Responsiivisen WordPress-teeman luonnin kannattavuus

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, Tietojenkäsittelyn koulutusohjelma

Kevät 2019

Emma Helenius

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki

Tekijä Emma Helenius **Vuosi** 2019

Työn nimi Responsiivisen WordPress-teeman luonnin kannattavuus

Työn ohjaaja/t Tommi Saksa

TIIVISTELMÄ

Opinnäytetyössä tutkitaan responsiivisen WordPress-teeman luomista ja käytännöllisyyttä asiakasyritykselle toteutettavilla kotisivuilla. WordPress-teemaa päätettiin lähteä luomaan, sillä yritys oli jo valinnut WordPress-sisällönhallintajärjestelmän mutta ei ollut tyytyväinen sivuston ulkonäköön. Teemasta haluttiin yksinkertainen, mutta erilaisilla laitteilla sopiva. Siksi lähdettiin luomaan responsiivista sivustoa.

Työn tavoite on selvittää, miten responsiivinen WordPress-teema luodaan ilman että käytetään toista teemaa pohjana, sekä kuinka käytännöllistä sellainen on luoda. Aihetta tutkittiin luomalla oma WordPress-teema asiakasyrityksen kotisivuille.

Työn teoriaosuudessa kuvataan erilaisia web-tekniikoita, sisällönhallintajärjestelmiä ja responsiivisuutta yleisellä tasolla. Web-tekniikoista keskitytään HTML-, CSS-, PHP- ja JavaScript-kieliin. Paremmin kuvaillaan myös WordPress ja sen teemojen luonti. Käytännön osuudessa kerrotaan, kuinka WordPress-teema on suunniteltu ja luotu asiakasyritykselle.

WordPress-teemojen luonti osoittautui yksinkertaiseksi ja suoraviivaiseksi. Suurimmaksi ongelmaksi osoittautui se, että sekä WordPressista että Bootstrapista luodaan jatkuvasti uusia versioita. Uudet versiot eivät aina ole täysin yhteensopivia vanhojen versioiden kanssa, joten uuteen päivittäminen saattoi vaatia suuriakin muutoksia tiedostoissa. Isoimmaksi syyksi oman WordPress-teeman luontiin osoittautui se, että teemasta saa kevyemmän ja juuri sellaisen kuin haluaa.

Avainsanat WordPress, sisällönhallintajärjestelmä, Bootstrap, responsiivisuus

Sivut 24 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Business Information Technology
Visamäki

Author Emma Helenius **Year** 2019

Subject The Viability of Creating a WordPress theme

Supervisor Tommi Saksa

ABSTRACT

The thesis examines the creation and practicality of a responsive WordPress theme on the client company's website. It was decided that a WordPress theme would be created, because the company already had selected WordPress Content Management System but had not been happy with the appearance of the site. A simple theme was desired, but it should be suitable for different devices. That is why a responsive site was created.

The aim of this thesis was to find out how to create a responsive WordPress theme without using another theme as a basis, and how practical it is to create your own theme from scratch. The topic was studied by creating a personal WordPress theme for the client company's web site.

The theoretical part of the thesis describes various web technologies, content management systems, and responsiveness on a general level. Web technologies focus on HTML, CSS, PHP, and JavaScript. WordPress and its themes are better described as well. The practical section explains how the WordPress theme was designed and created.

Creating WordPress themes turned out to be quite simple and straightforward. The biggest problem turned out to be the new versions of both WordPress and Bootstrap. New versions are not always fully compatible with old versions, so new upgrades could require a lot of changes in the files. The biggest reason for creating your own WordPress theme turned out to be that the theme is lighter and is exactly the way wanted.

Keywords WordPress, content management system, Bootstrap, responsivity

Pages 24 pages

SISÄLLYS

1 JOHDANTO

Työn toimeksiantaja on asiakasyritys, joka on toivonut verkkosivu-uudistusta. Sisällönhallintajärjestelmäksi oli valittuna WordPress, mutta muuten projekti ei ollut vielä edistynyt, joten lähdettiin luomaan uudistettuja sivuja.

Projekti kokonaisuudessaan sisältää sivustojen ja sisällön luonnin sekä yrityksen konsultoinnin sivuston käyttöönotossa, mutta opinnäytetyö keskittyy WordPress-teeman luonnin eri vaiheisiin.

Verkkosivuista oli toivottu vanhemmalle väelle sopivia, mutta monella laitteella helposti toimivia. Bootstrap-framework päätettiin osaksi teeman luontia. Näin pystytään luomaan responsiiviset sivustot, jotka ovat helppokäyttöiset kaikenkokoisilla näytöllä.

Tämän työn teoriaosuudessa käsitellään WordPressin yleisiä tietoja, käyttötarkoituksia ja ominaisuuksia sisällönhallinnan ja kehittämisen kannalta. Tämän lisäksi kerrotaan yleistä tietoa niin WordPress-sisällönhallintajärjestelmästä, WordPress-teemoista, kuin Bootstrap-frameworkista.

Käytännön osuudessa WordPress-teeman luonti käydään kohta kohdalta läpi alkaen aloitusasennuksista ja käyttämällä niitä ratkaisuja, joita asiakasyritykselle tehtävässä teemassa käytetään. Loppupuolella myös alusta asti tehdyn teeman luonnin hyvät ja huonot puolet käydään läpi, verraten niitä valmiin teeman muokkaamiseen.

Työn tarkoituksena on tutkia, onko käytännöllistä luoda oma WordPress-teema en sijaan, että muokkaisi jo valmista teemaa. Tämä toteutetaankin luomalla asiakasyritykselle WordPress-teema ja vertailemalla omia kokemuksia teoriaan.

Tämä opinnäytetyö pyrkii helpottamaan valintapäätöstä oman teeman ja valmiin teeman välillä. Se myös pyrkii kuvailemaan oman WordPress-teeman luonnin, jotta tulevaisuuden teeman luonnit olisivat helpompia, nopeampia ja vaivattomampia.

2 WEB-TEKNIIKAT

Verkon ohjelmointi voidaan jakaa kahteen osa-alueeseen: asiakaspuolen ja palvelinpuolen ohjelmointiin. Asiakaspuolella tarkoitetaan sitä päätelaitetta, jolla palvelua käytetään. Palvelinpuolella tarkoitetaan niiden sovelluksien ohjelmointia, jotka toimivat www-, sovellus- tai tietokantapalvelimilla. (Keränen, Lamberg & Penttinen 2006, 30.) Tässä opinnäytetyössä web-tekniikoilla tarkoitetaan juuri näitä eri merkintäkieliä, joilla verkon ohjelmointi toteutetaan.

Asiakaspuolen julkaisukieliä ovat esimerkiksi HTML ja CSS. Niillä luodaan käyttöliittymä ja sen toiminnollisuus. Julkaisukielet ovat merkintäkieliä, joilla kuvataan dokumenttien rakennetta (Keränen ym. 2006, 30). Merkintäkielet koostuvat aineistoon lisättävistä tunnisteista (Tieteen termipankki 2014). Usein tämä tapahtuu sijoittamalla aloittavan elementin ja lopettavan elementin väliin vaikutettavan sisällön. Esimerkiksi HTML-kielessä saadaan tekstille kappalemuotoilu sijoittamalla aloittavan elementin `<p>` ja lopettavan elementin `</p>` väliin vaikutettava sisältö, eli tässä tapauksessa jokin tekstikappale. (Keränen ym. 2006, 30.)

Palvelinpuoli taas toteutetaan ohjelmointikielillä, kuten PHP. Ohjelmointikielillä ohjelmoidaan ne toiminnot, joihin julkaisukielet eivät sovi. Esimerkiksi voidaan hallinnoida lomakkeita, sähköposteja, hakuja ja tietokantoja. (Keränen ym. 2006, 30.) Tämän lisäksi osana web-tekniikoita ovat myös selainpohjaiset ohjelmointikielet, kuten JavaScript, sekä tietokannat, kuten MySQL. (Keränen ym. 2006, 30.)

2.1 HTML

HTML (HyperText Markup Language) on merkintäkieli, jonka avulla voi luoda verkkosivuja ja niiden elementtejä (W3C 2017). HTML-kieltä on suhteellisen helppo oppia käyttämään ja se antaa laajat luontimahdollisuudet. Sitä päivitetään jatkuvasti W3C:n toimesta (Shannon 2012).

HTML-tunnisteiden avulla kerrotaan nettiselaimille nettisivun rakenne ja miltä nettisivu pitäisi näyttää. Nämä tunnisteet luodaan yhdistelemällä tavallista tekstiä ja tiettyjä erikoismerkkejä. (Noble & Tittel, 4.)

