapahdu. (Andrew 2011; Belén 2016.) Kyseessä näyttäisikin olevan siten yksilöllinen kokemus siitä, haluaako web-kehittäjä nähdä vaivan oppimisesta.

Mozillan ohjelmistokehittäjä Albeza Belén (2016) nostaa esiin blogikirjoituksessaan CSS-ohjelmistokehysten haittapuolia. Kirjoituksessaan hän myös kritisoi web-kehittäjän tarvetta, sillä sivuston voi rakentaa lähes kokonaan kopioimalla valmiin layoutin ja käyttönäkymän. Näin web-kehittäjä ei itse opi kirjoittamaan CSS:ää, eikä suunnittelemaan sivuston ulkoasua ja rakennetta. Jos CSS:ää ei räätälöidä, sivusto päätyy näyttämään samanlaisilta kuin moni muu sivusto. Sivuston saa kyllä kasaan nopeasti ja se voi olla hyvä asia pienelle projektille, mutta isompaa jatkuvasti käytettävää sivustoa tai sovellusta tehdessä ja käyttäessä siinä valmista ohjelmistokehystä syntyy ”teknistä velkaa”. Tekninen velka alkaa näkyä muun muassa siinä, että ohjelmistokehyksen omia tyylimääritelmiä joudutaan jatkuvasti ylikirjoittamaan omilla määritelmillä, jolloin tekninen toteutus kärsii, ja sivuston latausnopeus ja kehittäjän työ hidastuvat.

Belén (2016) kehottaa ohjelmistokehysten käytön sijaan kirjoittamaan CSS-tiedoston itse, jolloin itse voi määritellä haluamansa tyylit ja sivuston saa muokattua haluamakseen. Jos ei halua koodata yksittäistä tiettyä ominaisuutta alusta asti, sen osan pystyy aina kopioimaan omaan projektiin mukaan tai käyttämään apuna kolmannen osapuolen komponentteja.

Haittapuoliin lukeutuu myös se, että koska monet ohjelmistokehyksistä tarjoavat ”koko paketin”, lisää tämä ylimääräistä CSS-koodia, koska harvoin sivustolla käytetään jokaista kehyksen osaa. Tästä tulee ongelma sivustolla, jolla latausnopeus on erittäin tärkeää ja varsinkin mobiilikäytössä.

Tällöin CSS:stä kannattaa poista osiot, joita ei tarvitse. Tosin tämän tekeminen itse on erittäin haastavaa, koska valmiissa kehyksessä yhtä kohtaa koskevat asiat voi olla linkitetty moneen eri kohtaan. Tästä syystä kannattaakin käyttää joidenkin ohjelmistokehysten valmiiksi tarjoavia CSS-muokkaimia, jolla kehykseen voi valita haluamansa osiot. (Andrew 2011; Bootstrap 2018b.)

4.2 **Ohjelmistokehysten valinta tutkimukseen ja esittelyt**

Awwwards (2017) opastaa pohtimaan seuraavia tekijöitä oikean CSS-ohjelmistokehyksen valintaan sivustolle web-kehittäjän näkökulmasta:

- Asentamisen nopeus
- Ymmärtämisen helppous
- Vaihtoehtojen runsaus
- Integrointi olemassa oleviin järjestelmiin
- Hyvä ja pitkä tuki

Ohjelmistokehykset voidaan myös jakaa kahteen eri kategoriaan kehyksen kompleksisuuden mukaan. Yksinkertaiset kehykset saattavat sisältää vain ruudukkopohjan (gridin) asetukset ja ehkä joitakin yksinkertaisia niihin liittyviä asetuksia. Näistä voidaan mainita mm. 1140 CSS Grid, Golden Grid System, Mueller Grid System, Titan Framework ja useita muita responsiivisia rakennepohjia. Tunnetumpia ohjelmistokehyksiä ovat kuitenkin ns. kokonaiset kehykset, jotka sisältävät esimerkiksi tyylitellyt typografiat, lomakkeet, napit, ikonit sekä komponentteja, joilla voidaan tehdä navigointiosiot, ilmoitukset, kuvakehykset ja niin edelleen. Näitä ovat mm. Bootstrap, Foundation ja Skeleton. (Awwwards 2017.)

Koska useimmat vähänkään laajemmat sivustot vaativat useita kokonaisten kehysten sisältämiä ominaisuuksia, valitaan tässä tutkimuksessa esitettäviin ohjelmistokehyksiin vain niitä, jotka tarjoavat valmiiksi kattavan sisällön eri toimintoja. Tämä rajaus tehdään toki siitäkin syystä, että yksinkertaisia kehyksiä on niin valtavasti tarjolla ja uusia syntyy koko ajan lisää, ettei kaikkia voida esittää. Myös kokonaisten kehysten tarjonta on runsasta: lyhyehkön tutkimisen jälkeen niitäkin löytynee satoja erilaisia. Tutkimuksessa on siten järkevintä keskittyä kokonaisiin kehyksiin, jotka ovat responsiivisia ja joita käytetään eniten. Tämä ei välttämättä tarkoita sitä, että ne olisivat parhaita ratkaisuja, mutta käyttömäärät kertovat aina omaa kieltään. Suurella käyttömäärällä voidaan myös olla varmemmalla pohjalla hyvän tuen osalta tulevaisuudessa ja ongelmatilanteissa apua löytyy varmasti helpommin.

GitHubin käyttäjät voivat käyttää tähtiä merkitsemään heitä kiinnostavia aiheita, jotta he löytävät aiheet myöhemmin helposti. GitHub myös ylläpitää listausta meneillään olevista trendeistä merkattujen tähtien perusteella. Palvelulla on 28 miljoonaa käyttäjää maailmanlaajuisesti, ollen

suosituin kehittäjien yhteisö, joten tähtien määrästä voidaan huomata erilaisten teknologioiden ja trendien suosio. (Github 2018a; Github 2018b; Github 2018c.) Erittäin hyväksi listaukseksi tätä tutkimusta ajatellen on GitHubin käyttäjien suosituimmat CSS-ohjelmistokehykset:

Taulukko 3. Suosituimmat CSS-ohjelmistokehykset Github-tähtien mukaan ja niiden Stackoverflow-keskustelujen määrä. Lukemien määrä on päivitetty 17.4.2018. (Sitecake 2018)

Ohjelmistokehys	Github-tähdet	Stackoverflow-keskustelut
1. Bootstrap	123705	272003
2. Semantic UI	40710	43847
3. Materialize	32277	15958
4. Foundation Zurb	27217	3566
5. Bulma	25971	542
6. Pure CSS	18495	500
7. Skeleton	15490	epävarma tulos
8. UIkit	12199	epävarma tulos
9. Miligram	7051	1
10. Tachyons	6803	127

Yllä olevasta taulukosta nähdään, että GitHubin käyttäjien keskuudessa selkeästi suosituin CSS-ohjelmistokehys on Bootstrap. Toisena oleva Semantic UI on saanut vain kolmanneksen Bootstrapin tähtien määrästä. Muut listassa olevat ohjelmistokehykset ovat noin muutaman tuhannen tähden eroilla toisistaan. Suosituimmat listassa olevat kehykset ovat ns. kokonaisia ohjelmistokehyksiä laajoilla ominaisuuksilla ja kaikki ovat responsiivisia. Laajuuseroja kuitenkin on. Viidentenä oleva Bulma ei sisällä omaa paketoitua JavaScriptiä. Listan kuudentena ja seitsemäntenä olevat Pure CSS ja Skeleton ovat ominaisuuksiltaan melko riisuttuja ja tarjoavat vain perusominaisuudet (lähinnä gridin) ollen samalla hyvin keveitä. Asetan tähän tutkimukseen otettavien ohjelmistokehysten rajaksi vähintään 10.000 Github-tähteä, jolla saadaan kahdeksan tutkittavaa kehystä. Hieman eri järjestyksessä, mutta samanlaisiin valintoihin, suosituimmista ohjelmistokehyksistä päätyvät myös Awwwards (2017), Prechu (2018), Shaleynikov (2018) ja Roy (2018). Roy nostaa kuitenkin vuoden 2018 viidenneksi suosituimmaksi ohjelmistokehykseksi Kuben.

