VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Timi-Artturi Mäkelä

Responsiiviset WordPress-verkkosivut

Liiketalous
2016

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Timi-Artturi Mäkelä
Opinnäytetyön nimi	Responsiiviset WordPress-verkkosivut
Vuosi	2016
Kieli	suomi
Sivumäärä	53
Ohjaaja	Raija Tuomaala

Tämän opinnäytetyön aiheena on WordPress-teemakehitys ja responsiivinensuunnittelu. Teoriaosuudessa käydään läpi WordPressin tiedostorakennetta teemakehityksen sekä yleisen WordPress-kehityksen kannalta. Lisäksi käydään läpi tapoja toteuttaa responsiivisia sivustoja ohjelmointikehystä avuksi käyttäen.

Käytännönosuudessa käydään läpi tapoja, kuinka kehittää sivustoja WordPress-alustalla verkkosivukehittäjän näkökulmasta. Yhtenä isona kokonaisuutena on kustomoidun sisällön luominen Custom Post Type -sisältötyypin ja Advanced Custom Fields -lisäosan avulla. Lisäksi käsitellään, kuinka voidaan luoda käyttäjälle mieluisa, jäsennelty ja intuitiivinen käyttöliittymä.

Opinnäytetyön tuotteena syntyi uusi responsiivinen WordPress-toteutus Suomen Potilasturvallisuusyhdistykselle. Toimeksiantajana toimi Awanic Oy. Toimeksiantaja ja Suomen Potilasturvallisuusyhdistys olivat tyytyväisiä lopputulokseen.

Avainsanat WordPress, teemakehitys, responsiivisuus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Timi-Artturi Mäkelä
Title	A Responsive WordPress Website
Year	2016
Language	Finnish
Pages	53
Name of Supervisor	Raija Tuomaala

This thesis researches theme development and responsive web design for WordPress content management system. The theoretical study of this thesis research the structure of WordPress from the theme and general development point of view for WordPress. The theoretical study researched also the techniques for implementing responsive web design with CSS framework.

The practical study of the thesis shows ways to develop themes with WordPress. One of the main goals was to create an intuitive and easy to use back end for the users. The thesis researched also ways to create custom content with custom post types and advanced custom fields plugins.

As a result of this thesis was a new responsive website and design for Suomen Potilasturvallisuusyhdistys. The project was commissioned by Awanic Oy. Both Suomen Potilasturvallisuusyhdistys and Awanic Oy were pleased with the end product.

Keywords WordPress, theme development, responsiveness

SISÄLLYS

TIIVISTELMÄ
ABSTRACT
1 JOHDANTO 11
2 WORDPRESS 13
3 BOOTSTRAP 15
3.1 Grid-järjestelmä 15
3.2 Ikonit 17
3.3 Asentaminen 18
4 SASS 20
4.1 Muuttujat 20
4.2 Sass-tiedoston kääntäminen CSS-tiedostoksi 21
5 TEEMAKEHITYS 23
5.1 Teeman tiedostot 23
5.1.1 Style.css 23
5.1.2 Functions.php 24
5.1.3 Tyylitiedostojen jonottaminen 25
5.2 Sivupohja-tiedostot ja hierarkinen malli 26
5.3 Sisältötyypit 27
5.3.1 Post 27
5.3.2 Page 28
5.3.3 Attachment 28
5.3.4 Nav menus 28
5.3.5 Custom Post Type 28
6 SISÄLLÖN SYÖTTÄMINEN WORDPRESSIIN 31
6.1 Sisällön syöttäminen sivuille ja artikkeliin 31
6.2 Media-kirjasto 32
7 SISÄLLÖN NÄYTTÄMINEN TEEMASSA 34
7.1 The loop 34
7.2 Sivupohja-funktiot 35

7.3 Advanced Custom Fields 35
7.3.1 Advanced Custom Fields kentän määrittely 36
7.3.2 Advanced Custom Fields sisällön näyttäminen sivupohjassa 37
7.3.3 Advanced Custom Fields repeater field 38
7.4 Advanced Custom Fields WordPressin tietokannassa 39
8 CUSTOM POST TYPE MÄÄRITTÄMINEN 40
8.1 Custom post type sisällön näyttäminen 41
8.2 Archive.php 42
8.3 Single.php 43
9 WORDPRESS JA TIETOTURVA 45
9.1 Päivitykset 45
9.2 Table prefix 45
9.3 Wp-config-tiedoston siirtäminen 47
9.4 Käyttäjäoikeudet 48
10 TULEVAISUUDEN NÄKYMÄT TEEMAKEHITYKSESSÄ 50
LÄHTEET 52

KUVIO- JA TAULUKKOLUETTELO

Kuvio 1.	Esimerkki Bootstrap hierarkiasta ja padding-ominaisuuden lisäämisestä kolumnin sisään	s.16
Kuvio 2.	Esimerkki pull-right luokan käytöstä	s.17
Kuvio 3.	Glyphicon bootstrap.css tiedostossa	s.18
Kuvio 4.	Taustavärin määritys lapsielementtiin sass-kielellä.	s.20
Kuvio 5.	Sass-muuttujan määrittely ja kutsuminen.	s.21
Kuvio 6.	Koala Sass-kääntäjän käyttäminen.	s.22
Kuvio 7.	Menun rekisteröinti functions.php-tiedostossa.	s.25
Kuvio 8.	Sivupohjan nimeäminen sivupohjatiedostossa.	s.26
Kuvio 9.	Kuvankaappaus sivun luomisnäkymästä.	s.27
Kuvio 10.	Custom post type admin-näkymässä.	s.29
Kuvio 11.	Oletus sisällönsyöttönäkymä Wysiwyg-kentällä.	s.31
Kuvio 12.	Media-kirjasto tiedoston maksimikoko.	s.33
Kuvio 13.	The loop määritteleminen sivupohjassa.	s.34
Kuvio 14.	Advanced Custom Fields admin-valikossa.	s.36
Kuvio 15.	Advanced Custom Fields kentän liittäminen sivuun.	s.37
Kuvio 16.	Advanced Custom Fields-kenttä sivun admin-näkymässä.	s.37
Kuvio 17.	Advanced Custom Fields-kentän kutsuminen sivupohjassa.	s.38
Kuvio 18.	Advanced Custom Fields repeater field kutsuminen sivupohjassa.	s.38
Kuvio 19.	Advanced Custom Fields kenttä suodattimena.	s.39
Kuvio 20.	Custom post type määrittäminen functions.php-tiedostossa.	s.41

Kuvio 21.	Custom post type sisällön näyttäminen WP_Queryn avulla the loopissa.	s.42
Kuvio 22.	Esimerkki Archive.php-tiedostosta.	s.43
Kuvio 23.	WordPress taulurakenne ja table prefix.	s.46
Kuvio 24.	Table prefix arvon muuttaminen.	s.46
Kuvio 25.	PhpMyAdmin taulujen prefixin vaihtaminen SQL-komennolla.	s.47
Kuvio 26.	Options taulun prefix SQL-kysely.	s.47
Kuvio 27.	Usermeta taulun prefix SQL-kysely.	s.47
Taulukko 1.	WordPress käyttäjän roolit.	s.48