Yksinkertainen HTML-sivusto voisi esimerkiksi olla seuraavanlainen:

```
<html>
<head>
<title>Sivun nimi</title>
</head>
<body>

<h1>Otsikko</h1>
```

```
<p>Ensimmäinen tekstikappale.</p>
<p>Toinen tekstikappale.</p>

</body>
</html>
```

Koodi 1. *Esimerkki yksinkertaisesta HTML-sivustosta.*

Tämä esimerkki (koodi 1) alkaa html-tunnisteesta ja loppuu html-tunnisteeseen. Alun html-tunniste, `<html>`, on aloitustunniste, ja lopun html-tunniste, `</html>`, on lopetustunniste. Näiden aloitus- ja lopetustunnisteiden väliin tulee sisältö, johon halutaan vaikuttaa. Tunnisteiden ja sisällön yhdistelmää kutsutaan elementiksi. Aloitus- ja lopetustunnisteet erottaa HTML:ssä vinoviivasta (/) ennen tunnisteen nimeä, niin kuin esimerkin `<html>` ja `</html>` -tunnisteet ovat eroteltu. (Larsen, 4.)

Samaa aloitus- ja lopetustunnisteen logiikkaa voidaan käyttää myös muissa elementeissä. Esimerkiksi body-elementti alkaa `<body>`-tunnisteella, sen sisältönä on verkkosivulla näkyvä sisältö ja se loppuu `</body>`-tunnisteeseen.

2.2 **CSS**

CSS (Cascading Style Sheets) antaa mahdollisuuden luoda elementeille sääntöjä, joiden mukaan ne näkyvät verkkosivuilla. Yksi sääntö voidaan jakaa kahteen osaan, valitsimeen ja deklaraatioon. Näistä valitsin kertoo mihin elementtiin viitataan, ja deklaraatio kertoo miten kyseinen elementti pitäisi muotoilla. Deklaraation sisällä voidaan määrittää erilaisten elementin ominaisuuksien arvoja. (Larsen, 192.)

CSS voi esimerkiksi sisältää seuraavanlaisen säännön:

```
h1 { color: white; }
```

Valitsin, eli tässä h1, kertoo että halutaan muokata otsikkoa. Aaltosulkeen sisällä on deklaraatio (color: white;) joka kertoo mitä h1-elementille halutaan tehdä. Valittu ominaisuus, color, kertoo että halutaan muokata väriä. Arvoksi on annettu white, eli otsikon (h1) väri (color) halutaan vaihtaa valkoiseksi.

CSS ei yksinään tee oikeastaan mitään, vaan se täytyy aina yhdistää HTML-sivustoon. CSS-sääntöjä on mahdollista upottaa suoraan HTML-sivustoon, mutta kätevämpi tapa on luoda erillinen .css-päätteinen tyylitiedosto. Tämän tyylitiedoston voi sitten liittää HTML:ään erillisellä elementillä. (Larsen, 193.)

Yksinkertaiseen esimerkkiin tarvitaan siis kaksi tiedostoa, sivusto.html ja tyylitiedosto.css. Muokataan siis aikaisemmasta HTML-esimerkistä sopiva:

```
<html>
<head>
<title>Sivun nimi</title>
<link    rel="stylesheet"    href="tyyli-
tiedosto.css">
</head>
<body>

<h1>Otsikko</h1>
<p>Ensimmäinen tekstikappale.</p>
<p>Toinen tekstikappale.</p>
<p   class=”kolmas”>Kolmas   tekstikappa-
le.</p>

</body>
</html>
```

Koodi 2. *HTML-sivusto, sivusto.html, johon on lisätty CSS-tyylitiedosto.*

Nyt sivustolle (koodi 2) on siis lisätty tyylitiedosto, sekä kolmas tekstikappale. Tyylitiedosto saadaan liitettyä omalla elementillään:

```
<link   rel="stylesheet"   href="tyylitie-
dosto.css">
```

Näin selaimet osaavat hakea tyylit kohteesta tyylitiedosto.css. Sen sijaan kolmas tekstikappale lisättiin havainnollistamaan mahdollisuutta yksilöidä muotoilua eri elementeille paremmin luokkien (class) avulla. Kuvitellaan vaikka, että esimerkin (koodi 2) kaikista tekstikappaleista halutaan muuten samanlaisia, mutta viimeisen kappaleen tekstin taustasta halutaan punainen. Se onnistuisi vaikkapa seuraavalla tyylitiedostoesimerkillä:

```
body { background-color : white; }

h1 { color: blue; }

p { color: black; }

p.kolmas { background-color: red; }
```

Koodi 3. *Tyylitiedosto.css -esimerkki.*

Tässä esimerkissä (koodi 3) sivun tausta on valkoinen, otsikko sininen, tekstit mustia, sekä kolmannen tekstikappaleen tekstin tausta punainen. Kolmanteen tekstikappaleeseen on viitattu käyttämällä hyväksi valitsimessa

p-tunnistimen kolmas-luokkaa. Kolmas-luokalla on voitu luoda p-elementti joka on erilainen kuin muut p-elementit. Lisäämällä `class="kolmas"` p-tunnistimeen sivusto.html:ssä (koodi 2) käytetään niitä muotoiluja jotka kerrotaan tyylitiedosto.css:ssä (koodi 3) p.kolmas-valitsimen deklaraatiossa.

2.3 **PHP & MySQL**

PHP (PHP HyperText Preprocessor) on yksi suosittu palvelinsovellusten ohjelmointikieli. PHP:n erottaa sen aloitus- ja lopetustunnisteesta <?php ja ?>. Sitä käytetään erityisesti dynaamisten web-sivujen luonnissa. Sillä voidaan luoda monia sellaisia ominaisuuksia, joihin HTML tai CSS eivät kykene. Esimerkiksi PHP:llä voidaan luoda ehtolauseita. Tätä ominaisuutta käytetäänkin paljon hyväksi sellaisilla verkkosivuilla, joissa sisältö halutaan helposti hakea useilla eri tavoilla näytettäväksi.

Luodessa verkkosivuja PHP:llä tarvitsee hakea tietoa useita kertoja näytettäväksi sivustolle. Tätä varten käytetään MySQL:ää. MySQL varastoi tietoa ja PHP:hen yhdistettynä luo täysin toimivan verkkosovelluksen. (Suehring & Valade, 12.)

PHP:n ja MySQL:n yhdistelmä on suosittu monestakin syystä. Näitä syitä ovat muun muassa ilmaisuus, helppous, nopeus, muokattavuus, yhteensopivuus, sopivuus verkkosivujen luontiin, sekä laaja tuki. (Suehring & Valade, 13.)

2.4 **JavaScript**

HTML kertoo selaimelle miten sisältö järjestellään, CSS kertoo sisällön tyylin ja ulkonäön, ja JavaScript kertoo miten sivusto käyttäytyy ja toimii. Esimerkiksi JavaScriptillä voi saada tekstin vaihtamaan väriä klikattaessa. (Suehring & Valade, 11.)

JavaScript on selainpohjainen ohjelmointikieli, joten sen toimivuus on riippuvainen selainohjelmasta. Sitä käytetään toiminnallisuuden ohjelmointiin, kun julkaisukieli ei sitä pysty toteuttamaan. Esimerkiksi lomakekenttien tarkistus ja selainikkunan avaaminen ovat tällaisia toimintoja. Myös JavaScriptiä voi kirjoittaa sekä HTML-tiedostoon että ulkoiseen Ja-vaScript-tiedostoon. (Keränen ym. 2006, 33.)

JavaScript upotetaan HTML-tiedostoon script-elementillä (esimerkki alla). Sen voi sijoittaa minne vain head- tai body-elementtien sisään, ja näitä script-elementtejä voi olla useita. (Suehring & Valade, 189.)

```
<script>
Tähän JavaScriptiä.
</script>
```

Jos haluaa tehdä ulkoisen JavaScript-tiedoston, tulee HTML-tiedostoon luoda erilainen script-elementti (alla) ja luoda ulkoinen JavaScript-tiedosto (tässä tapauksessa nimellä toiminnot.js). (Suehring & Valade, 190.)

```
<script src="toiminnot.js"></script>
```

2.5 **jQuery**

JavaScript-kirjastot ovat kokoelma valmista JavaScript-koodia, joiden avulla voi helposti lisätä toiminnallisuutta ja nopeuttaa sivustojen luontia. jQuery on yksi tällainen kirjasto. (Suehring & Valade, 219.)