Seuraavana esittelen tarkemmin 8 työhön valittavaa ohjelmistokehystä, joiden sisällöstä puhuttaessa käytän ohjelmistokehysten vain itsessään tarjoamia ominaisuuksia ilman lisäosia ja toisaalta myös niin, että niistä käytetään ohjelmistokehysten koko versiota ilman minkään osion poistamista.

4.2.1 Bootstrap

Bootstrapin alku pohjautuu vuoteen 2010, kun kaksi Twitterin työntekijää, Mark Otto ja Jacob Thornton, kehittivät sen nopeuttamaan yrityksen

kehittäjien työtä. Bootstrap tunnettiinkin nimellä Twitter Blueprint ennen kuin siitä tuli avointa 19.8.2011. Kuten muutkin useat muutkin ohjelmistokehykset, myös Bootstrap muodostuu HTML-, CSS- ja JavaScript-tiedostoista. Se käyttää esiprosessorina SASS:a ja on kehitetty erityisesti front-end-kehitystä silmällä pitäen. Se helpottaa rakentamaan mm. web-suunnittelukonsepteja ja typografiaa. Bootstrap itsessään ei sisällä lisäosia, mutta siihen on saatavilla kolmansien osapuolten lisäosia. (Shaleynikov 2017; Sitecake 2018.)

Alun jälkeen Bootstrapilta on tullut yli 20 versiota. Laaja kehittäjäyhteisö on parantanut CSS:ää vuosien aikana. Versio 2.0 toi responsiivisen suunnittelun valtavirran tietouteen ja 3.0:n myötä parannettiin responsiivisuutta mobiiliystävällisemmäksi, ikonit käyttämään web-fontteja ja Internet Explorer 7:n tuen lakkaamisen myötä CSS sujuvammaksi. Aikaisemmissa versioissa esiprosessorina käytettiin LESS:ä, mutta 4.0:ssa siirryttiin käyttämään SASS:a. Viimeisin versio 4.1.1 julkaistiin huhtikuussa 2018. (Cochran & Whitley 2014, 23-24; Bootstrap 2018a.)

Bootstrap on Github-kehittäjien parissa suosituin ohjelmistokehys. Bootstrapin suosio perustuu siihen, että se on käyttäjäystävällinen, sen ratkaisut on testattu usealla selaimella ja siten sen laatu on korkeaa.

Roy (2018) listaa Bootstrapin hyvät ja huonot puolet seuraavasti:

Taulukko 4. Bootstrapin hyvät ja huonot puolet

Hyvää	Huonoa
- Hyvin standardisoitu ohjelmistokehys, sisältää kaikki tarpeelliset komponentit front-endin tarpeisiin - Kaikki isot selaimet tukevat Bootstrapia - Tarjoaa yhtenäisen ulkoasun - Responsiivisuus ja mobiilivetoinen kehitys - Paljon ilmaisia ammattimaisia malleja, teemoja ja lisäosia	- Kompleksinen tyyli johtaa kirjoittamaan paljon HTML:ää - JavaScript on sidottu jQueryyn - Vaatii kehittäjältä paljon uudelleenkirjoittamista, jos rakennetta haluaa kustomoida - Bootstrap on todella suosittu ja sitä käyttävät web-sivustot näyttävät samanlaisilta, ellei niitä kustomoida. Tämä tekee erottautumisen vaikeaksi.

Bootstrapin koko versio sisältää runsaasti CSS-tyylimäärittelyjä, osin turhaakin, mikä hidastaa verkkosivun latautumista. Siksi Bootstrapia käytettäessä on järkevää hyödyntää Bootstrapin omaa työkalua, jolla voi valita vain haluamansa komponentit ja jättää turhat komponentit pois. Tämä ominaisuus on ollut käytössä 30.7.2016 päivityksen jälkeen. Työkalulla voi myös helposti määritellä haluamansa värit, typografiat ja erilaisia kokovaihtoehtoja ilman itse koskemista CSS-tiedostoon ja ilman tyylien ylikirjoittamista omalla CSS-tiedostolla. Bootstrapin työkalu muodostaa myös

automaattisesti min.css -päätteisen css-tiedoston, josta on poistettu turhat rivinvaihdot ja joka kuitenkin on saman sisältöinen kuin tavallinen css-tiedosto. Nämä kaikki toimenpiteet nopeuttavat sivun latautumista. (Bootstrap 2018b).

Vaikkakin Bootstrap sisältää JavaScript-tiedoston mukanaan, ei se vaadi kehittäjältä sen osaamista. Aloittelevalle web-kehittäjälle Bootstrap on siten helppo ohjelmistokehys alkuun. (Chishkala 2018.)

4.2.2 Semantic UI

Semantic UI on Bootstrapiin verrattuna uudempi (kehitys aloitettiin 2013) ohjelmistokehys, mutta monet kehittäjät pitävät sitä varteenotettavana, jopa parempana vaihtoehtona kuin Bootstrap. Semantic käyttää esiprosessorina LESS:ä. Käyttöliittymän ilme on virtaviivainen, hienovarainen ja tasainen. Web-kehittäjälle Semanticin CSS-tiedostossa käytetyt luokkien nimet ovat hyvin käyttäjäystävällisiä ja hyvin lähellä puhuttua kieltä. (Gerchev 2014; Chishkala 2018.) Esimerkiksi kolme saraketta sisältävä osio voidaan luoda HTML-dokumenttiin seuraavasti:

<div class=”three wide column”>

Tätä voi pitää suurena etuna Bootstrapiin verrattuna. Sen sijaan web-kehittäjältä ohjelmistokehyksen käyttö vaatii myös JavaScript-osaamista, jolla luodaan visuaalista sisältöä sivulle. (Chishkala 2018.) Jordon (2018) listaa blogissaan Coderwall-sivustolla Semanticin hyvät ja huonot puolet seuraavasti:

Taulukko 5. Semanticin hyvät ja huonot puolet (Jordon 2018.)

Hyvää	Huonoa
- Julkaistu MIT-lisenssin alaisuudessa - Erittäin hyvin dokumentoitu - Helppo oppia ja käyttää - Käyttää grid-layoutia ja LESS:ä - Hyviä visuaalisia ominaisuuksia - Avoin kehitys	- Ei kuva-slideriä - Ei thumbnail-luokkaa - Ei visibility-luokkaa - EI SASS:a, mutta käyttää LESS:ä - On vielä kehitysversiona

4.2.3 Materialize

Materialize CSS on ohjelmistokehys, joka perustuu Android OS:n designiin ja on siten hyvä valinta niille, jotka pitävät kyseisestä tyylistä. Kehys tarjoaa joitakin hyviä visuaalisia ominaisuuksia, joita muut kehykset eivät tarjoa. (Roy 2018)

Roy (2018) listaa Materializen hyvät ja huonot puolet seuraavasti:

Taulukko 6. Materializen hyvät ja huonot puolet

Hyvää	Huonoa
- Elementtien omanlainen näkymä ja käyttäytyminen - Erilainen visuaalinen ulkoasu tarjoaa vaihtelua - Hyvä dokumentointi - Responsiivisuus	- Ei tue joitakin vanhempia selaimia - Tiedostot ovat isoja ja raskaita

Materialize käyttää pohjana saman tyyppistä 12 sarakkeeseen jakautuvaa responsiivista näkymää, kuten Bootstrapkin. Visuaaliselta puolelta suurin eroavaisuus ja hyvä puoli on HTML-elementeille tuleva luokka, jolla määritellään elementin sijainti myös syvyyssuunnassa (z-koordinaatti). Näin elementit voivat mennä välillä osin toistensa päällekin ja luoda taustalla oleville elementeille hienon varjostuksen. Lisäksi Materializella voi luoda pyöreän ”leijuvan” napin, jonka tarkoitus on tarjota sitä kautta tärkeimmät toiminnot. (Prusty 2015.)

Materializen avulla tehdyllä sivustolla pystyy hyvin erottautumaan massasta, koska sillä tehtyjä sivuja ei vielä ole paljon. Materialize soveltuu erittäin hyvin hybridimobiilisovelluksille. (Prusty 2015.)

4.2.4 Foundation Zurb

Foundation julkaistiin avoimen lähdekoodin versiona syyskuussa 2011. Se oli alun perin tyyliohje ZURB-suunnittelutoimistolle, jonka asiakkaita olivat mm. Facebook, eBay ja NYSE. Ohjelmistokehyksen nimikin (suomeksi ”perusta”) viittaa siihen, kuinka ohjelmistokehys toimii pohjana sivuston suunnittelussa. (Shiotsu 2016.)