KESKEISET KÄSITTEET JA LYHENTEET

MySQL	Relaatiotietokantaohjelmisto
Back end development	Datan käsittelyä, jossa ei näytetä sisältöä loppukäyttäjälle. Datan näyttäminen loppukäyttäjälle tapahtuu front end -kehittämisessä.
Front end development	Vastaa kehityksen osasta, jossa näytetään sisältöä loppukäyttäjälle.
PHP	Web-sovelluksia varten kehitetty back end -ohjelmointikieli.
Bootstrap	Front end komponenttikirjasto
Javascript	Dynaaminen komentosarjakieli
CSS	Cascading Style Sheet tiedostotyyppi, joka vastaa verkkosivujen tyyliohjeista.
SASS	Skriptikieli joka on kehitetty laajentamaan CSS-ohjelmointikielen ominaisuuksia.
Core	Wordpressin ydinkoodi, joka sisältää WordPress-asennuksen oletustiedostot.
WordPress plugin	WordPress-lisäosa, joka laajentaa WordPressin toiminnallisuutta.

AngularJS	Googlen kehittämä javascript-ohjelmistokehys.
JSON	Objektipohjainen datan varastointi- ja välitysformaatti.

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on siirtää sivusto www.potilasturvallisuusyhdistys.fi WordPress-alustalle. Samalla uudistetaan sivuston ulkoasu ja parannetaan sisällön jäsentelyä. WordPress mahdollistaa asiakkaalle helpon sivuston päivittämisen. Opinnäytetyössä pyritään tekemään sivuston päivittämisestä mahdollisimman helppoa ja nopeaa. Opinnäytetyön on tilannut Awanic Oy ja Suomen Potilasturvallisuusyhdistys.

WordPress on tällä hetkellä ehdottomasti käytetyin sisällönjulkaisujärjestelmä eli CMS. Tällä hetkellä yli kuusikymmentä miljoonaa verkkosivua toimii WordPress-alustalla. Tässä opinnäytetyössä WordPress-julkaisujärjestelmästä käsitellään pääasiassa oman teeman kehitystä. Teemakehityksessä käytetään WordPressin omaa sivupohjajärjestelmää sivupohjien tekemiseen. WordPress on todella taipuisa ja monipuolinen. Sen toiminnallisuutta voi laajentaa lähes rajattomasti erilaisilla laajennuksilla eli lisäosilla tai luomalla itse yksilöityjä ratkaisuja. Olennaista on kuitenkin, ettei WordPress ydinkoodiin tehdä suoraan muutoksia. Työssä käytetään mahdollisuutta laajentaa WordPressin toiminnallisuuksia luomalla omia custom post type -sisältötyyppejä. Näin saadaan luotua asiakaskohtaisia rakenteita WordPressiin ilman, että WordPressin ydintä tarvitsee muokata.

Responsiiviset sivustot tarkoittavat sivustoja, jotka toimivat hyvin riippumatta siitä, millä laitteella sivustoja selataan. Tässä opinnäytetyössä käytetään responsiivisen toteutuksen avuksi Bootstrap-komponenttikirjastoa, joka sisältää paljon hyödyllisiä valmiita rakenteita, joita voidaan muokata tarpeiden mukaan. Bootstrap on hyvin testattu ja luotettava apuväline responsiiviseen verkkosivutoteutukseen. Erityisen hyödylliseksi Bootstrapin tekee sen grid-järjestelmä, joka mahdollistaa luotettavan sivurakenteen luomisen nopeasti.

Tämän opinnäytetyön aihe valittiin sen ajankohtaisuuden vuoksi. WordPress on edelleen kovassa nosteessa ja se on saavuttanut maineen luotettavana ja monipuolisena julkaisujärjestelmänä. Opinnäytetyön tekijällä on kokemusta jo entuudes-

taan WordPress-sivustojen ja -teemojen tekemisestä. Opinnäytetyön tavoitteena on syventää tekijän taitoja ja osaamista.

2 WORDPRESS

Wordpress on kehittynyt blogi-alustasta täysivaltaiseksi sisällönjulkaisujärjestelmäksi. Jotain kuvaa WordPressin mahdollisuuksista antaa se, että se pyörittää tällä hetkellä suurten mediatalojen sivustoja, kuten CNN's blogs, Wall Street Journal's All Things D ja Reutres. (Damstra, Stern & Williams 2015, 43). Mikä tekee WordPressistä niin suositun?

Yksi tärkeä syy on varmasti WordPressin hinta; se on ilmainen. WordPress on vapaan lähdekoodin järjestelmä ja se on julkaistu GPLv2-lisenssin alaisena. GPLv2-lisenssi mahdollistaa sen, että WordPressin alkuperäistä koodia saadaan muuttaa. Lisenssi on yksi tärkeä tekijä WordPressin voittokulussa. Käytännössä kaikilla on mahdollisuus alkaa julkaisemaan sivuja WordPressillä.

Laaja käyttäjäkunta ei ole pelkästään hyvä asia. Mitä enemmän tiettyä alustaa käytetään sitä mielenkiintoisempaa siihen murtautumisesta tulee. WordPress on tietoinen suosion tuomista ongelmista ja se julkaisee jatkuvasti tärkeitä tietoturvapäivityksiä, joiden tulee olla ajantasalla. Kappaleessa 9 WordPress ja tietoturva kerrotaan keinoja lisätä WordPressin tietoturvaa.

WordPress on nopea asentaa, eikä se vaadi erityisiä tietoteknisiä taitoja. WordPressiä voi laajentaa tarpeiden mukaan erilaisilla lisäosilla. WordPressin toiminnallisuuksien laajentaminen ei jää pelkkiin lisäosiin, vaan sen toiminnallisuuksia voidaan laajentaa myös kooditasolla esimerkiksi lisäämällä rakenteeseen omia funktioita. Tärkeää toiminnallisuuksien lisäämisessä on jättää WordPressin ydin manipuloimatta. Mikäli WordPressin ydinkoodia on manipuloitu, niin päivitettäessä uuteen versioon tehdyt muutokset häviävät ja sovellus voi hajota.

WordPressin tuki yltää hyvin pitkälle menneisyyteen. Hyvin todennäköisesti sivusto ei hajoa, vaikka päivitetään hyvin vanha WordPress-asennus uuteen versioon. WordPressillä on laaja ja kattava dokumentaatio osoitteessa www.codex.wordpress.org. Codexista löytää käytännössä kaiken tarvittavan tie-

don sivustojen muokkaukseen ja toteutukseen. Codex on yhtenä päälähteenä myös tässä opinnäytetyössä erityisesti käytännön osuudessa. WordPressin laajan käytön ja suosion takia sille on saatavilla laaja tukiverkosto erilaisiin ongelmatilanteisiin. Yksi tukiverkoista sijaitsee WordPress.org-palvelun keskustelufoorumilla. Keskustelufoorumilta löytyy kehittäjien ja käyttäjien kysymyksiä ja vastauksia ongelmatilanteisiin. WordPress-yhteisö on järjestänyt jo useamman vuoden ajan WordCamp-tapahtumia. WordCamp luo mahdollisuuden päästä tapaamaan muita WordPress-kehittäjiä, kuuntelemaan luentoja ja osallistumaan myös mahdollisiin WordPress-workshoppeihin. Ensimmäinen WordCamp Finland järjestettiin Tampereella 2015.