Kun kirjoittaa JavaScript-koodia, voidaan johonkin tiettyyn elementtiin viitata getElementById-metodilla, joka hakee elementin erikseen merkityn tunnisteen (id) avulla. jQuerystä kuitenkin löytyy valmiita valitsimia, jotka mahdollistavat paljon paremmat mahdollisuudet hallita sivua JavaScriptillä. (Suehring & Valade, 219.)

jQueryn hyviä puolia on, että se toimii pelkkää JavaScriptiä todennäköisemmin samalla tavalla useammalla eri selaimella. Eri selaimet tulkitsevat JavaScriptiä eri tavoin, mutta jQueryn toiminnot selvittävät mikä selain on käytössä ja laittaa eri JavaScriptin näkymään eri selaimilla samalla tavalla.

3 SISÄLLÖNHALLINTAJÄRJESTELMÄT

Sisällönhallintajärjestelmä on ohjelmistokokonaisuus tai tietojärjestelmä, jonka tarkoituksena on hallita sisältökokonaisuuksia. Sen avulla käyttökelpoinen järjestys säilyy ja tiedot voidaan halutessa järjestää uudelleen ja ottaa helposti järjestelmän kautta eri käyttötarkoituksiin. (Jaakohuhta 2011, 112.) Sisällönhallintajärjestelmille ei kuitenkaan ole kovin selkeää määritelmää ja sillä saatetaan tarkoittaa eri tavoin painottuneita tietojärjestelmiä toimialasta riippuen. (Web-opas 2012.)

Tässä opinnäytetyössä viitataan sisällönhallintajärjestelmiin, joilla voidaan luoda, muokata, järjestellä ja julkaista sisältöä internetissä. Esimerkiksi WordPress-sisällönhallintajärjestelmällä voi luoda ja julkaista sisältöä netissä. Kaikki sisällönhallintajärjestelmät eivät kuitenkaan ole tällaisia, vaan voivat olla tarkoitettu sisällönhallintaan esimerkiksi intranetin tai vain yhden tietokoneen sisäisesti. (Wpbeginner n.d.)

Monen sisällönhallintajärjestelmän tarkoituksena on, että verkkosivujen ylläpito sekä sisältöjen luominen ja muokkaaminen onnistuisivat selaimella ilman ohjelmointitaitoja. (Omni Partners n.d.) Näin mainostavat itseään myös kolme suosituinta verkkosivujen sisällönhallintajärjestelmää (W3Techs 2017), WordPress (WordPress n.d.c.), Joomla (Joomla n.d.) ja Drupal (Drupal n.d.).

Muita suosittuja verkkosivujen sisällönhallintajärjestelmiä ovat esimerkiksi:

- Magento
- Blogger
- Shopify
- TYPO3
- Bitrix
- Squarespace
- PrestaShop
- Adobe Dreamweaver (W3Techs 2017).

Tässä opinnäytetyössä keskityn WordPress-sisällönhallintajärjestelmään. Toimeksiantaja oli jo valinnut sen pohjakseen projektin alkaessa, ja se on tällä hetkellä käytetyin verkkosivujen sisällönhallintajärjestelmä (W3Techs 2017). Itsekin halusin päästä tutustumaan WordPressiin ja sen ominaisuuksiin syvällisemmin, joten päätin sen olevan myös itselleni sopiva aihe.

4 WORDPRESS-SISÄLLÖNHALLINTAJÄRJESTELMÄ

WordPress on 2003 aloitettu, avoimen lähdekoodin sisällönhallintajärjestelmä, jolla voi luoda erilaisia verkkosivuja, blogeja ja sovelluksia (WordPress n.d.b). Se on luotu PHP:lla ja MySQL:llä, ja nykyään ylläpitää lähes neljännestä internetistä. WordPressin palvelut on jaettu kahteen osa-alueeseen, WordPress.com:iin ja WordPress.org:iin. Näiden suurin ero on se, että .com on vähemmän muokattavissa, mutta sivustoa ei tarvitse asentaa vaan se on heti käyttövalmis. Sen sijaan .org tarjoaa täydet muokkausmahdollisuudet, mutta WordPress täytyy asentaa omaan webhotelliin. (WordPress n.d.a.) Tässä projektissa käytetään .org:in palveluja, jotta pystytään luomaan oma teema.

4.1 WordPressin käyttötarkoituksia

Alun perin WordPress on luotu blogeja varten, ja nykyäänkin se on hyvin suosittu blogialustana, mutta aikojen saatossa siitä on muotoutunut monipuolinen palvelutarjoaja (WordPress n.d.a). Sitä voi siis nykyään käyttää blogien ja yksinkertaisten nettisivujen lisäksi monimutkaisempienkin portaalien ja yrityssivustojen luontiin, ja jopa sovelluksia voi luoda WordPressilla. Sillä voi luoda mm. henkilökohtaisen blogin tai verkkosivun, valokuvablogin, yrityksen verkkosivut, ammattiportfolion, hallituksen verkkosivun, lehti- tai uutissivuston, verkkoyhteisön tai verkkosivujen verkoston. (WordPress n.d.c.)

Kokematon käyttäjä voi sillä luoda nopeasti ja helposti sivustoja, mutta kokeneempi web-kehittäjä voi halutessaan luoda sillä lähes monipuolisesti juuri haluamansalaiset verkkosivut. Ilmaisuudessaan se soveltuu sekä pieniin, yksityisiin projekteihin, mutta myös isommat kaupalliset tahot voivat käyttää sitä ilman lisenssimaksuja. (WordPress n.d.c.)

4.2 WordPressin ominaisuuksia

Johtuen sen jatkuvasta kehityksestä ja tuhansista liitännäisistä, WordPress antaa käyttäjälle hyvin laajat käyttömahdollisuudet ja lähes rajattoman määrän ominaisuuksia (WordPress n.d.c). Tässä luvussa esittelen tarkemmin WordPressin tärkeimpiä ominaisuuksia, eli sisällönhallinta-, kehittäjä- ja turvallisuusominaisuuksia.

4.2.1 Sisällönhallintaominaisuudet

Sisällönhallintajärjestelmänä WordPress tarjoaa perinteisiä hallintaa helpottavia asioita. Esimerkiksi käyttäjä voi luoda sivuja ja julkaisuja, jotka voi muotoilla helposti ja nopeasti, jonka jälkeen voi valita haluaako sivun tai julkaisun säilyttää luonnoksena, julkaista heti vai julkaista tiettyyn aikaan.

Sivuista ja julkaisuista voi myös tehdä helposti yksityisiä, julkisia tai salasanalla suojattuja. WordPress antaa myös mahdollisuuden luoda sivustolle erilaisia käyttäjiä, jotka voivat muokata ja hallita eri asioita. (WordPress n.d.c.)

Valmiina WordPressissa on myös kommentointimahdollisuus blogijulkaisuihin sekä näiden kommenttien hallinta. Jonkinlainen hakukoneoptimointi on myös valmiina, mutta siihen voi myös itse vaikuttaa erilaisilla liitännäisillä. (WordPress n.d.c.)

WordPress on käännetty usealle kielelle, eikä pelkästään hallintapuolelta, vaan myös monet sisällöt löytyvät valmiiksi useammilla kielillä, muun muassa suomeksi (WordPress n.d.c). WordPressiin voi helposti siirtyä monesta muusta sisällönhallintajärjestelmästä, mm. bloggerista, LiveJournalista, Movable Typestä, TypePadista, Tumblrista ja jopa WordPressista sisäisesti (WordPress n.d.c).

4.2.2 Kehittäjäominaisuudet

Teemat ovat yksi iso kokonaisuus WordPressissä. Teemoilla voidaan vaikuttaa miltä sivusto näyttää. Niitä on valmiina, niitä voi asentaa lisää, niitä voi hallita ja muokata, ja niitä voi tehdä itse lisää. Itse tehdyt teemat voi joko pitää vain omalla sivullaan tai jakaa WordPressin teema-arkistossa. (WordPress n.d.c.)

WordPressin liitännäiset (plugins) antavat monipuolisen mutta helpon tavan saada sivustosta persoonallisempi. Esimerkiksi liitännäisillä voidaan lisätä gallerioita, sosiaalisia verkostoja, foorumeita, sosiaalista mediaa, roskapostisuojaa, kalentereita, hienosäätöä hakukoneoptimointiin ja vaikkapa erilaisia lomakkeita. Näitä liitännäisiä voi myös luoda itse ja lisätä WordPressin liitännäisarkistoon. (WordPress n.d.c.)