Jos Foundationia verrataan Bootstrapiin pinnallisesti, voi näyttää, että niiden komponentit ja ominaisuudet ovat hyvin samankaltaisia. Erot löytyvät kuitenkin enemmänkin pinnan alta. Molempien ohjelmistokehysten käyttäessä CSS-esiprosessorina SASS:a, ohjelmistokehyksiä voidaan kustomoida kätevästi. Kuitenkin, jos haluaa tehdä hyvin yksilöityä designia, Foundation on tähän joustavampi. Molemmilla ohjelmistokehyksillä on erittäin hyvä tuki kaikille perusselaimille tietokoneissa ja mobiilissa. Suorituskyvyssä ja nopeudessa ei myöskään ole suuria eroja kehysten kesken. Foundationilla on Bootstrapin ohella riittävän laaja kehittäjäyhteisö, joten mahdollisiin ongelmiin yleensä löytyy ratkaisu. Foundation on siten vakaa ja hyvä valinta ohjelmistokehykseksi. Valinta kehyksen kesken tulee siis tehdä, kuten monessa muussakin, visuaalisen mieltymyksen mukaan. Jos kehykseltä haluaa joustavuutta, siihen Foundation on parempi valinta kuin Bootstrap. (Shiotsu 2016.) Foundationia ovat käyttäneet pohjana mm. PBS, National Geographic, Washington Post ja Mozilla. (Subbu 2017.)

Taulukko 7. Foundationin hyvät ja huonot puolet

Hyvää	Huonoa
- Hyvä tuki eri selaimille - Laaja kehittäjäyhteisö - Joustaa muokkauksille hyvin	-

4.2.5 Bulma

Bulma on Jeremy Thomasin vuonna 2016 kehittämä ohjelmistokehys ja sen lähdekoodi on avointa, kuten monen muunkin ohjelmistokehyksen. Bulman ilme on hyvin moderni ja sen rakenne pohjautuu flexboxiin, mikä tarjoaa kehittäjälle helpommin muokattavan gridin. (Sevilleja 2017; Roy 2018.) Roy (2018) listaa Bulman hyvät ja huonot puolet seuraavasti:

Taulukko 8. Bulman hyvät ja huonot puolet (Roy 2018; Sevilleja 2017.)

Hyvää	Huonoa
- CSS-tiedosto on hyvin kevyt ja helppo kustomoida - Pohjautuu flexboxiin, mikä mahdollistaa joustavamman suunnittelun - Moderni ulkoasu - Hyvin dokumentoitu	- Ohjelmistokehys on vielä kehitysvaiheessa (tilanne 1.3.2018) - CSS käy todella hitaasti Internet Explorerilla - Ei toimi Internet Explorer 10.0:aa vanhemmalla versiolla

Bulmaan ei sisälly ollenkaan JavaScriptiä, mutta se integroituu minkä tahansa JavaScript-kehyksen (esim. Angular, React) kanssa. Bulmalla on myös oma ohjeensa, jolla se voidaan integroida käyttämään Vue.js:ää. (Thomas ym. 2018, 9.)

4.2.6 Pure CSS

Pure CSS on Yahoon responsiivinen ohjelmistokehys, joka tarjoaa minimalistisen, mutta hyvin dokumentoidun vaihtoehdon. Pure tarjoaa muokattavan gridin, sisäänrakennetut vertikaalisen ja horisontaalisen valikon, alasvetovalikon, omat napit, joustavan lomakenäkymän, yleiset taulukkotyylit ja selkeän, minimalistisen näkymän, jota voi helposti laajentaa. Pure CSS:n suurin etu on sen todella pieni koko, ohjelmistokehys vie vain 3,8 kilotavua. Pure itse mainitsee ohjelmistokehyksen sisältävän vain tarpeellisimmat ulkoasumäärittelyt natiiveille HTML-elementeille sekä kaikkein yleisimmille käyttöliittymäkomponenteille. Tästä syystä ohjelmistokehystä voikin erityisesti suositella, jos sivustolla vieraillaan pääasiassa mobiililaitteilla. (Reifman 2016; Purecss 2018.)

Pure CSS:n ote ohjelmistokehyksen suunnittelussa on selkeästi ollut se, että Pure CSS tarjoaa vain yksinkertaisen, kuitenkin responsiivisen pohjan, jonka päälle web-kehittäjä itse tuottaa omaa tyyliä. Pure CSS mainitseekin

sivustollaan, että on paljon helpompaa kirjoittaa muutaman lisärivin CSS:ää kuin ylikirjoittaa valmiita tyylejä. (Purecss 2018.)

Taulukko 9. Pure CSS:n hyvät ja huonot puolet (Reifman 2016.)

Hyvää	Huonoa
- CSS-tiedosto on todella kevyt - Tarjoaa minimalistisen näkymän - Toimii hyvänä pohjana projektille	- Ei sovellu, jos tarvitsee laajempaa pakettia tai siinä tapauksessa on muokattava

4.2.7 Skeleton

Skeleton on Dave Gamachen kehittämä ohjelmistokehys ja keskittyy Pure CSS:n ohella olemaan mahdollisimman kevyt. Skeleton rakentuu kahdesta CSS-tiedostosta: suositusta normalize.css:stä ja skeleton.css:stä, joista jälkimmäinen tarjoaa ohjelmistokehyksen tyylit, jotka on rajattu noin 400 riviin pakkaamatonta CSS-koodia. Skeletonin tärkein osa on grid system sekä HTML:n peruskomponenttien tyylimäärittelyt. Myös Skeleton on kehitetty mobiililaitteiden ehdolla. Kehys toimii hyvin yleisimmillä selaimilla ja siihen voi rakentaa päälle SASS- ja LESS-laajennuksia. Kehys on hyvin yksinkertainen, mutta tarjoaa responsiiviset ominaisuudet. (Martsoukos 2015.)

Skeleton kertoo omilla sivuillaan, että kehys sopii pienempiin projekteihin tai jos projektissaan ei tarvitse laajempien ohjelmistokehysten kaikkia tarjoamia ominaisuuksia. Skeleton tarjoaa tyylit vain muutamiin HTML-elementteihin sekä gridin. (Skeleton 2018.)

Taulukko 10. Skeletonin hyvät ja huonot puolet (Martsoukos 2015.)

Hyvää	Huonoa
- CSS-tiedosto on kevyt - Toimii hyvänä pohjana projektille	- Ei sovellu, jos tarvitsee laajempaa pakettia tai siinä tapauksessa on muokattava

4.2.8 UIkit

UIkit on kevyt ja helppokäyttöinen ohjelmistokehys, jolla on selkeät luokkanimet ja valtavasti vaihtoehtoja. UIkitin tyyli on selkeä, raikas ja moderni. Pakettiin kuuluvat CSS- ja JavaScript-tiedostot. UIkitissa koostuu coresta, johon jo sisältyy kaikki tarvittavat perustoiminnot ja myös joitakin JavaScriptin tuomaa elävyyttä sivustolle. Coren päälle voi lisätä sivustolle tarvittavia erikoisempia komponentteja, vaikkapa HTML-editorin, jotka voi liittää oman projektin lisäksi. UIkit rakentuu LESS:llä, ja se on apuna, kun UIkitin sivustolla olevalla omalla työkalulla pääsee kustomoimaan komponentteja ja teemaa. (Letsch 2015.)

Taulukko 11. UIkitin hyvät ja huonot puolet (Letsch 2015.)

Hyvää	Huonoa
- Helppokäyttöinen - Selkeät luokkanimet - Runsaasti vaihtoehtoja - Erilaisia lisättäviä komponent-teja saatavilla	-

4.3 **Ohjelmistokehysten käyttöönotto**

Ohjelmistokehykset otetaan käyttöön joko lataamalla ohjelmistokehyksen tarjoamat omat css- ja js-tiedostot omalle palvelimelle ja liittämällä ne HTML-sivun head-osiossa osaksi omaa verkkosivustoa tai käyttämällä viittauksessa suoraa linkkiä ohjelmistokehyksen omalle sivustolle.
Vaihtoehto 1, ladatut tiedostot omalla palvelimella (Bootstrap-esimerkki):

```
<!doctype html>
<html>
<head>
<link rel="stylesheet" href="css/bootstrap.min.css">
<script src="js/jquery.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</head>
...
```

Vaihtoehto 2, tiedostot Bootstrapin omalla palvelimella:

```
<!doctype html>
<html>
<head>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css">
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></script>
</head>
...
```

Vaihtoehdoista ensimmäinen on toimintavarmuudeltaan parempi, sillä sivusto toimii, vaikka kolmannen osapuolen palvelut olisivat alhaalla, sillä data on tallennettuna omalle palvelimelle ja on siten aina käytettävissä.