3 BOOTSTRAP

Bootstrap on vuonna 2011 julkaistu avoimen lähdekoodin komponenttikirjasto. Bootstrap on tällä hetkellä käytössä 65 872 sivustolla. (Boostrap 2015a) Voidaan siis puhua merkittävästä kirjastosta. Bootstrapin on kehittänyt Mark Otto ja Jacob Thornton. Otto ja Thornton olivat kehityksen aikana töissä Twitterillä, josta nimi Twitter Bootstrap. Kirjasto kehitettiin alunperin yhtenäistämään yrityksen front end -koodia. (Spurlock 2013, 1)

Bootstrapin CSS-luokkia kutsutaan, kuten mitä tahansa CSS-luokkaa html-elementin class-attribuutilla. Bootstrap oli aluksi vain CSS-kirjasto, mutta myöhemmin siihen on lisätty myös javascript-toiminnallisuuksia ja ilmainen ikonipankki. Bootstrapin hyödyllisin ja leimallisin osuus on kuitenkin sen grid-järjestelmä.

3.1 Grid-järjestelmä

Bootstrapilla rakennettu html-dokumentti koostuu kolmesta pääelementistä container, row ja column. Rakenne on hierarkinen ja sitä tulee noudattaa, muuten Bootstrapin toiminnasta oikein ei ole takeita. Row-elementin tulee olla aina container-elementin sisällä ja row-elementin sisälle sijoitetaan n-määrä column-elementtejä. Containereita on kahta eri lajia container ja container-fluid. Container on eri media queryissa määritelty tiettyyn kokoon, kun taas container-fluid -luokka on aina koko sivun leveyksinen. Bootstrap-komponentteihin ei tule osoittaa margin- tai padding-ominaisuuksia, muuten rakenne hajoaa. Mikäli padding halutaan kolumniin tulee padding-ominaisuus kohdistaa omaan esimerkiksi div-elementtiin, joka sijaitsee Bootstrap-kolumnin sisällä, joka havainnoillistetaan kuviossa 1.

```
<!-- BOOTSTRAP HIERARKINENRAKENNE -->

<div class="container">
    <div class="row">
        <div class="col-xs-12">
            <div class="padding-20px">
                <p>Sisältöä johon on lisätty padding 20px</p>
            </div>
        </div><!-- /col -->
    </div><!-- /row -->
</div><!-- /container -->

```

Kuvio 1. Esimerkki Bootstrap-hierarkiasta ja padding-ominaisuuden lisäämisestä kolumnin sisään.

Bootstrap jakaa sivun 12:sta kolumniin, joille voidaan määritellä eri leveys eri kokoluokissa, joita Bootstrapissa on viisi pienimmästä ruudun leveydestä suurimpaan xs, sm, md, lg ja xl. Kokoluokat määritellään leveyden mukaan. Ne on toteutettu CSS media queryilla. Kapeammat kuin 544 pikseliä käyttävät luokkaa col-xs, 544px–768 pikseliä leveät luokkaa col-sm, 768–992 pikseliä leveät luokkaa col-lg ja 1200 pikseliä leveämmät luokkaa col-xl. Eri koot on määritelty `bootstrap.css`-tiedostoon luokkina, joita voidaan kutsua. Kutsu tapahtuu aivan kuten normaalin css-luokan kutsu. Grid-luokan alkuun tulee aina col-, joka tulee sanast column eli sarake. col –luokan jälkeen määritellään ruudun leveys aiemmin mainituilla luokilla esimerkiksi pienille laitteille xs. Viimeiseksi määritellään, kuinka paljon leveyttä elementille halutaan määrittää tietyssä ruutukoossa. Esimerkiksi jos halutaan määrittää div-elementille leveys, joka on puolet ruudusta lg-koossa, kahdeksan kolumnia leveä md-koossa, mutta täysileveyksinen xs-koossa, määritellään se seuraavasti. `<div class=”col-xs-12 col-md-8 col-lg-6”>` Mikäli div-elementille määritellään leveys vain xs-koossa, tulee siitä saman leveyksinen jokaisessa kolmessa eri kokoluokassa.

Kun elementit ovat täysileveyksisiä, järjestyvät ne automaattisesti allekkain. Vasemmalla sijaitseva elementti yläpuolelle ja oikealla olevat alapuolelle. Mikäli järjestystä halutaan vaihtaa niin, että oikealla olevan divin tulee olla vasemmalla olevan div-elementin päällä xs-koossa, pitää käyttää Bootstrapin apuna pull- ja push-luokkia. Kuvailtu tilanne voidaan ratkaista esimerkiksi kuvion 2 mukaisesti.

```
<!-- PULL RIGHT -->

<div class="col-xs-12 col-md-6 pull-right">
    <p>alapuolella xs-koossa/oikealla md-koossa</p>
</div>

<div class="col-xs-12 col-md-6">
    <p>yläpuolella xs-koossa/vasemmalla md-koossa</p>
</div>
```

Kuvio 2. Esimerkki pull-right luokan käytöstä.

Miten toimitaan, jos halutaan luoda esimerkiksi kuusi saraketta leveä div-elementti keskelle ikkunanäkymää? Tarvitseeko luoda tyhjä elementti vasemmalle puolelle, joka on kolme saraketta leveä? Vastaus on: ei tarvitse. Kun Bootstrap-sivupohjassa halutaan keskittää div-elementti, voidaan se toteuttaa esimerkiksi käyttämällä offset-luokkaa. Offset on toteutettu CSS:n padding-ominaisuudella. Offset luo halutun kolumnimäärän mukaisesti tyhjää tilaa. (Bootstrap 2015a) Offset-määrittelyn alkuun tulee aikaisemmin mainittu col- ja sen jälkeen haluttu ruutukoko, jossa offset otetaan käyttöön. Luokan loppuun määritetään haluttu määrä offsetiä. Esimerkkitapauksessa määritellään luokka seuraavasti `<div class="col-md-offset-3">`. Huomioitavaa on se, että offset määritellään omana luokkana, eikä muiden grid-luokkien yhteydessä.