WordPressissa tulee mukana monia script-kirjastoja, kuten jQuery, Plupload, Underscore.js ja Backbone.js (WordPress n.d.c). WordPress on luotu niin, että se toimii myös tulevaisuuden nettiselaimilla (WordPress n.d.c).

4.2.3 Turvallisuus

Koskaan ei voi olla täysin turvassa tietomurroilta. WordPressin avulla voi kuitenkin helposti pienentää tietomurron riskiä. Ensimmäinen askel on huolehtia siitä, että WordPress-sivusto pysyy ajan tasalla päivityksissä. WordPress-päivitykset sisältävät usein uusia ominaisuuksia, parantavat käytettävyyttä ja korjaavat havaittuja haavoittuvaisuuksia. (Sabin-Wilson 2013, 117.)

Helpoin ja nopein tapa on käyttää automaattista päivitysominaisuutta. Uuden päivityksen ilmestyttyä WordPressista on helposti löydettävissä päivitysnappula, joka automaattisesti päivittää WordPressin. Päivitykset voivat kuitenkin hajottaa vanhaa sisältöä ja rakennetta, joten on erityisen tärkeää ottaa varmuuskopio ennen tämän ominaisuuden käyttöä. WordPress-päivitykset on mahdollista lisätä myös manuaalisesti. (Sabin-Wilson 2013, 135.)

WordPress tarjoaa mahdollisuuden turvata sivusto ulkopuolisten lisäksi myös sivuston parissa työskenteleviltä käyttäjiltä. WordPressiin voi liittää useita eri käyttäjiä, joilla on erilaisia käyttöoikeuksia sivuston muokkaukseen. Kaikilla ei siis tarvitse olla oikeuksia muokata kaikkea sivustolla. Jollain voi olla oikeus vain muokata ja luoda sisältöä, kun taas toisella voi olla oikeus muokata vain sivuston ulkonäköä (teemaa). (Sabin-Wilson 2013, 120.)

WordPress auttaa tietoturvan ylläpitämisessä monilla muillakin tavoilla. Sen avulla sivuston voi suojata luomalla varmuuskopioita, lisätä sivustolle tietoturvaskannerin, rajoittaa sisäänkirjautumisia, sekä monella muulla tavalla. (Sabin-Wilson 2013, 122.)

5 WORDPRESS-TEEMAT

Sivuteema hallitsee sitä, miltä sivusto näyttää ulkoisesti ja mitä se sisältää. Muuttamalla teemaa voidaan vaikuttaa siihen mitä käyttäjä näkee tullessaan sivustolle, sekä mitä hän sivustolta voi löytää ja miten. (WordPress n.d.d.) WordPress-teemat siis näyttää halutulla tavalla WordPressiin tallennetun tiedon ja sisällön. Hallitsemalla teemaa, voidaan hallita sivuston käyttökokemusta. (WordPress n.d.d.)

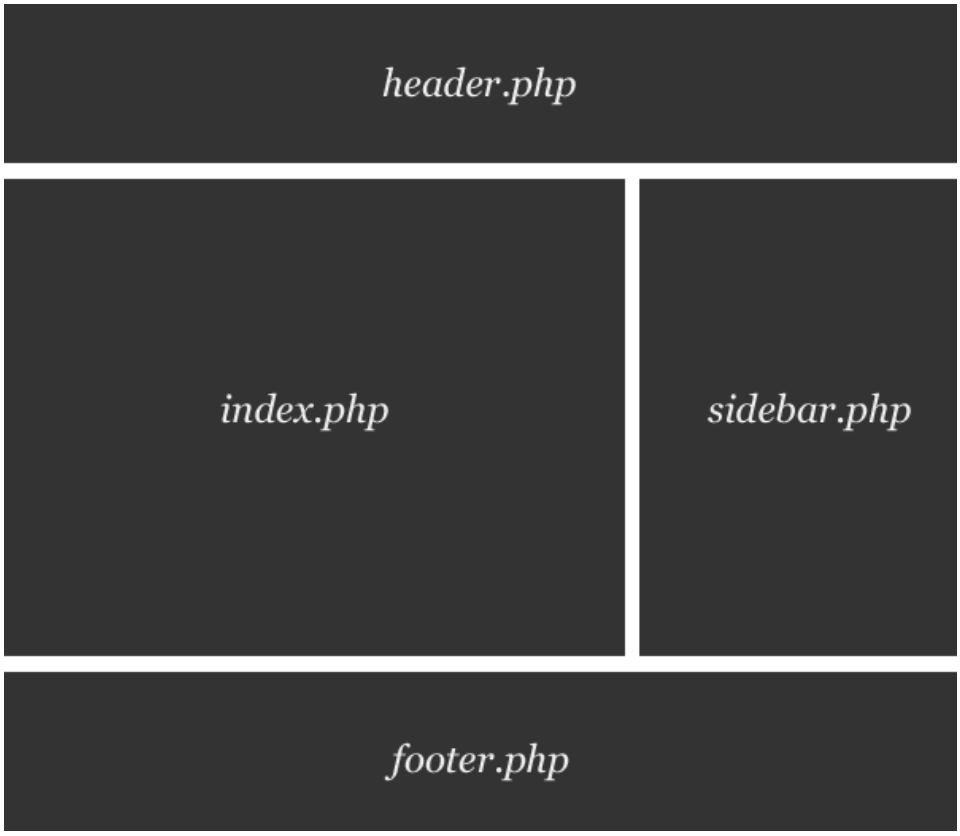
Teemalla voi luoda erilaisia asetteluja: Sivu voi olla staattinen tai responsiivinen, tai siinä voi olla yksi tai useampi palsta. Teema voi tuoda sisällön juuri sinne minne halutaan, jotkin ominaisuudet voidaan saada näkymään vain tietyillä laitteilla tai toiminnoilla. CSS:n avulla teemalla voidaan muokata typografiaa ja muotoilla elementtejä. (WordPress n.d.d.)

WordPress-teemat ovat joukko erilaisia tiedostoja, jotka yhdessä toimimalla luovat sen, miltä sivusto näyttää ja miten se toimii. (WordPress n.d.d.) WordPress-teemat muodostuvat joukosta tiedostoja (template files tai sivupohjatiedostot). Näistä pakollisia ovat index.php (pääsivupohjatiedosto) ja style.css (päätyylitiedosto). (WordPress n.d.d.) Muita mahdollisia tiedostoja ovat muun muassa header.php, footer.php, sidebar.php, frontpage.php, single.php, page.php, search.php ja 404.php. (WordPress n.d.e.)

5.1 Template files

WordPress-teemat siis rakentuvat erilaisista tiedostoista. Niistä index.php on WordPressin päätiedosto. Se tulostaa sivuston. Jos teema sisältää muita omia sivupohjatiedostoja, tulee sen sisältää myös index.html. style.css on päätyylitiedosto. Se on teemoissa pakollinen ja sen tulee sisältää tietoa teemasta. (WordPress n.d.g.)

Tiedosto header.php sisältää sivuston yläosan/alun muotoilut, kun taas footer.php sisältää sivuston alaosan/lopun muotoilut. Näiden lisäksi voidaan luoda esimerkiksi sidebar.php, jolla voidaan lisätä sivustoon vaikkapa vimpaimia. WordPressin vimpaimet ovat erillisiä lisäosia, joilla sivuun voidaan liittää lisäominaisuuksia, esimerkiksi hakupalkki. (WordPress n.d.g.)



Kuva 1. *Yksinkertainen esitys siitä, mitä sivupohjatiedostot saattaisivat muodostaa sivulle asetettuna (Tadlock 2011).*

Tiedosto front-page.php sisältää etusivun, single.php yksittäisten artikkeleiden ja page.php yksittäisten sivujen muotoilut. Näiden lisäksi WordPress antaa mahdollisuuden muokata monia muitakin sivutyyppejä. Jos mitään muuta muotoilua ei ole, käytetään index.php:tä. (WordPress n.d.g.)

5.2 **Template tags**

Template tags, eli karkeasti suomeksi käännettynä sivupohjatiedostotunnistimet, ovat yksinkertaisia PHP-koodeja, joilla voidaan luoda mahdollisimman dynaamiset sivupohjatiedostot. (Sabin-Wilson 2013, 442.)

Tyypillinen template tag näyttää seuraavanlaiselta:

```
<?php get_footer(); ?>
```

Tällä pätkällä aloitetaan PHP (`<?php`), käytetään PHP:tä tuomaan tietoa (alatunniste) MySQL-tietokannasta (`get_footer();`) ja lopetetaan PHP (`?>`). Mikä tieto tuodaan riippuu siis siitä, mitä `get_footer();` -kohtaan laitetaan. (Sabin-Wilson 2013, 443.)