Toisaalta linkkinä oleva kirjasto voi olla jo käyttäjän välimuistissa, jolloin latausaika lyhenee, kun samaa kirjastoa ei tarvitse ladata uudestaan. Palvelimelta voidaan myös siten hakea aina uusin versio ilman tiedoston päivittämistä. Toinen vaihtoehto toimii hyvin erityisesti paljon käytettyjen ohjelmistokehysten kohdalla, sen sijaan vähemmän käytettyjen ohjelmistokehysten kohdalla voisi olla parempi käyttää ensimmäistä vaihtoehtoa.

4.4 **Ohjelmistokehysten valintaperusteet**

Tehtäessä valintaa verkkosivun taustalla olevan kehyksen käytöstä, on tehtävä ensin päätös siitä, käytetäänkö ensinkään valmista ohjelmistokehystä vai ryhdytäänkö sivustoa rakentamaan kaikilta osin itse. Päätöksessä web-kehittäjän on pohdittava seuraavia asioita sivustonsa suhteen:

- Sivuston elinkaaren pituus
- Projektin kiireellisyys
- Sivustolle vaadittavat tekniset ratkaisut
- Kehittäjän resurssit ja osaaminen
- Visuaalisuus
- Vaatimukset selaintuesta

Näistä kolme ensimmäistä ovat siinä mielessä kriittisiä, että ne voivat ratkaista ylipäätänsä sen, tuleeko web-kehittäjän edes valita ohjelmistokehystä ratkaisukseen. Jos sivuston elinkaaren pituus on pitkä ja sitä tullaan päivittämään jatkuvasti, teknistä velkaa karttaen tulisi kirjoittaa oma CSS-tiedosto. Toisaalta projektin kiireellisyys voi puolestaan vaatia, että saadaan sivusto nopeasti pystyyn, jolloin ohjelmistokehys on parempi ratkaisu. Jos taas sivuston visuaalisen ilmeellä halutaan erottautua, ei tulisi käyttää ohjelmistokehyksiä. Kehittäjän omilla resursseilla ja osaamisella on luonnollisesti iso merkitys, osaako hän edes tehdä omia CSS-tiedostoja, jotka toimivat oikealla tavalla ja responsiivisesti. Näiden tietojen pohjalta voidaan luoda karkea jako siitä, tuleeko kehystä käyttää:

Taulukko 12. Milloin tulisi käyttää CSS-ohjelmistokehystä ja milloin ei?

Käytä CSS-ohjelmistokehystä	Älä käytä
- Sivuston lyhyt elinkaari - Projektilla on kiire - Ei ylimääräisiä vaativia teknisiä ratkaisuja - Kehittäjän vähäiset resurssit - Kehittäjän osaaminen ei korkea - Visuaalinen perusilme riittää	- Sivuston pitkä elinkaari - Projektilla ei kiire - Sivustolle luodaan vaativia teknisiä ratkaisuja - Kehittäjän resurssit hyvät - Kehittäjän osaaminen korkea - Haetaan erilaista visuaalista ilmettä - Harvinaiset selainvaatimukset (älä käytä tai valitse sopiva ohjelmistokehys)

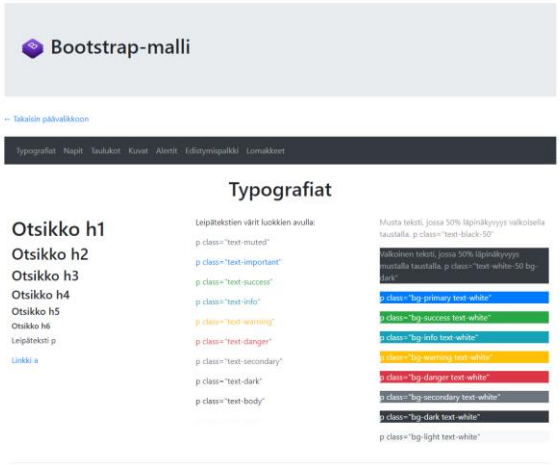
5 KEHYSTEN EROAVAISUUDET

Osana tätä tutkimusta rakensin edellä esitetyillä 8 ohjelmistokehyksellä mallisivuston, joka on saatavilla HAMK:n palvelimelta osoitteesta http://shell.hamk.fi/~antti1724/opinnaytetyo/index.html. Sivustot rakennetaan vain front-end-pohjalta, ja niissä ei kosketa back-end-puoleen. Sivustolla on esitetty jokaisesta ohjelmistokehyksestä ominaisuuksiltaan ja rakenteeltaan samankaltaiset sivut, joista on helppo nähdä visuaalisia eroja. Sivuilla on esiteltynä mm. seuraavia visuaalisia ominaisuuksia:

- Typografiat
- Napit
- Taulukot
- Kuvat
- Laatikot
- Ilmoitukset
- Viestit
- Lomakkeet ja kentät

Ohjelmistokehysten laajuuksissa on isoja eroja, joten kaikkiin kehyksiin yllä olevia ominaisuuksia ei ole määritelty tai osassa on näiden lisäksi vielä paljon muutakin. En kuitenkaan halunnut tehdä sivuista kaiken kattavia oppaita ohjelmistokehyksen käyttöön, vaan olevan niiden vain lyhyt kattaus kehyksen yleiseen näkymään, jotta ne voisivat olla tukena ohjelmistokehyksen valinnassa.

5.1 Ohjelmistokehysten visuaaliset ja toiminnalliset eroavaisuudet

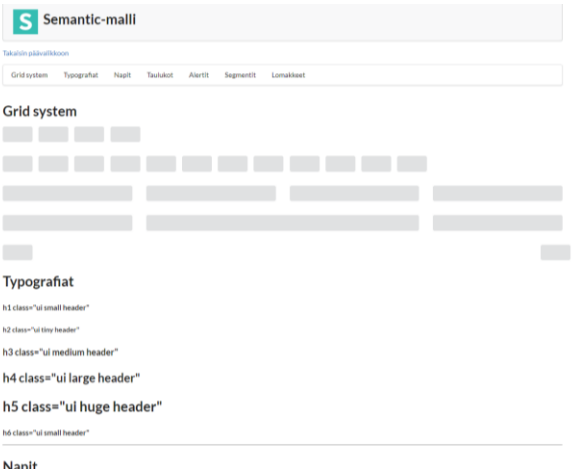


Kuva 7. Kuvakaappaus Bootstrapin mallisivusta

Bootstrap on ominaisuuksiltaan kattava paketti ja sisältää monessa tap

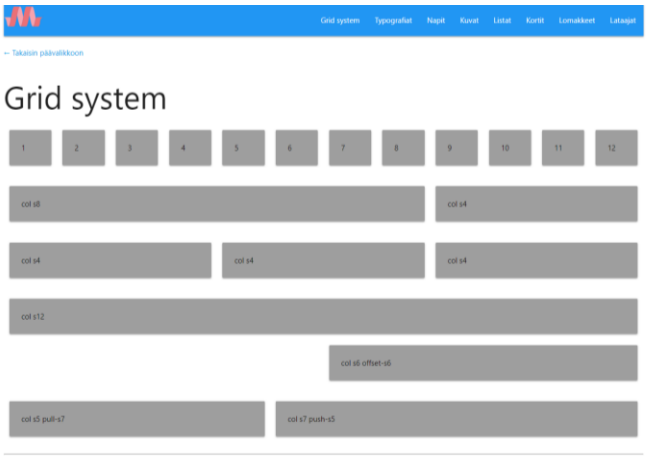
auksessa enemmän kuin tarpeeksi määrittelyjä sivustolle. Kehyksen käyttö oli minulle hyvinkin tuttua, joten uutta opettelua ei ollut ja sivun luominen oli helppoa. Bootstrapin omilla sivuilla on myös kattava opas siitä, kuinka eri osiota saa tehtyä ja oppaita Bootstrapin käyttöön tuntui löytyvän myös runsaasti.