3.2 Ikonit

Bootstrap sisältää ikonikirjaston, glyphicons. Glyphiconit ovat fontti-ikoneita, joita kutsutaan span-elementin sisällä luokalla glyphicon ja tarkentamalla kutsun tiettyyn ikoniin. Esimerkiksi hakukentistä tutun suurennuslasin voi kutsua html-koodissa seuraavasti. `<span class="glyphicon glyphicon-search"></span>` Huomioitavaa on, ettei ikonin kutsua saa sekoittaa toiseen komponenttiin, vaan kutsu tulee tapahtua tyhjässä span-elementissä. (Bootstrap 2015b) Ikoni kohdistetaan span-elementin :before-valitsimeen. Ikoni tulostuu span-elementin eteen pseudo-elementtinä. Kuviosta 3 voi katsoa, kuinka *search glyphicon* on määritelty `bootstrap.css`-tiedostossa.

```
313   .glyphicon-search:before {
314     content: "\e003";
315   }
```

Kuvio 3. Glyphicon `bootstrap.css`-tiedostossa.

Fontti-ikonit ovat vektorikuvioita, joten niitä voi skaalata ilman, että kuvan tarkkuus kärsii. Myös bootstrapin tarjoamat ikonit ovat lisenssiltään CC BY 3.0. alaisia, joten niitä voi käyttää ilmaiseksi myös kaupallisiin tarkoituksiin. Tällä hetkellä ikoneita on 250 kappaletta. Ikonit löytyvät Boostrapin kotisivuilta yläpalkista Components. Bootstrapin tarjoamat ikonit eivät ole ainoita Creative Commons -lisenssin alaisia ikoneita. Yksi suosituimmista Creative Commons -ikonipankeista on Font Awesome, joka sisältää hyvin laajan ikonikirjaston.

3.3 Asentaminen

Bootstrapin sivustolla www.getbootstrap.com voi ladata Bootstrapin uusimman version etusivulta. Tarjolla on myös MaxCDN ylläpitämä bootsrap CDN-linkki. Bootstrapin sivuilta löytyy laaja dokumentaatio, josta näkee ajantasalla tavat käyttää Bootstrap-ohjelmointikehystä oikein, sekä kaikki tarjolla olevat komponentit.

Bootstrap asennetaan WordPressiin jonottamalla latauksessa mukana tulleet tyylitiedostot `wp_enquque_style()` -funktiolla. Tarkempi kuvaus funktion käytöstä löytyy kappaleesta 5.1.3 Tyylitiedostojen jonottaminen. Bootstrap-asennukseen kuuluu myös javascript-tiedostoja, jotka voidaan asentaa `wp_enqueue_script()`-funktiolla, joka toimii hyvin samaan tapaan kuin `wp_enqueue_style()`-funktio. Funktiolle määritetään oikea polku tiedostoon ja oikeat parametrit oikeille paikoilleen funktion kutsussa ja näin saadaan lisättyä javascript-tiedosto turvallisesti WordPress-sovellukseen.

Tässä opinnäytetyössä käytetään Bootstrapin virallista Syntically Awesome Stylesheets eli Sass-käännöstä, jonka saa ladattua Bootstrapin kotisivuilta. Sass-käännönnästä käsitellään enemmän kappaleessa neljä. Käännöstä ylläpidetään

avoimen lähdekoodin hengessä github-palvelussa. Bootstrap on saatavilla myös Less-käännöksellä.

4 SASS

Syntically Awesome Stylesheets eli Sass on skriptikieli, joka on kehitetty laajentamaan CSS-kielen ominaisuuksia. CSS-tyylitiedostot voivat kasvaa nopeasti hyvin isoiksi, eikä CSS-syntaksi ole luettavuudeltaan kovin hyvää. Sass parantaa tyylitiedostojen luettavuutta huomattavasti. Yksi merkkittävimmistä ominaisuuksista luettavuuden kannalta on Sassin syntaksin sisentäminen. Sass käyttää syntaksissa html-kieltä muistuttavaa hierarkiaa. Sitä on erittäin helppo ja selkeä lukea. Kuviossa 4 on vertailun vuoksi esimerkki, jossa määritetään harmaa taustaväri ul-lapsielementtiin.

```
4
5   /* CSS */
6   nav > ul {
7       background-color: #ccc;
8   }
9
10  // SCSS
11  nav {
12      ul {
13          background-color: #ccc;
14      }
15  }
16
```

Kuvio 4. Taustavärin määritys lapsielementtiin Sass-kielellä.

4.1 Muuttujat

Sass mahdollistaa muuttujien käytön tyylitiedostoissa. Muuttuja määritellään lisäämällä dollari-merkki, muuttujan nimi, kaksoispiste, muuttujaan varastoitava arvo ja loppuun puolipiste. Muuttujat nopeuttavat ja selkeyttävät CSS-kehitystä. Hyvä esimerkki muuttujien käytöstä on värien määrittäminen. Väriarvot määritellään heksalukuina ja helposti arvot voivat mennä sekaisin kehitysprosessin edetessä, mutta ei Sass-kielessä. Sass-kielessä voidaan määritellä värikoodille muuttuja, johon varastoidaan väriä vastaava heksaluku. Tämän jälkeen väriä voi kutsua suoraan muuttujan nimellä halutuissa kohdissa kuten kuviossa 5 Sass-muuttujan määrittely ja kutsuminen.

```
30
31  // SCSS
32  $harmaa: #ccc;
33
34  nav {
35      ul {
36          background-color: $harmaa;
37      }
38  }
39
```

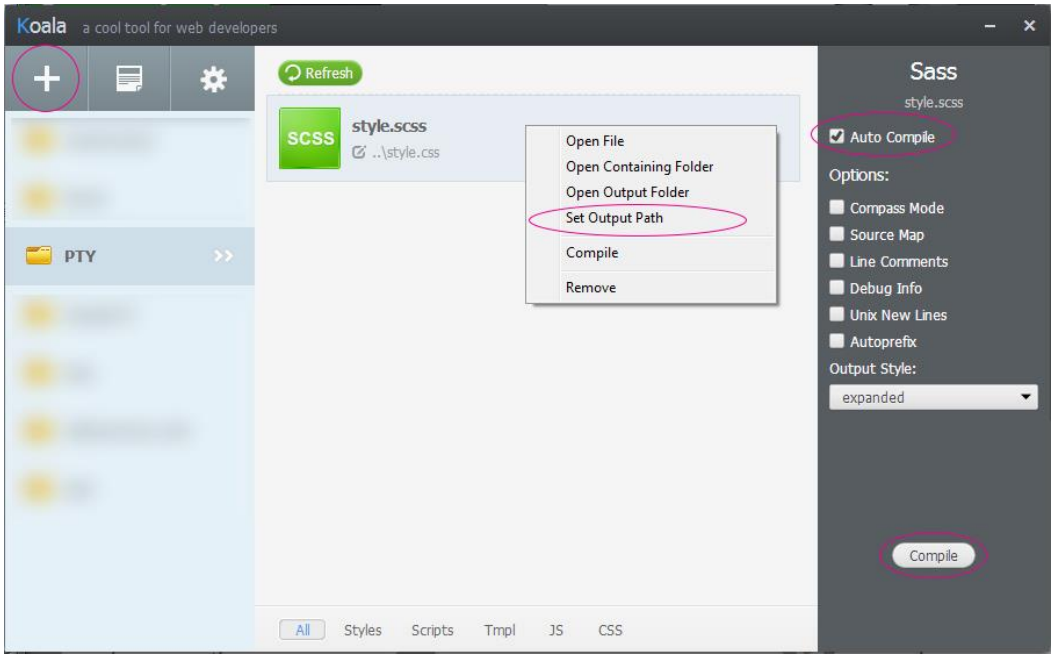
Kuvio 5. Sass-muuttujan määrittely ja kutsuminen.