Nämä tunnistimet helpottavat WordPress-sivujen luontia. Esimerkiksi header.php, footer.php ja sidebar.php on hyvä tehdä ja viitata niihin

muissa sivupohjatiedostoissa (mm. index.php, front-page.php) template tageilla. Tällöin jos haluaa muokata alatunnistetta, voi sen toteuttaa muokkaamalla vain footer.php -tiedostoa. Jos template tageja ei käyttäisi, joutuisi jokaisen sivupohjatiedoston ylä- ja alatunnisteet muokkaamaan yksitellen. (Sabin-Wilson 2013, 443.)

5.3 **Silmukka**

WordPressin Silmukka (The Loop) on yksi WordPressin tärkeimmistä ominaisuuksista, varsinkin blogisivustoa luotaessa. Sen avulla saadaan näkymään luodut artikkelit (posts) ja sivut (pages) omalla sivulla. Yksinkertaisena tämä silmukka voisi esiintyä seuraavanlaisesti:

```
<?php
if (have_posts()) :
while (have_posts()) :
the_post();
the_content();
endwhile;
endif;
?>
```

`<?php` ja `?>` sisään tulee tarvittava PHP-koodi. `if (have_posts()) :` ja `endif;` välissä on ne asiat, joiden tulee tapahtua jos julkisia artikkeleita on luoto. `while (have_posts()) :` ja `endwhile;` välissä ovat ne asiat joiden tulee toistua, eli ne muodostavat silmukan. While `(have_posts())` viittaa siihen, että silmukka tapahtuu niin kauan, kuin uusia artikkeleita löytyy. Tässä esimerkissä silmukka toistaa artikkeleiden indeksit ja sisällön. (Sabin-Wilson 2013, 450.)

5.4 **Koukut**

Koukut (hooks) mahdollistaa sen, että kehittäjien on helppo yhdistää omaa koodia WordPressin. WordPressissa koukku on yleisnimitys paikoista, joihin voi lisätä omaa koodia tai muuttaa WordPressin oletustoimintoja. Nämä koukut voidaan jakaa toimintoihin (action) ja suodattimiin (filter). (Gordon 2015.)

Toimintokoukku reagoi tiettynä hetkenä ja antaa mahdollisuuden toimia, esimerkiksi julkaista sosiaalisessa mediassa kun joku julkaisee artikkelin. Suodatinkoukku mahdollistaa WordPress-datan saamisen ja muokkauksen ennen kuin se lähetetään tietokantaan tai selaimelle. Esimerkki tästä voisi olla oman koodin lisääminen artikkelin loppuun. (Gordon 2015.)

5.5 **Vimpaimet**

Widget eli vimpain on WordPressin työkalu. Se on luotu käytettäväksi erityisesti sivupalkkien sisällön hallintaan. Sivupalkista voi luoda vimpain-alueen, jolloin sivupalkkiin voi lisätä erilaisia ominaisuuksia ilman, että tarvitsee tietää mitään eri merkintäkielistä tai koodaamisesta. (Sabin-Wilson 2013, 415.)

Vimpaimia voi lisätä vimpain-alueille WordPressissä selaimen kautta. Ne ovat kuin pienempiä rakennuspalikoita, joilla omaan sivustoon voi lisätä erilaisia ominaisuuksia. Vimpaimilla voidaan esimerkiksi liittää sivupalkkiin hakupalkki, linkkilista tai ylimääräinen valikko. (Sabin-Wilson 2013, 415.)

6 RESPONSIIVISUUS

Responsiivisella web-suunnittelulla verkkosivut saadaan näyttämään hyviltä joka näyttökoossa. Käyttäen CSS:n ja HTML:n eri yhdistelmiä, voidaan sisällön kokoa tai sijaintia muuttaa tai jotain sisältöä voidaan piilottaa, jotta sivu näyttäisi mahdollisimman hyvältä eri näyttökoissa. (W3schools n.d.)

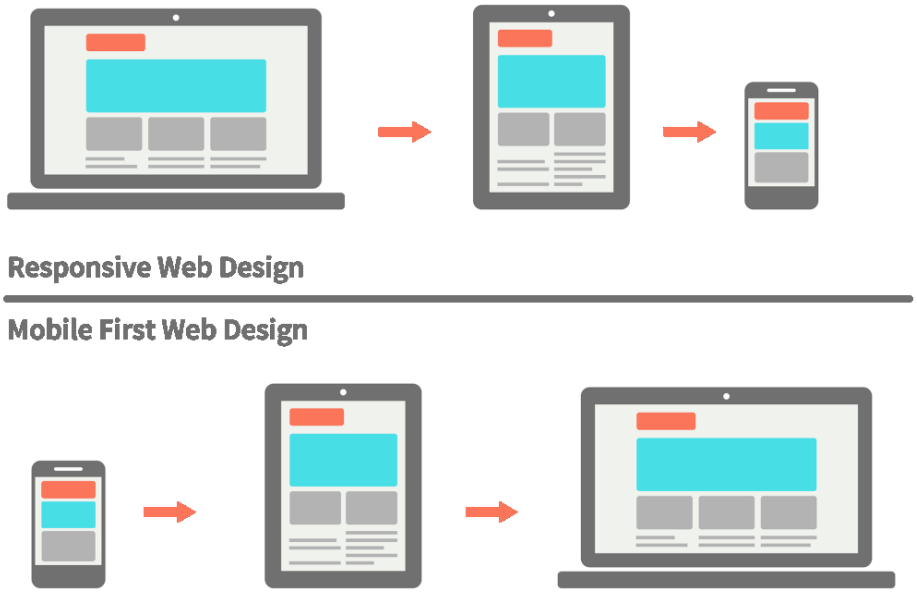
Kuva 2. *Esimerkki siitä, miten sisältö voitaisiin näyttää eri tavoilla erikokoisilla näytöillä responsiivisuutta käyttämällä (Caldwell 2013).*

Puhelimien ja tablettien käyttö on yleistynyt räjähdysmäisesti, mutta niiden näyttökoot usein hankaloittavat tietokonenäytöille suunniteltujen nettisivujen käyttöä. Nykyään erilaisia näyttökokoja onkin jo niin paljon, että on hyvä toteuttaa verkkosivut niin, että ne voivat toimia millä tahansa näyttökoolla. (LePage 2017.)

Tällainen responsiivinen sivusto voidaan toteuttaa vaikkapa niin, että puhelinkäyttäjät näkevät kaiken sisällön yhdellä sarakkeella, mutta tablettien käyttäjät näkevät saman sisällön kahdella sarakkeella. (LePage 2017.)

6.1 Mobile First

Aikaisemmin useimmat sivustot on tehty lähtökohtaisesti tietokoneille, ja vasta sen jälkeen toteutettu sivusto (jos on toteutettu) puhelimille sopivaksi. Vuonna 2009 Luke Wroblewski kuitenkin aivan toisenlaisen lähestymistavan. Uusi mobile-first-konsepti kehotti toteuttamaan koko homma aivan toisin päin, eli sivusto toteutetaankin ensisijaisesti puhelimille ja vasta toissijaisesti tietokoneille. (Code My Views n.d.)



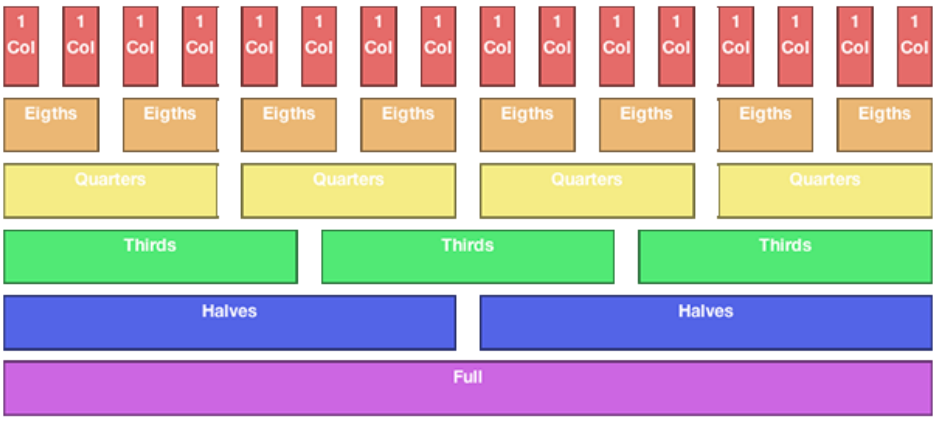
Kuva 3. *Kuinka mobile-first-suunnittelu (Mobile First Web Design) eroaa perinteisestä responsiivisesta suunnittelusta (Responsive Web Design) (Ganzola 2017).*

Vuonna 2016 puhelimet tuottivat 43,6% kaikesta verkkosivuliikenteestä ja arvioiden mukaan vuoden 2017 prosentti nousisi yli 50 prosenttiin. Luku on vuosi vuodelta noussut aina suuremmaksi, joten nyt onkin hyvä aika aloittaa mobile-first-suunnittelu. (Statista 2017.)