Koska ohjelmistokehyksistä eniten käytetyin on Bootstrap, kerron seuraavana omia havaintojani muiden ohjelmistokehysten käytöstä vertaamalla niitä pääasiassa Bootstrapiin.



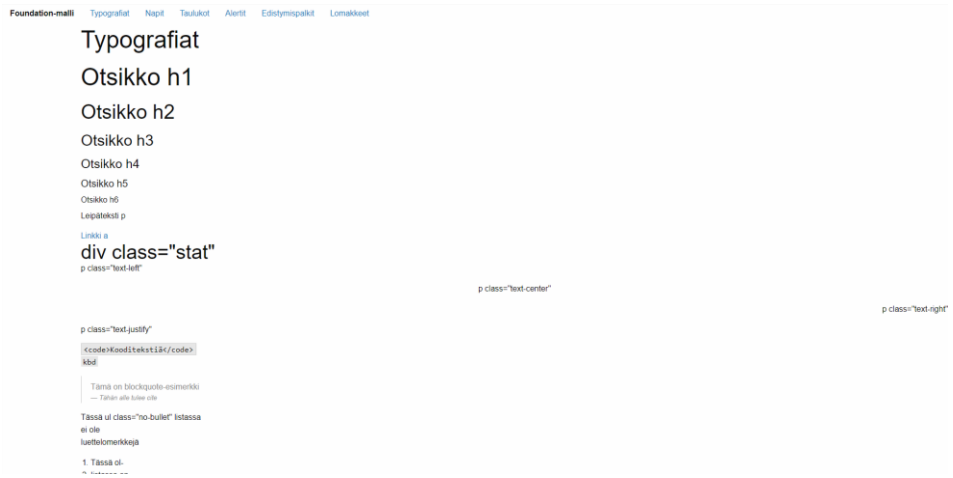
Kuva 8. Kuvakaappaus Semanticin mallisivusta

Pidin Semanticin ulkoasusta hyvin paljon. Sen typografia ja värit toivat miellyttävää vaihtelua ollen samalla kuitenkin hyvin selkeitä. Luokkien käytössä oli päädytty ratkaisuun, jossa jokainen näkyvä elementti aloitetaan sanalla ui. Luokkien käyttö oli toisaalta selkeää niiden ymmärrettävyytensä vuoksi, mutta hyvin erilaista verrattuna Bootstrapiin. Selkeys paistaa esille kaikessa Semanticin luokkien nimissä jopa niinkin paljon, että pelkkää lähdekoodiakin lukemalla voi kuvitella, minkälainen elementti saadaan aikaiseksi. Erilaisuutensa vuoksi alkuun pääsy vie kuitenkin aikansa, että eri luokkia oppii käyttämään niiden täydellä potentiaalilla. Semanticin omilla sivuilla on kuitenkin erittäin hyvät ja kattavat esimerkit elementtien käytöstä ja niiden avulla alkuun pääseminen helpottuu.



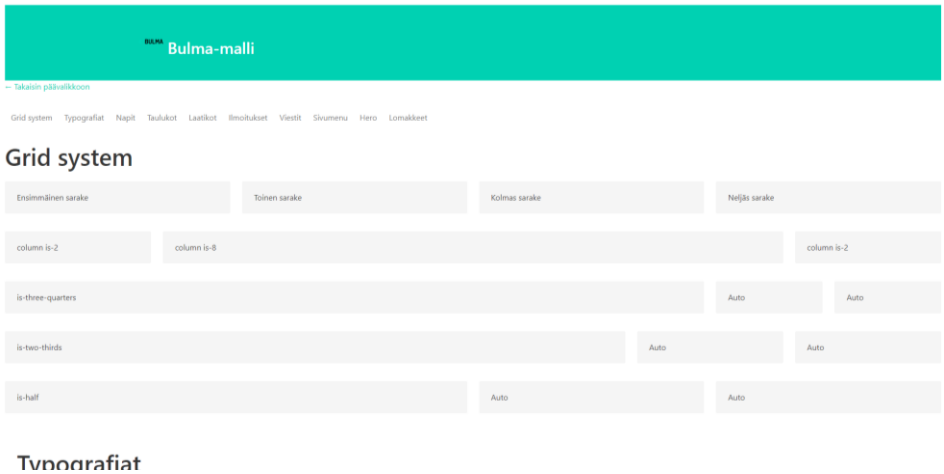
Kuva 9. Kuvakaappaus Materializen mallisivusta

Materialize noudattaa ulkoasussaan Googlen ulkoasua ja se näkyy. Tämä luo käyttäjälle tuttua näkymää ja on siten helppokäyttöinen liittymä. Oma tunne oli, että ohjelmistokehys sopisi erityisen hyvin web-pohjaiselle sovellukselle ja erityisesti mobiilikäyttöiselle web-sovellukselle. Materializen omalla JavaScript-tiedostolla saatiin hyviä lisäyksiä sivulle, joilla mallisivun toiminnallisuudet alkoivat muistuttamaan Android-käyttöliittymää. Itse luokkien nimet vaativat kehittäjältä opettelua, sillä ne poikkeavat monelta osin Bootstrapin vastaavista luokkanimistä, mutta Bootstrapista tietämätön oppinee luokkien käytön harjoittelemalla. Semanticiin verrattuna luokkien nimet eivät kuitenkaan olleet niin selkeitä. Toisaalta Materialize tarjosi verkkosivuillaan hyviä esimerkkejä elementtien käyttämisestä.



Kuva 10. Kuvakaappaus Foundationin mallisivusta

Foundation on ominaisuuksiltaan lähes yhtä kattava kuin Bootstrap. Samat tyylimäärittelyt löytyivät oikeastaan kaikkiin Bootstrapin tarjoamiin elementteihin, joten paketti on laaja. Eroavaisuudet löytyvät visuaalisuudesta: Foundation on kulmikkaampi ja värimaailma poikkeaa Bootstrapista.

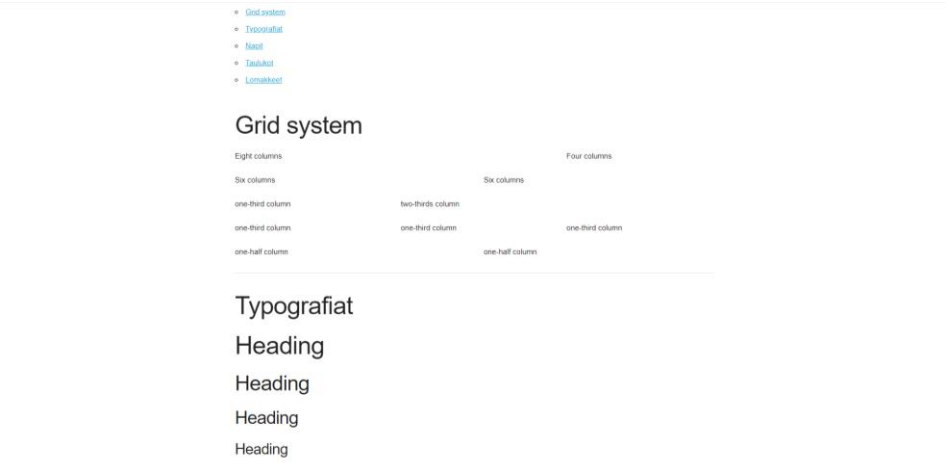


Kuva 11. Kuvakaappaus Bulman mallisivusta

Bulmassa tuntuma on hyvin samankaltainen kuin Bootstrapissa. Siinä on lähes kaikki samat elementit kuin Bootstrapissa ja luokkien nimetkin ovat monelta osin kuin kopioita Bootstrapin luokista, joskin ne ovat enemmän ihmiskieltä muistuttavassa muodossa. Esimerkiksi luokkien tarkenteissa on käytössä is-primary, is-info, is-success, is-warning ja is-danger, jotka ovat Bootstrapissa samat, paitsi is-alun tilalla on yleensä kyseiseen elementtiin viittaavaa tekstiä. Näin Bootstrapia käyttämään oppineet oppivat varmasti myös nopeasti Bulman käytön, ja silti, Bulman värimaailma ja pienet yksityiskohdat tekevät siitä erilaisen ja erottuvan. JavaScriptin puuttuminen Bulman paketista ei tässä mallisivulla näkynyt ja JavaScript-kehyksen saa aina halutessaan lisättyä mukaan, mikäli tarvetta on.