Värin määrittelyn ja kutsumisen jälkeen voi nopeasti vaihtaa värejä koko dokumentissa muuttamalla muuttujien arvoja.

Sass mahdollistaa myös laskemisen tyylitiedostossa. Täytyy kuitenkin muistaa, että lopuksi tiedosto käännetään CSS-kieleen, joten reaaliaikaisia laskutoimituksia tyyleissä ei voida toteuttaa Sass-tiedostoilla. Jos reaaliaikaista arvojen laskua CSS-tiedostossa halutaan tehdä täytyy avuksi ottaa javascript-kieli.

4.2 Sass-tiedoston kääntäminen CSS-tiedostoksi

Selaimet eivät osaa suoraan lukea Sass-kieltä, joten Sass-tiedostot tulee kääntää selaimelle ymmärrettävään muotoon. Tiedostot käännetään CSS-muotoon. Käännettävien Sass-tiedostojen pääte on scss. Työkaluja Sass-käännösten tekemiseen on saatavilla runsaasti. Opinnäytetyössä käytettiin graafistaversiota Koala, jonka voi ladata koala-app.com-sivustolta. Koala-sovelluksessa määritellään projektille kansio ja sen Sass-tiedoston sijainti, sekä käännettävän CSS-tiedoston sijainti kansiorakenteessa. Lopuksi kun sijainnit on määritelty voidaan painaa oikealta ”compile” ja Koala kääntää SCSS-tiedoston haluttuun kansioon. On mahdollista asettaa myös Koala kääntämään SCSS-tiedostot automaattisesti CSS-tiedostoiksi aina kun SCSS-tiedosto tallennetaan tekstieditorissa. Kuvattu toiminto saadaan päälle valitsemalla ”Auto Compile”. Kuvio 6 Koala SCSS-tiedoston kääntäminen CSS-tiedostoksi havainnollistaa Koalan käytön helppoutta.



Kuvio 6. Koala Sass-kääntäjän käyttäminen.

5 TEEMAKEHITYS

Teemat ovat yksi pääalue WordPress-kehityksessä. Teema vastaa suuresta osasta projektikohtaisesta kustomoinnista. Teeman olennaisin tehtävä on vastata sivuston ulkoasusta ja datan sijoittelusta käyttäjälle. WordPress sisältää oman template-järjestelmän, joka koostuu erilaisista sivupohja-funktioista, jotka näyttävät haluttua dataa tietyssä kohdassa sivua. Teemoilla on ulkoasun lisäksi myös muita tehtäviä. Teemat voivat sisältää myös paljon toiminnallisuuksia. Tässä opinnäytetyössä käsitellään teemoja WordPressin oman sivupohjajärjestelmän näkökulmasta. Muita vaihtoehtoja on esimerkiksi käyttää WP REST API -rajapintaa, joka palauttaa sivun sisällön JSON -datana. WP REST API:a käytetään apuna esimerkiksi javascript-sivupohjiin, joilla luodaan yhden sivun sovelluksia.

5.1 Teeman tiedostot

Teeman tiedostot sijaitsevat `wp-content/themes` -kansiossa. Teemalle luodaan themes-kansion sisään oma alakansio, joka nimetään teeman mukaisesti. Teemalle tarvitsee luoda vähintään `index.php` ja `style.css` tiedostot. Ilman näitä tiedostoja WordPress ei tunnista kansion sisältöä teemaksi. Yleensä teemaan kuuluu useita muita tiedostoja, joista yhtenä tärkeimpänä jo aikaisemmin mainittu `functions.php`.

5.1.1 Style.css

`Style.css`-tiedosto on teeman ydin. Se yksilöi teeman ja antaa sille sen tarvitsemat metatiedot, kuten teeman nimi, tekijän nimi, lisenssi ja versionumero. Teeman nimi on tärkeä graafisen käyttöliittymän kannalta, sillä tällä nimellä käyttäjä näkee teeman käyttöliittymän teemojen valinta -kohdassa. Graafisen käyttöliittymän kautta saatavilla olevat teemat löydät kohdasta *Appearance* > *Themes*. Kyseisestä sijainnista voi aktivoida halutun teeman. On mahdollista luoda teemalle `screenshot.jpg` -tiedosto teeman juureen, joka näkyy teemaa valitessa. Tämä auttaa käyttäjää tunnistamaan teeman ja tuo teemalle lisää brändiarvoa.

Teeman tunnistaminen on yksi `style.css` tiedoston tärkeä tehtävä, mutta ei ainoa. `Style.css`-tiedosto vastaa teeman ulkoasusta ja datan sijoittelusta teemassa. `Style.css` on täysin normaali CSS-tiedosto metadatan määrittelyä lukuunottamatta. Tärkeää on sijoittaa tiedosto teeman kansion juureen, muuten WordPress ei tunnista teemaa.

5.1.2 Functions.php

Functions.php vastaa teeman toiminnallisuuksista ja resursseista. `Functions.php`-tiedostossa määritellään teemassa käytettävät omat funktiot ja koukut joilla voidaan vaikuttaa WordPressin ajamiseen ja muuttaa WordPressin toiminnallisuuksia. Mikäli tehdään monilla sivustoilla käytettäviä funktioita, niin jossain kohdassa tulee vastaan tilanne, kun toiminnallisuudet kannattaa erottaa WordPress-lisäosaksi.

`Functions.php`-tiedostoon määritellään teemassa käytettävät tyylitiedostot ja niiden polut. `Wp_enqueue_style()`–funktiolla saadaan liitettyä teemaan CSS-tyylitiedostot. Tyylit voidaan liittää myös perinteisesti head-osiossa ja kaikki toimisi todennäköisesti hyvin, mutta `wp_enqueue_style()`–funktion avulla lisätyt tyylitiedostot toimivat varmemmin yhteen WordPress-asennuksen kanssa. `Functions.php`-tiedostoon voidaan lisätä myös *custom post type* -sisältötyyppien määrittely. Aluksi rekisteröimällä uusi *custom post type* ja sen jälkeen lisäämällä `add_action()`-funktiolla. Custom post typen määrittelyä käsitellään luvussa 8 Custom post type määritteleminen.