Mobile-first-suunnittelussa ajatellaan ensisijaisesti, mikä toimisi mobiilikäyttäjillä, mikä rajoitteita mobiilikäyttäjillä on ja mikä on tunnusomaista mobiilikäyttäjille. Mobile-first-strategian kanssa pitääkin tarkkaan harkita, luokitella ja järjestellä sivuston sisältö, ja pitää huolta että sivusto pysyy selkeänä. Ylimääräistä sisältöä voi kuitenkin luoda isommille näytöille, jos se parantaa käyttökokemusta. (Ganzola 2017.)

6.2 Grid system

Responsiivisissa sivuissa grid tarkoittaa eräänlaista pohjaruudukkoa, johon sivusto luodaan. Tämä pohjaruudukko muodostuu riveistä ja palstoista, joiden avulla sivu voidaan jakaa osiin ja eri osat liikutella halutuille paikoille erikokoisissa näytöissä. (Zielinski 2014.)



Kuva 4. *16-palstainen pohjaruudukko (grid), jossa havainnollistettu erilevyisiä sisältöjä (Zielinski 2014).*

Kuvan 4 pohjaruudukossa jokainen väri kuvaa yhtä riviä. Punaiset (eli ylimmät) laatikot kuvaavat yksittäisiä palstoja, kun taas violetti (eli alin) laatikko kuvaa koko leveyttä. Laatikot niiden välissä kuvaavat eri tavoin kokonaisuuksiksi jaettuja palstoja.

Responsiivisen web-suunnittelun frameworkit ovat kehitysympäristöjä, joissa tämä pohjaruudukko on luotu valmiiksi. Näillä valmiilla frameworkeilla responsiivisuuden luo helposti ja nopeasti. Hyvät frameworkit tarjoavat myös valmiita malliesimerkkejä siitä, kuinka päästä alkuun. Tällainen framework on esimerkiksi Bootstrap. (Zielinski 2014.)

6.3 Bootstrap framework

Vuonna 2010 Twitter-yrityksessä luotu Bootstrap on nykyään tituleerattu suosituimmaksi front-end-kehitysympäristöksi ja avoimen lähdekoodin projektiksi (Bootstrap n.d.a.). Bootstrapin valmiilla HTML-, CSS- ja JS-puitteilla voidaan luoda responsiiviset verkkosivustot nopeammin ja helpommin. Boostrap-frameworkia käyttämällä sivustoista voidaan siis tehdä kerralla useammalle laitteelle sopivat. (Bootstrap n.d.b.)

Bootstrap-frameworkin voi helposti ladata Bootstrapin omilta verkkosivuilta ja lisätä omaan verkkosivuprojektiin. Bootstrap on MIT-lisenssin alainen, mikä tarkoittaa, että sen voi ottaa vapaasti myös yrityskäyttöön (Bootstrap n.d.c.).

Bootstrap valittiin tähän projektiin, koska se on tunnetuin työkalu kyseiseen käyttöön, minkä johdosta siitä on eniten käyttöohjeita ja -vinkkejä. Se on myös erittäin laaja ja monipuolinen, ja minulla oli aikaisempia kokemuksia sen käytöstä, joten tiesin sen varmasti sisältävän kaiken mitä tähän projektiin kaipasin.

6.4 **Muita frameworkeja responsiivisuuden luontiin**

Bootstrapin lisäksi responsiivisuuden luontiin on useita erilaisia frameworkeja, tässä on niistä muutama esimerkki. Eri vaihtoehtoja on kuitenkin valtavasti.

6.4.1 Foundation

Foundation on toinen erittäin suosittu front-end framework. Sen lähtökohdat ovat vuonna 2008, jolloin luotiin ZURB Style Guide asiakastöihin. (foundation) Foundation käyttää HTML-, CSS- ja JavaScript-yhdistelmää luodakseen alustan responsiivisille sivuille, ja se on rakennettu Sassille. (Zurb Foundation n.d.)

Foundation on kehitetty responsiivisten verkkosivujen luonnin lisäksi myös responsiivisten sähköpostien luontiin. Se mainostaa tarjoavansa selkeät merkintätavat, mobile-first-luonnin, sekä mahdollisuuden muokata eri osa-alueita. (Zurb Foundation n.d.)

6.4.2 Skeleton

Skeleton on erittäin yksinkertainen framework. Se on vain noin 400 riviä pitkä. Se onkin erityisesti niille, jotka eivät koe tarvitsevansa isoja ja monimuotoisia frameworkeja. Yksinkertaisuudessaan se soveltuukin parhaiten yksinkertaisten sivujen tekoon.

Niin kuin Bootstrap, se sisältää 12 palstaisen pohjaruudukon, joka kapenee ja kasaantuu näytön koon mukaan. Tämän lisäksi se voi helpottaa joidenkin perinteisten elementtien luontia, kuten otsikoiden, nappuloiden ja lomakkeiden. (Skeleton n.d.)

6.4.3 HTML Kickstart

Kickstart luotiin aluksi Bootstrapin jatkoksi. Aikojen saatossa Bootstrap on kuitenkin poistettu siitä kokonaan, ja Kickstart on nykyään oma kokonaisuutensa. Sen kehittäjä koki muut frameworkit isoiksi, hitaiksi, vaikeiksi laajentaa ja yksipuolisiksi. (Kickstart n.d.)

Kickstartia on kehitetty muiden frameworkien haastajaksi. Se mainostaa olevansa reilusti pienempi kuin Bootstrap ja lähes puolet pienempi kuin Foundation CSS:n osalta. JavaScriptiä Kickstartissa on tuskin lainkaan. Se ei myöskään väitä itseään frameworkiksi vaan ohjelmakirjastoksi, jonka pitäisi taata nopeampi sivusto. (Kickstart n.d.)

6.4.4 Itse tehty toteutus

Joskus sivustoon kaivataan vain pohjaruudutus (grid), jolloin valmiit frameworkit voivat olla turhia ja vain hidastaa sivua. Tällöin on mahdollista luoda ihan oma pohjaruudutus.

Itse tehdyn pohjaruudutuksen hyviä puolia ovat erityisesti keveys ja nopeus. Se vie mahdollisimman vähän tilaa ja keveyden johdosta sivu voi toimia nopeammin. Kasaamalla omat CSS ja HTML yhdistelmät, saa lopputuloksesta varmasti juuri sellaisen kuin haluaakin. Tämän lisäksi kaikki on helposti omassa hallinnassa: voi valita palstojen määrän, koon sekä sen, miten näytön koko vaikuttaa palstoihin. (Zell 2016.)

7 TOTEUTUS

7.1 Tavoitteet ja tarkoitus

Tämän toteutuksen tavoitteena oli uudistaa asiakasyrityksen verkko-sivut luomalla uusi teema. Asiakasyrityksen toiveena oli yksinkertaiset ja helppokäyttöiset verkkosivut, jotka kuitenkin soveltuvat monelle erilaiselle laitteelle. Ulkonäön suhteen tavoitteena oli luoda konepajalle sopivat, ammattimaisen näköiset sivut, jotka ovat helppokäyttöiset myös kokemattomammille netin käyttäjille.

Johtuen asiakkaan kokemattomuudesta verkkosivujen teossa, projektiin kuului myös opastaminen verkkosivujen ylläpidossa, sisällön tuottamisessa, sekä monissa muissa ongelmissa. Kuitenkin tähän raporttiin on sisällytetty projektista vain se osa, joka on liittynyt WordPress-teeman luontiin. WordPress-teeman suunnittelu on kuvailtu lyhyesti, jotta toteutuksen rakenne olisi helpompi ymmärtää.