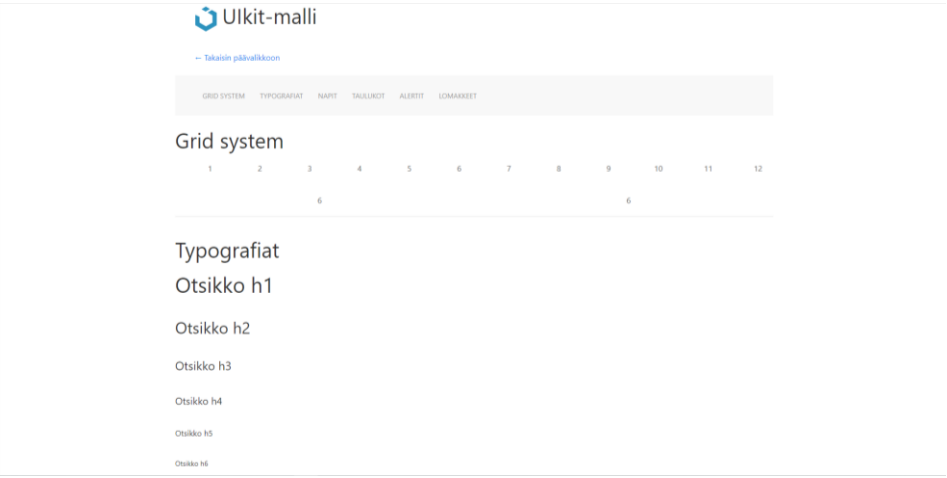
Kuva 12. Kuvakaappaus Pure.css:n mallisivusta

Pure CSS:ssä oli erittäin niukasti ominaisuuksia, mikä olikin ohjelmistokehyksen valitsema erottumistekijä. Mallisivu jäi siten varsin karun ja pelkistetyn näköiseksi, sillä tämän työn periaatteen mukaisesti en lähtenyt muokkaamaan sivuston css-tiedostoa. Kehystä käyttävän on kuitenkin huomioitava, että tätä kehystä käyttäessä css:ää on pakko muokata, jos sivustosta aikoo saada yhtään hienomman näköisen. Ohjelmistokehyksen omat sivut esittelivät ominaisuudet kohtalaisesti.



Kuva 13. Kuvakaappaus Skeletonin mallisivusta

Skeletonissa oli Pure CSS:n ohella hyvin vähän ominaisuuksia. Se oli ohjelmistokehyksistä ainoa, jossa ei ollut edes navigaatiopalkkia. Tosin omilla sivuillaan opastetaan, kuinka navigaatiopalkin voi luoda. Skeletonin ja Pure CSS:n ohjelmistokehysten mallisivujen luonti tuntui ominaisuuksien niukkuuden vuoksi jopa vähän turhalta, sillä näitä kehyksiä käyttävän tulisi lähes joka tapauksessa kuitenkin lisättävä oma tyylisivu, jolloin ulkoasu tulisi näyttämään täysin erilaiselta. Työni rajauksen mukaisesti en siihen kuitenkaan ryhtynyt.



Kuva 14. Kuvakaappaus UIkitin mallisivusta

UIkit oli alun opettelun jälkeen erittäin helppokäyttöinen kehys ja sen rakenteen luominen kaikkein nopeinta. Typografialtaan ja muotoilultaan kehys muistuttaa Bootstrapia, mutta värimaailma on hieman pehmeämpi. Uikit oli tutkimistani ohjelmistokehyksistä vähiten suosituin, mutta se ei aiheuttanut ongelmia uuden opettelun kannalta, sillä UIkitin omilla sivuilla oli erittäin hyvä opas kehyksen käyttöön.

Ohjelmistokehyksistä vaativin kehittäjälle tuntui oman näkemykseni mukaan olevan Pure CSS, sillä siinä omat CSS-taidot täytyy olla hyvät ja

kehyksen kustomointi erilaisin laajennuksin vaatii mielellään myös JavaScriptin osaamista. Sen sijaan Bootstrapia jo osaavalle luonteva askel osaamisen laajentamiseksi voisi hyvin olla Bulma, joka on ominaisuuksiltaan kattava paketti ja sen luokkanimet muistuttavat paljon Bootstrapia.

5.2 **Ohjelmistokehysten latausnopeudet**

Ohjelmistokehyksistä luodaan samankaltaiset testisivut, johon otetaan testiin ainoastaan ohjelmistokehyksen pohjarakenne eli joko grid system tai flexbox. CSS-tiedosto ladataan sekä omalta palvelimelta että ohjelmistokehyksen ensisijaisesti suosittelemalta kolmannen osapuolen palvelimelta. Jos ohjelmistokehys käyttää JavaScriptiä, tämä ominaisuus poistetaan testisivulta. Sivulle ei luoda mitään muuta elementtiä, vaan sen tarkoitus on vain osoittaa, kuinka ohjelmistokehyksellä luodun sivun pohja latautuu antaen osviittaa CSS-tiedoston latautumisesta ja HTML-kielen mahdollisesta kompleksisuudesta käytettävällä kehyksellä.

Latausnopeudet testauksessa käytetään ilmaista Pingdom-työkalua. Pingdom on sivusto, joka tarjoaa kokonaisarvostelun www-sivuston nopeudesta. Lisäksi se arvioi seuraavia sivuston osioille: suorituskyky, pyynnöt (requests), latausnopeus ja sivun koko. Sivustolla voidaan käyttää mittauksessa neljää eri Pingdomin palvelinsijaintia: Melbournea, New York Cityä, San Josea ja Tukholmaa.

Tässä mittauksessa käytetään seuraavia esivaatimuksia: palvelinsijainniksi asetetaan Tukholma. Sivusto ladataan Elisan kotisivupalvelimelle, josta testaukset tehdään. Tämä siitä syystä, että HAMK:n sisäiselle palvelimelle Pingdom ei pääse. Sivustot ladataan 10 kertaa ja otetaan saatujen latausnopeustulosten keskiarvot. Tämä tehdään kahdella vaihtoehdolla: CSS-tiedosto on Elisan palvelimella (”oma palvelin”) ja ulkoisella palvelimella. Muilla Pingdomin antamilla arvoilla ei tässä ole merkitystä, kun luodaan samankaltaiset sivustot, joissa sisältöä ei juurikaan ole. Testin tarkoitus on ensinnäkin nähdä CSS-tiedoston koon vaikutus nopeuteen, CSS-tiedoston luokkien rakentamisen selkeys HTML-elementeissä ja lisäksi, mikä vaikutus on sillä, mistä CSS-tiedosto haetaan.

Latausnopeuksien mittaustulokset ovat liitteessä 1. Alla olevassa taulukossa on tulosten keskiarvot ohjelmistokehyksittäin sekä kaikkien ohjelmistokehysten keskiarvot jaoteltuna CSS-tiedoston sijainnin mukaan.

Taulukko 13. Ohjelmistokehyksillä luotujen mittaussivujen latausnopeudet millisekunteina omalla palvelimella ja ulkoisella palvelimella olevalla css-tiedostolla. Mittaukset on suoritettu 5.7.2018 klo 12-14 välisenä aikana.

CSS-tiedosto:	omalla palvelimella	ulkoisella palvelimella
Bootstrap	72,7	49,5
Semantic UI	155,3	90,3
Materialize	66,5	55,0
Foundation Zurb	62,5	51,4
Bulma	70,5	61,7
Pure CSS	31,9	42,2
Skeleton	27,4	42,1
UIkit	88,6	57,4
Keskiarvo	71,9	56,2

Taulukon arvot ovat millisekunteja. Eroavaisuudet eivät ole suuria, mutta ladatut HTML-sivutkin olivat vain yhden rivin gridejä, joten ladattavaakaan ei ollut paljon. HTML-sivujen rakenteiden osalta ei juurikaan ollut eroja. Isommilla sivustoilla, jossa elementtejä olisi runsaammin, erot voisivat korostua enemmän. Siten nämä nopeudet kertovat hyvin pelkän CSS-tiedoston koon ja sijainnin vaikutuksen nopeuteen. HTML:n kompleksisuus ei tällä tutkimustavalla tullut esille pelkkää gridiä luotaessa.