Functions.php-tiedostoon rekisteröidään teemassa käytettävät menut. Rekisteröinti suoritetaan `register_nav_menus()`-funktiolla. Rekisteröinnin jälkeen pitää vielä suorittaa `add_action`-funktio, joka kytkee luodun funktion WordPress koukkuun. `Add_action()` ottaa vastaan kaksi parametriä ensimmäinen on koukku, johonka funktio liitetään. Toinen parametri on funktion nimi, joka halutaan liittää koukkuun. Näiden toimintojen jälkeen menu näkyy admin-näkymässä.

```
81
82      // Navigaation rekisteröinti
83      function register_theme_menus() {
84          register_nav_menus(
85              array(
86                  'primary' => __( 'mainnavigation' ),
87                  'secondary' => __( 'subnavigation' )
88              )
89          );
90      }
91
92      add_action('init', 'register_theme_menus');
93
```

Kuvio 7. Menun rekisteröinti functions.php-tiedostossa.

5.1.3 Tyylitiedostojen jonottaminen

WordPress-teeman tyylitiedostot lisätään yleensä teemaan `functions.php`-tiedostossa jonottamalla tyylitiedostot `wp_enqueue_style()` –funktion avulla. WordPress tyylitiedoston jonottaminen kääritään funktion sisälle, mille voidaan antaa mielivaltainen nimi esimerkiksi `malli_jonotus()`. `malli_jonotus()` –funktion sisällä jonotetaan tyylitiedosto WordPress-funktiolla `wp_enqueue_style()`, joka ottaa vastaan viisi parametria `$handle`, `$src`, `$deps`, `$ver` ja `$media`. `$handle` on tyylitiedoston nimi ja sen tulee olla uniikki. `$src` on polku, jossa tyylitiedosto sijaitsee. Polun määrittämiseen voidaan käyttää apuna `get_stylesheet_uri()` -funktiota, joka palauttaa teeman `style.css`-tiedoston sijainnin. Yleensä tämä sijainti vastaa teeman juurta ja näin ollen voidaan helposti osoittaa missä kansiossa haluttu tiedosto sijaitsee hierarkisessa suhteessa `style.css`-tiedostoon. `$deps` kohtaan listataan tyylitiedoston riippuvuudet toisista tyylitiedostoista. `$ver` on tyylitiedoston versionumero. `$media` kohtaan voidaan määrittää sääntöjä mille eri ruutukoolle kyseistä tyylitiedostoa halutaan käyttää. (Developer WordPress 2016a)

Lopuksi funktio `malli_jonotus()` kutsutaan `add_action()` –funktion avulla tietyssä WordPress-koukussa. Koukkujen avulla voidaan WordPressiä ajettaessa ajaa oma funktio halutussa kohdassa. Tyylitiedosta rekisteröitäessä halutaan tehdä koukku init-koukkuun, joka ajetaan kun WordPress on latautunut, mutta headereita ei ole vielä lähetetty. Toiseksi parametriksi `add_action()` –funktioon laitetaan luotu funktio `malli_jonotus()`. Näillä toimienpiteillä

voidaan jonottaa tyylitiedosto luotettavasti WordPress sovellukseen. (Developer WordPress 2016b)

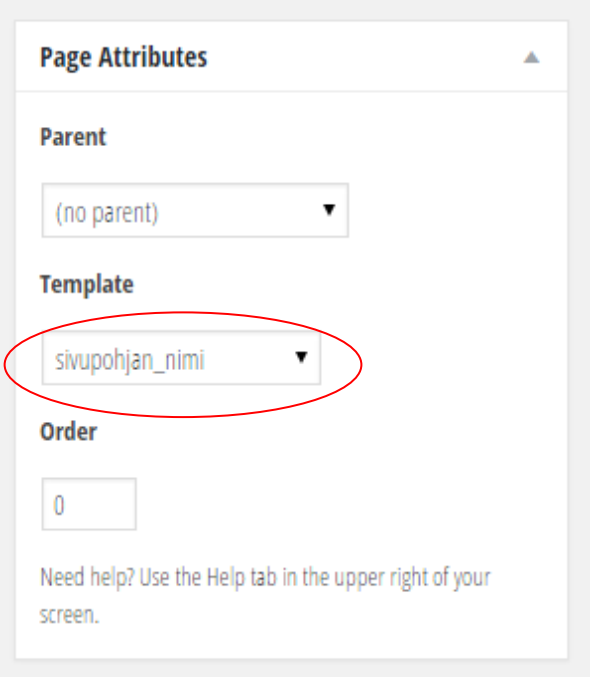
5.2 Sivupohja-tiedostot ja hierarkinen malli

Teeman sivujen ulkoasu koostuu tyylitiedostosta `style.css` ja itse sivupohja-tiedostosta, joita teemassa on joko yksi tai useampia. WordPress-teema sisältää vähintään `index.php`-sivupohjan. Sivupohjat ovat järjestyneet WordPressin sisällä hierarkiseen järjestelmään. WordPress-hierarkia on monimutkainen kokonaisuus, jonka ymmärtäminen on välttämätöntä teemakehitykselle. Voimassa olevan sivuhierarkian löytyy wordpress.org-sivuilta kohdasta https://developer.wordpress.org/themes/basics/template-hierarchy/. Mikäli WordPress ei löydä oikean nimistä sivupohjaa sivulle, ottaa se käyttöön `index.php`-tiedoston rakenteen. Sivupohjan voi määritellä yksittäiselle sivulle, artikkelille tai arkistosivulle. Sivupohjatiedostot ovat PHP-tiedostoja, jotka sijoitetaan teeman juureen. Sivupohja nimetään kirjoittamalla tiedoston yläosaan PHP-kommentin sisään *Template name:* ja sen jälkeen *sivupohjan nimi*, josta on esimerkki kuviossa 8.

```
1  <?php
2  /*
3   * Template Name: sivupohjan_nimi
4   *
5  */
6
7  ?>
```

Kuvio 8. Sivupohjan nimeäminen sivupohjatiedostossa.

Graafisessa käyttöliittymässä WordPress-käyttäjä löytää sivupohjan nimen sivun luomisnäkymästä kohdasta template, jota havainnollistetaan kuviossa 9. Huomioitavaa on, ettei sivupohjan tiedostonimellä ole merkitystä sivupohjan nimen kannalta.



Kuvio 9. Kuvankaappaus sivun luomisnäkymästä.

5.3 Sisältötyypit

Teemakehityksen kannalta on tärkeä tietää, minkälaista dataa tallennetaan järjestelmään. Onko data blogihenkistä vai täysin yksilöityä vain asiakaskohtaista dataa. WordPress sisältää oletuksena viisi eri sisältötyyppiä post, page, attachment, revision ja nav menus. Kaikkien sisältötyyppien sisältöä säilytetään WordPressin tietokannan `wp_posts`-taulussa, jossa sijaitsee myös kenttä `post_type`, joka määrittelee sisältötyypin. WordPressin oletussisältötyypit riittävät hyvin yksinkertaisiin sivustoihin, mutta jos halutaan toteuttaa monimutkaisempaa sivustorakennetta, kannattaa miettiä oman sisältötyypin luomista. Sen mahdollistaa WordPressin ominaisuus *custom post type*, jota käsitellään tarkemmin luvussa 5.3.5.