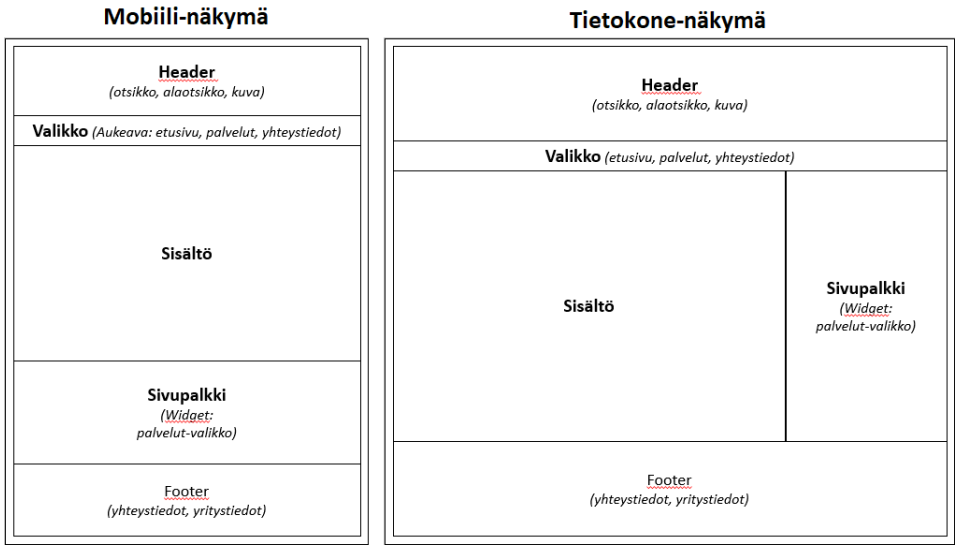
7.2 Suunnittelu

Projekti aloitettiin suunnittelemalla sivun sisältö. Sinne toivottiin omat sivut eri palveluille, joita yritys tarjoaa. Pinnoittamona yritys siis toivoi, että sivu korostaisi niitä pinnoituspalveluja, joita he tarjoavat. Etusivun toivottiin olevan selkeä. Sen toivottiin mahdollistavan sen, että jokainen asiakas löytää heti, mitä eri pinnoituspalveluja yritys tuottaa.

Päädyttiinkin siihen, että perinteisen valikon lisäksi etusivulta löytyisi myös selkeä lista eri pinnoituspalveluista. Tämän lisäksi ei yrityksellä ollut paljoa toiveita. Tärkeänä pidettiin, että sivulta löytyisi yrityksen nimi, motto, yhteystiedot, palvelut ja palvelujen kuvaukset.

Joitain muitakin ideoita yrityksen suunnalta kuitenkin tuli. Esimerkiksi suunniteltiin jonkinlaista blogia, johon yritys voisi kirjoittaa omia kuulumisiaan tai vastata kysymyksiin. Ajatus kuitenkin hylättiin, koska koettiin, ettei tällaisen ylläpitoon olisi tarpeeksi resursseja pidemmän päälle.

Tästä suunnitelmat jatkuivat rautalankamallilla (kuva 5). Luotiin kaksi rautalankamallia: mobiili-näkymälle ja tietokonenäkymälle. Näin oli valmiiksi suunnitelma, miten sivun eri osien haluttiin asettuvan kapeilla ja leveillä näytöillä. Tämä suunnitelma hyväksytettiin asiakkaalla, minkä jälkeen päästiin toteutukseen.



Kuva 5. *Sivuston rautalankamalli mobiili- ja tietokone-näkymästä.*

7.3 Aloitus

Teeman luonti aloitettiin luomalla testiympäristö omalle tietokoneelle. Tähän käytettiin WampServeriä eli WAMPia. WAMP on ilmainen Windows-koneille tarkoitettu web-kehitysympäristö. Tässä sitä käytettiin, jotta saatiin luotua paikallinen palvelin omalle Windows-koneelle. Tämän lisäksi ladattiin WordPressin viimeisin versio. Se onnistui WordPressin omilta sivuilta.

Kun sekä WAMP että WordPress oli ladattu, asennettiin WAMP. WAMP:in asennuksen jälkeen luotiin WAMPilla MySQL-tietokanta WordPressia varten. Tämän jälkeen pääsi asentamaan WordPressia.

WordPressin asennus aloitettiin purkamalla ladattu WordPress-tiedosto WAMPin www-kansioon. Tämä löytyi siitä kansiosta, jonne WAMP oli asennettu. WordPress-kansion nimeksi muutettiin ”Asiakasyritys”, joten itse asennukseen päästiin menemällä selaimella osoitteeseen http://localhost/Asiakasyritys/. Sivustolla näkyviä ohjeita seuraamalla asennus onnistui helposti.

Näiden lisäksi tuli ladata Bootstrap-framework tiedostona. Näin alkulataukset olivat valmiina.

7.4 Teeman luominen

Jotta teeman luominen päästiin aloittamaan, tuli luoda teemalle oma kansio kohteeseen C:\wamp64\www\Asiakasyritys\wp-content\themes. Tälle uudelle teema-kansiolle annettiin nimeksi ”Asiakasyritys_teema”.

Asiakasyritys-teema kansioon luotiin tiedostot, footer.php, header.php, index.php, sidebar.php ja style.css. Tähän kansioon myös purettiin valmiiksi

ladattu Bootstrap-tiedosto. Koska edellä mainitut php-tiedostot jakavat sivuston osiin, jaettiin myös perinteinen sivuston HTML-malli osiin. Taulukossa 1 on kuvattu jokainen Bootstrapin lisäksi luotu tiedosto sekä niiden ominaisuudet ja sisältö.

Taulukko 1. *Asiakasyritys-teemaa varten luodut tiedostot sekä niiden ominaisuudet ja sisältö.*

Tiedosto	Ominaisuus/Sisältö
style.css	Teeman tiedot, sekä kaikki css-muotoilu joka haluttiin valmiiden Bootstrap-muotoilujen lisäksi tai tilalle
header.php	Asiat, jotka kuuluvat HTML-sivun aloitukseen, sekä otsikon, aliotsikon, bannerin ja ylävalikon luontiin.
index.php	Vaaditaan WordPress-sivuston luontiin. N Asiat, jotka kuuluvat yksittäisten sivujen sisällön näyttöön. Siinä kutsutaan header.php, sidebar.php, sekä footer.php, eli tehdään sivuston HTML:stä kokonainen.
sidebar.php	Asiat, jotka kuuluvat sivupalkin luontiin. Tässä tapauksessa sidebar.php luotiin vimpain-alueeksi.
footer.php	Asiat, jotka kuuluvat HTML-sivun lopetukseen, sekä alapalkin luontiin.
functions.php	Asiat, jotka kuuluvat erilaisiin toimintoihin. Esimerkiksi tässä tiedostossa on luotu sivupalkin vimpain-alue.

8 TULOKSET

Wordpress-teemojen luonti tuntui yksinkertaiselta ja suoraviivaiselta. Ihan täysin ongelmitta ei kuitenkaan selvitty. Yleensä ongelmia esiintyi sen takia, että Wordpressista tai Bootstrapista oli tullut uusi versio, joihin vanhat ohjeet ja vinkit eivät enää toimineetkaan. Näistä ongelmatilanteista kuitenkin päästiin usein kokeilemalla jotain toista tapaa tai yhdistelemällä vanhoja ja uusia ohjeita.

Isoin syy oman teeman luontiin on se, että luomalla oman teeman on täysi hallinta teeman sisällöstä ja ominaisuuksista, jolloin voi luoda täysin omanlaisen teeman (WordPress n.d.g). Kun teema sisältää vain halutut ominaisuudet, se pysyy myös mahdollisimman kevyenä. Yhtenä syynä itsellä oli myös oppiminen: kun kerran on itse luonut alusta asti oman teeman, luulisi olevan helpompi myös muokata muiden tekemiä teemoja, kun teemojen sisällön luonti on tuttua. WordPress itse luetteleekin oman teeman luonnin syiksi muun muassa mahdollisuuden oppia lisää CSS:stä, HTML:stä ja PHP:stä, sekä niiden taitojen hyötykäytön, teemanluonnin luovuuden, sekä mahdollisuuden päästä jakamaan teema muiden käyttöön. (WordPress n.d.g).