Nopeimmin ohjelmistokehyksistä latautuivat Skeleton ja Pure CSS. Tämä ei yllätä, sillä kehyksillä olikin keveimmät css-tiedostot. Sen sijaan yllätyksenä tuli, että kyseiset kehykset latautuivat nopeammin, kun ne ladattiin omalta palvelimelta, eikä ulkoiselta palvelimelta. Muiden kehysten kohdalla tilanne oli täysin päinvastainen: ulkoiselta palvelimelta ladattu css-tiedosto oli parempi ratkaisu nopeuden kannalta kuin sen pitäminen omalla palvelimella. Syytä tähän poikkeavuuteen en saanut ratkaistua.

Laajoista kehyksistä nopeimmin latautui Bootstrap CSS-tiedoston sijaitessa ulkoisella palvelimella. Lähelle samaa lukemaa pääsivät Foundation, Materialize, UIkit ja Bulma. Selkeästi hitaimmin latautui Semantic. Semanticia hidastaa se, että CSS-tiedosto lataa ulkoisesta palvelimesta vielä ylimääräisen fonttimäärittelyn. Kehittäjälle, joka arvostaa sivuston nopeutta, tämä kannattaa ottaa huomioon. Kehystä käyttäessä voisi valita tutumman fontin tilalle.

Yleisesti ottaen latausnopeuksien keskiarvot kertovat siitä, että nopeuden korostuessa css-tiedosto kannattaa hakea ulkoiselta palvelimelta.

6 YHTEENVETO

Työni alkutavoitteena oli vastata seuraaviin kysymyksiin: miten eri ohjelmistokehykset eroavat toisistaan, mikä ohjelmistokehys webkehittäjän tulisi valita omaan projektiinsa, mitkä ovat ohjelmistokehysten käytön hyvät ja huonot puolet, kannattaako ohjelmistokehystä käyttää ja mitkä ovat suosituimmat ohjelmistokehykset.

Eroavaisuuksia ohjelmistokehyksissä oli paljon, erityisesti visuaalisia ja jonkin verran rakenteellisia, mutta myös samankaltaisuutta löytyi. Lähimpänä toisiaan olivat mielestäni Bootstrap ja Bulma. Bulma olisikin helpoin askel Bootstrapia osaavalle webkehittäjälle laajentaa osaamista muihin ohjelmistokehyksiin, sillä sen luokkanimet ja elementtien samankaltaisuudet muistuttavat paljon toisiaan. Ilme on silti erilainen.

Ohjelmistokehyksen valinnassa webkehittäjän tulee pohtia erityisesti visuaalista ilmettä ja toisaalta myös sivuston rakenteellisuutta ja päivitettävyyttä. Tärkeää on myös itse ohjelmistokehyksen tuki ja yhteisö, mikä korostuu vähemmän käytettyjen kehysten kohdalla. Latausnopeus korostuu, mikäli sivustoa selataan ensisijaisesti mobiilista. Mitatut latausnopeudet olivat pääasiassa paremmat, kun CSS-tiedosto sijaitsi ulkoisella palvelimella

CSS-ohjelmistokehysten käytössä voitiin nähdä monia hyviä puolia, joista erityisesti sivuston rakentamisen nopeus nousi tärkeimmäksi eduksi. Sivuston rakentamista aloittaessa on kuitenkin pohdittava monia tekijöitä siitä, kannattaako ohjelmistokehystä edes valita omaan projektiin, ja jos kannattaa, vaihtoehtoja on runsaasti. Kuitenkin nopeuttaakseen omaa projektia, voisi olla järkevää lähes joka tapauksessa valita pohjaksi edes minimaalisen responsiivisuuden tarjoama CSS-ohjelmistokehys. Ohjelmistokehyksen jatkuva kehitys, päivitykset ja kehittäjäyhteisö ovat tärkeitä tukipilareita websivuston rakentamiselle. Ohjelmistokehyksiä käyttäessä voi parhaimmillaan oppia itsekin toteuttamaan vastaavia ratkaisuja itse ja luomaan responsiiviset ominaisuudet juuri haluamallaan tavalla.

Ohjelmistokehyksissä nähtiin myös monia haittapuolia, joista suurimmat ongelmat olivat kehysten käytön vaikutus sivuston teknisiin ratkaisuihin sekä latausnopeuteen. Monesti kehys tuo ylimääräistä turhaa koodia, mikä hidastaa sivustoa. Toisaalta, monet ohjelmistokehykset tarjoavatkin tähän omaa ratkaisuaan, jossa paketista voidaan valita juuri ne halutut ominaisuudet, mitä tarvitsee, ja ominaisuudet yleensäkin ovat kustomoitavissa. On kuitenkin muistettava, että isoimmissa verkkosivustoissa ohjelmistokehystä ei kannata välttämättä edes valita, jos sivuston vaatimat tekniset ratkaisut ovat sen verran vaativia, että CSS-tiedostoa jouduttaisi jatkuvasti ylikirjoittamaan omilla tyylimäärittelyillä. Silti, näitäkin sivustoja on toteutettu, ja myös isoissa projekteissa.

On selvää, että kehykset ovat tulleet jäädäkseen, joten front-end-puolen kehittäjän tulisi hallita niiden käyttö, koska yhteisprojekteissa niiden osaamista voidaan hyvinkin vaatia, mikäli erityisesti halutaan tehdä sivusto tietyllä kehyksellä.

Opinnäytetyön tekeminen antoi minulle laajan kuvan siitä, kuinka valtavan suurta CSS-ohjelmistokehysten tarjonta on, vaikkakin se on hyvin polarisoitunutta muutamaan suosituimpaan kehykseen. Vähemmän käytettyjen kehysten käyttäminen oli haastavaa, sillä esimerkkejä ja oppaita oli vähän saatavilla. Opin matkan varrella paljon CSS:stä ja opin käyttämään muitakin ohjelmistokehyksiä kuin Bootstrapia. Tulevissa projekteissani tulen varmasti käyttämään näitä kehyksiä, joten työstä oli itselleni iso hyöty.

LÄHTEET

Andrew, P. (2011). Discussing the Pros and Cons of Using a CSS Framework. Blogijulkaisu 14.3.2011. Haettu osoitteesta https://speckyboy.com/discussing-the-pros-and-cons-of-using-a-css-framework/

Awwwards (2017). What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. Blogijulkaisu 20.2.2017. Haettu 28.2.2018 osoitteesta https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html

Belén, A. (2016). You might not need a CSS framework. Blogijulkaisu 21.4.2016. Haettu osoitteesta https://hacks.mozilla.org/2016/04/you-might-not-need-a-css-framework/

Bootstrap (2018a). About. Haettu 28.2.2018 osoitteesta https://getbootstrap.com/docs/3.3/about/

Bootstrap (2018b). Customize and download. Haettu 22.3.2018 osoitteesta https://getbootstrap.com/docs/3.3/customize/

Bootstrap (2018c). Grid system. Haettu 6.7.2018 osoitteesta https://getbootstrap.com/docs/4.0/layout/grid/

Borgen, P. (2017). The ultimate CSS battle: Grid vs Flexbox. Blogijulkaisu 11.12.2017. Haettu 6.7.2018 osoitteesta https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf

Bradford, L. (2017). Sass and LESS Preprocessors. Blogijulkaisu 13.11.2017. Haettu 19.4.2018 osoitteesta https://www.thebalancecareers.com/sass-vs-less-2071912

Chaffey, D. (2018). Mobile Marketing Statistics compilation. Blogijulkaisu 30.1.2018. Haettu 17.4.2018 osoitteesta https://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

Chishkala, I. (2018). Twitter Bootstrap vs. Semantic UI. Blogijulkaisu. Haettu 17.4.2018 osoitteesta https://www.upwork.com/hiring/development/twitter-bootstrap-vs-semantic-ui/

Cinarli, B. (2014). An Introduction to CSS Pre-Processors: SASS, LESS and Stylus. Blogijulkaisu 21.1.2014. Haettu 16.4.2018 osoitteesta https://htmlmag.com/article/an-introduction-to-css-preprocessors-sass-less-stylus

Cochran, D. (2012). Twitter Bootstrap web development how-to: a hands-on introduction to building websites with Twitter Bootstrap's powerful front end development front end framework. Birmingham, UK: Packt Publishing Ltd.