5.3.1 Post

Post-tyyppi sisältää artikkeleita, joita käytetään esimerkiksi blogisisällön luomiseen. Oletuksena Post-tyypin artikkelit järjestyvät aikajärjestykseen. Post- sisältötyypille voidaan oletuksena määritellä kategorioita, joiden mukaan artikkeleja on

helpompi jäsentää. Artikkelien kategoria määritellään julkaisuvaiheessa admin-näkymässä. Julkaisija voi lisätä jo valmiiksi lisätyn kategorian listasta tai lisätä kokonaan uuden kategorian, jos tarve vaatii. Kuten todettua kategorioiden lisääminen post-sisältötyyppiin ei tarvitse erillistä koodaamista, vaan niiden hallinta tapahtuu admin-näkymästä suoraan. Mikäli käytetään *custom post type* -sisältötyyppi tarvitsee kategoriat ottaa erikseen käyttöön, kun sisältötyyppiä rekisteröidään.

5.3.2 Page

Page-tyyppi sisältää staattisia sivustoja hierarkisessa järjestyksessä. Hierarkinen rakenne mahdollistaa ylä- ja alasivujen luonnin. Sivuille on mahdollista luoda erillisiä sivupohjia, joita voidaan käyttää yhdellä tai useammalla sivulla.

5.3.3 Attachment

Attachement sisältää kaiken tietueeseen liittyvän median, jota on ladattu WordPressiin, kuten kuvia ja tiedostoja. Attachment tyypin postauksia voidaan selata ohjausnäkymästä kohdasta media. Attachement voidaan liittää tietokannassa tiettyyn postaukseen.

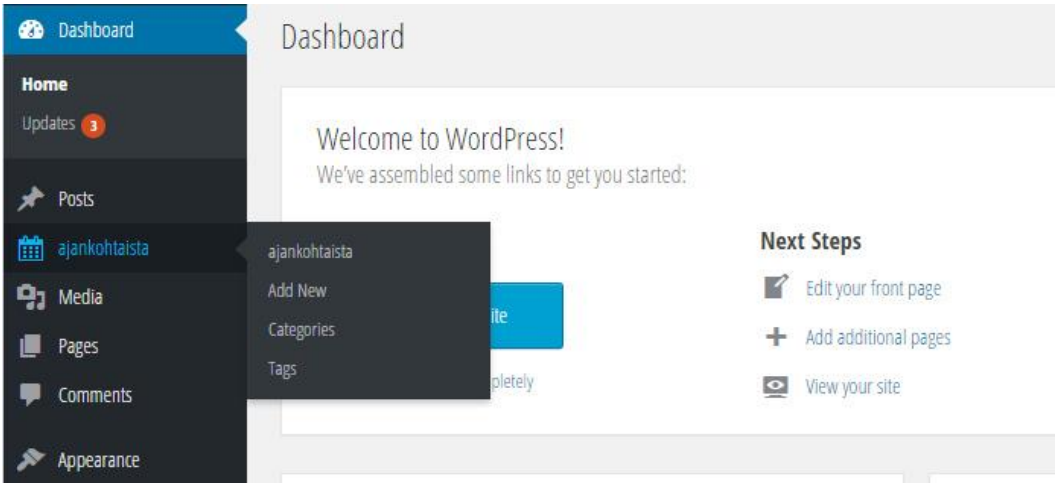
5.3.4 Nav menus

Nav menus sisältää navigaatioelementtejä, joita voidaan liittää navigaatioon WordPressin menu management -ominaisuudella. Teemaan voidaan rekisteröidä omia navigaatioita. Aluksi navigaatiovalikko rekisteröidään `functions.php`-tiedostossa `register_navigation_menus()`-funktiolla. Rekisteröintifunktiossa valikolle määritetään nimi jolla navigaatio tunnistetaan koodissa ja selite, joka näkyy admin-valikossa kohdassa ”Theme Locations”.

5.3.5 Custom Post Type

Custom post type lisättiin WordPress coreen versiossa 3.0 (WordPress Codex 2015a). Se mahdollistaa oman sisältötyypin luomisen oman tarpeen mukaan. Esi-

merkiksi, jos sivuilla on uutisosio, voidaan luoda sisältötyyppi uutiset, jolle määritellään uutisen tarvittavat ominaisuudet. *Custom post type* helpottaa sisällön hallintaa ja jäsentelee sitä selkeästi. Käyttöliittymäpuolella *custom post type* parantaa myös käyttäjän käyttökokemusta. Kun luodaan uusi custom post type, tulee graafiseen käyttöliittymään sen tyypin nimi omana navigaatioelementtinä (kuvio 10). *Custom post type* on nimensä mukaisesti yksilöllinen sisältötyyppi.



Kuvio 10. Custom post type admin-näkymässä.

Custom post typen yksi suuri etu on se, että sille voidaan tehdä omia sivupohjia sisältötyypin tai sen taksonomian mukaan. Sivupohjat muodostetaan, kuten mikä tahansa WordPress-sivupohja. *Custom post type* -sivupohjaa määriteltäessä tärkeintä on tiedoston nimi. WordPress tunnistaa *custom post typen* sivupohjat tiedostonimen mukaan. Sivupohjaa ei liitetä julkaisuun admin-näkymän kautta vaan WordPress tunnistaa sivupohjan yhteyden sisältötyyppiin sivupohjan tiedostonimestä. Listaussivu, joka näyttää kaikki tulokset tietyn sisältötyypin mukaisesti. Listaussivu nimetään `archive-[sisältötyypinnimi].php`. Sisältötyypin nimi kirjoitetaan, kuten se on määritelty `register_post_type()`-funktiossa. Yksittäinen tietue sisältötyypistä voidaan näyttää sivupohjassa `single-[sisältötyypinnimi].php`. Single-sivupohjassa sisältö voidaan näyttää The loopin avulla, kuten muissa sivupohjissa.

Custom post typen toteutukseen on kaksi eri tapaa, joko toteutus tehdään suoraan teeman `functions.php`-tiedostoon tai toteutuksesta tehdään oma lisäosa. *Custom post typen* määrittely kooditasolla ei eroa merkittävästi näissä mainituissa toteutustavoissa. *Custom post type* rekisteröidään molemmissa samalla tavalla. Ero toteutustapojen välillä on lähinnä rakenteellinen.