Toinen vaihtoehto olisi käyttää valmista teemaa ja muokata siitä halutunlainen. Kun oman teeman luominen on aina hyvin suuritöistä ja päivitys pitää hoitaa itse, voi valmista teemaa käyttämällä päästä pienellä työllä jos sopiva pohjateema löytyy. Kuitenkin, jos sivulle halutaan jokin tietty asia, voi se olla hankalaa jos käytössä on valmis teema. Joskus voi jopa käydä niin, että valmista teemaa muokataan niin paljon ja niin kauan, että helpompaa olisi ollut luoda ihan oma teema alusta alkaen. (McCollin & Silver 2013, 32.) Itse myös huomasin, ettei kaikkien teemojen muokkaaminen ole edes täysin mahdollista silloin, kun tekijät ovat tallentaneet osan teeman luovista tiedostoista omille palvelimilleen. Joskus myös käyttöoikeudet saattavat hankaloittaa muiden tekemien teemojen käyttöä, varsinkin yritysssivuja luotaessa.

9 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli kuvailla responsiivisen WordPress-teeman luominen asiakasyritykselle sekä selvittää, onko oman teeman luominen järkevää. Responsiivinen teema saatiin aikaan käyttämällä Bootstrap-frameworkia. Tätä teemaa on tarkoitus käyttää asiakasyrityksen verkkosivuilla.

Asiakas oli toivonut mahdollisimman yksinkertaista ja tavallista sivustoa, joten kovin erikoisia ominaisuuksia ei tämän projektin aikana päässyt toteuttamaan. Oli kuitenkin mielenkiintoista päästä yhdistämään Bootstrap-framework Wordpress-teemaan, kun en sitä aikaisemmin ollut tehnyt.

WordPress-teeman luominen tuntui yksinkertaiselta ja suoraviivaiselta, mutta ongelmiakin ilmeni. Erityisesti ongelmia aiheuttivat uudet ohjelmistoversiot ja niiden yhteensopivuusongelmat.

Voitiin todeta, että oman WordPress-teeman luonti on joissain tapauksissa kannattavaa, mutta sen hyviä ja huonoja puolia kannattaa miettiä ennen teemanluontitavan päättämistä. Erityisesti oman teeman hyviä puolia ovat sen keveys ja mahdollisuus valita juuri halutut ominaisuudet. Sen sijaan valmista teemaa kannattaa käyttää pohjana jos lähes samanlainen teema löytyy jo valmiina ja aktiivisesti päivitettynä. Tällöin valmista teemaa muokkaamalla voidaan päästä paljon vähemmällä työllä kuin luomalla oma teema.

Tämä opinnäytetyö esittelee responsiivisen WordPress-teeman luonnin perusteita sekä sen, miten olen itse luonut teeman. Toivonkin sen hyödyttävän lukijoita tulevaisuuden responsiivisten WordPress-teemojen luonnissa.

LÄHTEET

Bootstrap (n.d.a). About - Bootstrap. Haettu 17.5.2017 osoitteesta http://getbootstrap.com/about/

Bootstrap (n.d.b). Bootstrap - The world's most popular mobile-first and responsive front-end framework. Haettu 17.5.2017 osoitteesta http://getbootstrap.com/

Bootstrap (n.d.c). Getting started - Bootstrap. Haettu 17.5.2017 osoitteesta http://getbootstrap.com/getting-started/

Caldwell, A. (2013). Responsive Web Design Examples with CSS Tips and Tricks. Haettu 30.8.2017 osoitteesta http://brolik.com/blog/responsive-web-design-examples-with-css-tips-and-tricks/

Code My Views (n.d.). Mobile First Design: Why It's Great and Why It Sucks. Haettu 30.8.2017 osoitteesta https://codemyviews.com/blog/mobilefirst

Drupal (n.d.) Mikä on Drupal? Haettu 17.8.2017 osoitteesta http://www.drupal.fi/

Ganzola, F. (2017). Understanding the difference between mobile-first, adaptive and responsive design. Haettu 17.8.2017 osoitteesta http://fredericgonzalo.com/en/2017/03/01/understanding-the-difference-between-mobile-first-adaptive-and-responsive-design/

Gordon, Z. (2015). WordPress Hooks: Actions, Filters, and Examples. Haet-tu 30.8.2017 osoitteesta http://blog.teamtreehouse.com/hooks-wordpress-actions-filters-examples

Jaakohuhta, H. (2011). Tietotekniikan sanakirja. Helsinki: Readme.fi Oy.

Joomla (n.d.). About Joomla! Haettu 17.8.2017 osoitteesta https://www.joomla.org/about-joomla.html

Keränen, V., Lamberg, N. & Penttinen, J. (2006). Web-julkaiseminen & multimedia. 1. painos. Jyväskylä: Docendo Finland Oy.

Kickstart (n.d.). Kickstart - About. Haettu 30.8.2017 osoitteesta http://getkickstart.com/about/

Larsen, R. (2013). Beginning HTML and CSS. 1. painos. Hoboken: John Wiley & Sons, Inc.

LePage, P. (2017). Responsive Web Design Basics. Haettu 17.8.2017 osoitteesta https://developers.google.com/web/fundamentals/design-and-ui/responsive/

Liew, Z. (2016). How to build a responsive grid system. Haettu 30.8.2017 osoitteesta https://zellwk.com/blog/responsive-grid-system/

McCollin, R. & Silver, T. (2013). WordPress Theme Development Beginner's Guide. 3. painos. Birmingham: Packt Publishing Ltd.

Noble, J. & Tittel, E. (2008). HTML, XHTML and CSS For Dummies. 6. painos. Hoboken: John Wiley & Sons, Inc.

Omni Partners (n.d.). CMS eli Content Management System. Haettu 17.8.2017 osoitteesta https://omnipartners.fi/sanakirja/cms-eli-content-management-system/

Sabin-Wilson, L. (2013). WordPress All-in-One For Dummies. 2. painos.Hoboken: John Wiley & Sons, Inc.

Shannon, R. (2012). What is HTML? Haettu 17.8.2017 osoitteesta http://www.yourhtmlsource.com/starthere/whatishtml.html

Skeleton (n.d.). Skeleton: Responsive CSS Boilerplate. Haettu 30.8.2017 osoitteesta http://getskeleton.com/

Statista (2017). Percentage of all global web pages served to mobile phones from 2009 to 2017. Haettu 17.8.2017 osoitteesta https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/

Suehring, S. & Valade, J. (2013). PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies. Hoboken: John Wiley & Sons, Inc.

Tadlock, J. (2011). Creating basic theme templates. Haettu 30.8.2017 osoitteesta http://justintadlock.com/archives/2011/09/28/creating-basic-theme-templates

Tieteen termipankki (2014). Tekstuaalitieteet:merkintäkieli. Haettu 17.8.2017 osoitteesta http://tieteentermipankki.fi/wiki/Tekstuaalitieteet:merkint%C3%A4kieli

W3C (2017). HTML. Haettu 17.8.2017 osoitteesta https://www.w3.org/html/

W3schools (n.d.). HTML Responsive Web Design. Haettu 17.8.2017 osoitteesta https://www.w3schools.com/html/html_responsive.asp

W3Techs (2017). Usage of content management systems for websites. Haettu 17.8.2017 osoitteesta https://w3techs.com/technologies/overview/content_management/all/

Web-opas (2012). Mikä on CMS? Haettu 17.8.2017 osoitteesta http://www.webopas.net/cms.html

WordPress (n.d.a). About – WordPress.com. Haettu 17.5.2017 osoitteesta https://wordpress.org/about/

WordPress (n.d.b). Blog Tool, Publishing Platform, and CMS – WordPress. Haettu 17.5.2017 osoitteesta https://wordpress.org/

WordPress (n.d.c). WordPress. Haettu 17.5.2017 osoitteesta https://wordpress.org/about/features/

WordPress (n.d.d). What is a Theme? Haettu 17.5.2017 osoitteesta https://developer.wordpress.org/themes/getting-started/what-is-a-theme/

WordPress (n.d.e). Template Files. Haettu 17.5.2017 osoitteesta https://developer.wordpress.org/themes/basics/template-files/

WordPress (n.d.f). Template Hierarchy. Haettu 17.5.2017 osoitteesta https://developer.wordpress.org/themes/basics/template-hierarchy/

WordPress (n.d.g). Theme Development. Haettu 17.5.2017 osoitteesta https://codex.wordpress.org/Theme_Development

Wpbeginner (n.d.). What is: Content Management System (CMS). Haettu 17.8.2017 osoitteesta http://www.wpbeginner.com/glossary/content-management-system-cms/

Zielinski, W. (2014). The Responsive Web Series — Part 2: Responsive Grids. Haettu 30.8.2017 osoitteesta http://www.radius180.com/the-responsive-web-series-part-2-responsive-grids/

Zurb Foundation (n.d.). About Foundation. Haettu 17.8.2017 osoitteesta http://foundation.zurb.com/showcase/about.html