Cochran, D. & Whitley, I. (2014). Bootstrap Site Blueprints. Design mobile-first responsive websites with Bootstrap 3. Birmingham, UK: Packt Publishing Ltd.

Coron, T. (2017). What is Sass? Blogijulkaisu 5.7.2017. Haettu 16.4.2018 osoitteesta https://www.creativeblog.com/web-design/what-is-sass-111517618

Gerchev, I. (2014.) Introducing: Semantic UI Component Library. Blogijulkaisu 7.2.2014. Haettu 17.4.2018 osoitteesta https://www.sitepoint.com/introducing-semantic-ui-component-library/

Github (2018a). Front page. Haettu 28.2.2018 osoitteesta https://github.com

Github (2018b). About stars. Haettu 28.2.2018 osoitteesta https://help.github.com/articles/about-stars/

Github (2018c). Trending repositories in Github. Haettu 28.2.2018 osoitteesta https://github.com/trending

Heisler, Y. (2016). Mobile internet usage surpasses desktop usage for the first time in history. Blogijulkaisu 2.11.2016. Haettu 17.4.2018 osoitteesta http://bgr.com/2016/11/02/internet-usage-desktop-vs-mobile/

Horek, K. (2014). *Learning Zurb Foundation.* Birmingham, UK: Packt Publishing Ltd.

Jordon, N. (2018). Move over Bootstrap and Foundation, welcome Semantic UI. Blogijulkaisu 24.1.2018. Haettu 5.4.2018 osoitteesta https://coderwall.com/p/ham3gg/move-over-bootstrap-and-foundation-welcome-semantic-ui

Korpela, J. (2011). *HTML5. Uudet ominaisuudet.* Jyväskylä: WSOYpro Oy.

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

LePage, P. (2018). Responsive Web Design Basics. Blogijulkaisu 3.1.2018. Haettu 19.4.2018 osoitteesta https://developers.google.com/web/fundamentals/design-and-ux/responsive/

Letsch, F. (2015). UIkit web framework #1: Overview and Introduction. YouTube-video 21.3.2015. Haettu 19.4.2018 osoitteesta https://www.youtube.com/watch?v=JxplDTY65oY

Maheedharan, V. (2017). CSS Preprocessors - Powerful Tools for Smarter Styling of Web Pages and User Interfaces. Blogijulkaisu 15.2.2017. Haettu 19.4.2018 osoitteesta https://www.cabotsolutions.com/2017/02/css-preprocessors-powerful-tools-smarter-styling-web-pages-user-interfaces

Marcotte, E. (2011). *Responsive web design.* New York: A Book Apart.

Martsoukos, S. (2015). Getting Started with Skeleton, the Simple CSS Boilerplate. Blogijulkaisu 21.1.2015. Haettu 17.4.2018 osoitteesta https://www.sitepoint.com/getting-started-with-skeleton-simple-css-boilerplate/

Mozilla (2018). What is JavaScript? Haettu 9.4.2018 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

Prechu, P. (2018). 100+ Best CSS Frameworks for Responsive Design. Blogijulkaisu 21.2.2018. Haettu 28.2.2018 osoitteesta https://cssauthor.com/css-frameworks/

Prusty, N. (2015). Make Material Design Websites with the Materialize CSS Framework. Blogijulkaisu 9.2.2015. Haettu 17.4.2018 osoitteesta https://scotch.io/tutorials/make-material-design-websites-with-the-materialize-css-framework

Purecss. (2018). Haettu 17.4.2018 osoitteesta https://purecss.io

Reifman, J. (2016). Pure.css Offers an Intriguing Alternative to Bootstrap. Blogijulkaisu 21.7.2016. Haettu 17.4.2018 osoitteesta https://code.tutsplus.com/tutorials/purecss-offers-an-intriguing-alternative-to-bootstrap--cms-25176

Roy (2018). Top 5 Popular CSS Frameworks of 2018. Blogijulkaisu 17.1.2018. Haettu 28.2.2018 osoitteesta https://www.markupbox.com/blog/top-5-popular-css-frameworks-of-2017/

Sevilleja, C. (2017). Get to Know Bulma: My current Favorite CSS Framework. Blogijulkaisu 21.9.2017. Haettu 17.4.2018 osoitteesta https://scotch.io/bar-talk/get-to-know-bulma-my-current-favorite-css-framework

Shaleynikov, A. (2017). Top 5 Most Popular CSS Frameworks that You Should Pay Attention to in 2017. Blogijulkaisu 23.10.2017. Haettu

28.2.2018 osoitteesta https://hackernoon.com/top-5-most-popular-css-frameworks-that-you-should-pay-attention-to-in-2017-344a8b67fba1

Shillcock, R. (2013). All About Grid Systems. Blogijulkaisu 22.8.2013. Haettu 19.4.2018 osoitteesta https://webdesign.tutsplus.com/articles/all-about-grid-systems--webdesign-14471

Shiotsu, Y. (2016). Bootstrap vs. Foundation: Which Framework Is Right for You? Blogijulkaisu vuodelta 2016. Haettu 17.4.2018 osoitteesta https://www.upwork.com/hiring/development/bootstrap-vs-foundation-which-framework-is-right-for-you/

Sitecake (2018). CSS Frameworks. Haettu 17.4.2018 osoitteesta https://sitecake.com/resources/css-frameworks.html

Skeleton (2018). Skeletonin kotisivut. Haettu 5.7.2018 osoittesta http://getskeleton.com/

StatCounter (2018). Desktop vs Mobile vs Tablet Market Share World-wide. Haettu 17.4.2018 osoitteesta http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

Subbu (2017). Steb-by-step guide to Foundation framework to develop responsive web applications. Blogijulkaisu 9.10.2017. Haettu 17.4.2018 osoitteesta https://valuebound.com/resources/blog/develop-responsive-web-applications-using-foundation-framework

Thomas, J., Potiekhin, O., Lauhakari, M., Shah, A., Berning, A. (2018). *Creating interfaces with Bulma*. Haettu 17.4.2018 osoitteesta https://bleedingedgepress.com/book_excerpts/01E9D1/creating_interfaces_with_bulma_sample.pdf

Tutorialspoint (2018). What is CSS? Haettu 9.4.2018 osoitteesta https://www.tutorialspoint.com/css/what_is_css.htm

W3C (2018). HTML & CSS. Haettu 11.4.2018 osoitteesta https://www.w3.org/standards/webdesign/htmlcss

W3Schools (2018). HTML5 New Elements. Haettu 15.4.2018 osoitteesta https://www.w3schools.com/html/html5_new_elements.asp

Liite 1

Ohjelmistokehysten latausnopeuksien tulokset

	Latausaika, ulkoinen	Latausaika, sisäinen	Mittauskerta, tulos millisekunteina																			
			Ulkoinen palvelin										Oma palvelin									
Ohjelmistokehys	palvelin ka.	palvelin ka.	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Bootstrap	49,5	72,7	42	39	82	42	45	40	68	41	56	40	67	74	81	82	70	70	71	73	74	65
Semantic UI	90,3	155,3	110	87	83	83	86	93	80	98	104	79	149	156	179	141	140	157	157	152	171	151
Materialize	55,0	66,5	59	54	79	47	47	56	50	48	57	53	69	83	69	63	61	62	64	63	67	64
Foundation	51,4	62,5	62	56	46	47	61	57	45	43	49	48	66	63	59	61	65	62	61	61	62	65
Bulma	61,7	70,5	65	53	54	80	68	71	51	56	60	59	66	71	68	68	74	67	71	69	86	65
Pure CSS	42,2	31,9	43	53	42	39	39	42	42	41	43	38	35	31	31	31	33	31	32	30	31	34
Skeleton	42,1	27,4	43	41	45	39	43	47	46	36	40	41	28	33	27	24	32	27	28	27	24	24
Uikit	57,4	88,6	47	60	61	63	60	51	59	61	55	57	81	100	88	85	78	82	87	80	116	89
Keskiarvo	**56,2**	**71,9**																				