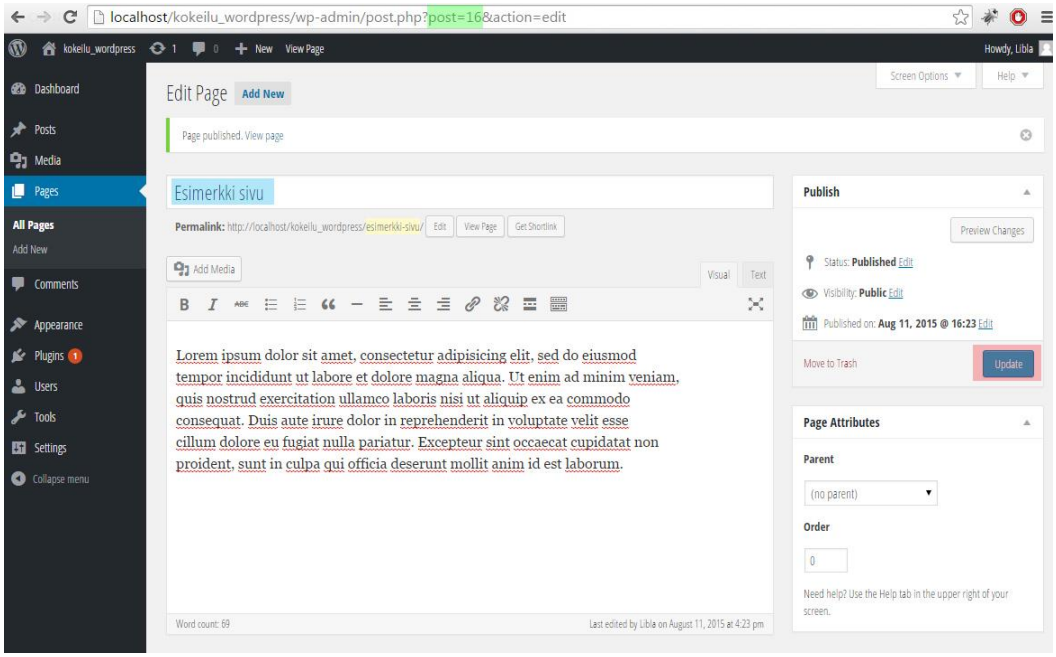
Lisäosaan tehtävä *custom post type* määritellään PHP-tiedostoon, joka luodaan `wp-content/plugins`-kansioon. Tiedostolle tehdään pluginille pakolliset määritykset, kuten nimi. Tämän jälkeen WordPress löytää tehdyn lisäosan. Lisäosassa toteutettu *custom post type* on nopea laittaa ja ottaa pois päältä WordPressin admin-näkymästä. Kun aloitetaan uutta projektia ei tarvitse, kuin kopioida tehty lisäosa, liittää se `plugins`-kansioon ja aktivoida plugin. Tämän jälkeen lisäosa näkyy WordPressin lisäosissa. Lisäosaan kannattaa määritellä sellaisia sisältötyyppejä, joita voi käyttää useissa projekteissa esimerkkinä voidaan antaa esimerkiksi uutiset, tuotteet ja arviot. Lisäosan etuna on myös koodin eroteltavuus. Lisäosa ei sijaitse teeman tiedostojen seassa, vaan eroteltuna `wp-content/plugins`-kansiossa.

Teeman `functions.php`-tiedostoon tehtävät *custom post typet* ovat tiettyyn projektiin liittyviä yksilöllisiä sisältötyyppejä. Niiden päälle laittaminen tapahtuu kooditasolla määrittelemällä *custom post typet* `functions.php`-tiedostoon. Kertauksena voidaan siis sanoa, jos tarvitaan samaa sisältötyyppiä monessa projektissa tehdään siitä laajennusosa, jos taas sisältötyyppi on yksilöllinen vain yhteen projektiin sopiva toteuta se teeman `functions.php`-tiedostossa.

6 SISÄLLÖN SYÖTTÄMINEN WORDPRESSIIN

6.1 Sisällön syöttäminen sivuille ja artikkeliin

Yksinkertaisimmillaan käyttäjän sisällön syöttäminen WordPressiin on hyvin yksinkertaista. Admin-näkymässä on oletuksena yksi sisällönsyöttökenttä tekstille ja media-sisällölle oletuskenttä on tyypiltään *Wysiwyg*. Wysiwyg-kenttään voidaan syöttää montaa eri tyyppistä sisältöä. Wysiwyg vastaa ominaisuuksiltaan tekstinkäsittelyohjelmistoa. Käyttäjä voi määritellä tekstin sijoittelun ja tyylit graafisesta käyttöliittymästä. Lisäksi Wysiwyg-kenttä tukee myös media-sisältöä joko sivuston sisäisesti tai upotettuna muilta sivustoilta. Tässä opinnäytetyössä käsitellään myös muita kenttävaihtoehtoja, kuten *Advanced custom fields*-laajennuksen tarjoamia vaihtoehtoisia tapoja syöttää sisältöä WordPressiin. Erotelluilla kentillä verkkosivun kehittäjä voi vaikuttaa merkittävästi sisällönsyötön helppouteen, jäsentelyyn sekä loogisuuteen.



Kuvio 11. Oletus sisällönsyöttönäkymä Wysiwyg-kentällä.

Kuviossa 11 näkyy sisällönsyötön kannalta kriittiset kohdat. Kuvassa sinisellä on korostettu kenttä, johonka syötetään sivun otsikko. WordPress luo sivulle auto-

maattisesti autoincrement id:n, jonka näkee sivun luomisnäkymässä url-osoitteesta kohdasta `post=[julkaisun id]`. Kuvassa id on korostettu vihreällä värillä. Oletuksena WordPress luo automaattisesti otsikon pohjalta sivulle tai julkaisulle myös kestolinkin, joka sijaitsee otsikon alla. Kestolinkkiä voidaan muuttaa manuaalisesti edit-painikkeesta, joka sijaitsee kestolinkin oikealla puolella.

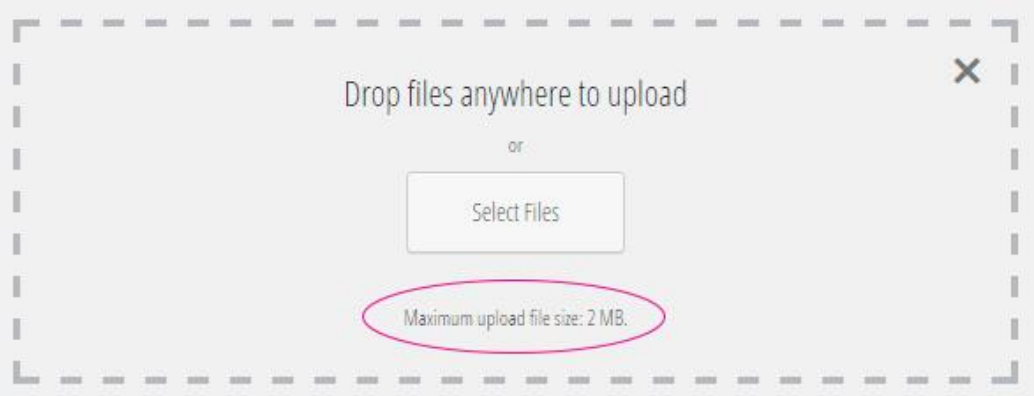
Punaisella korostetusta painikkeesta sivu voidaan joko julkaista tai päivittää, jos editoidaan valmiiksi luotua sivua.

6.2 Media-kirjasto

Media-kirjasto on kuten nimestä voi päätellä paikka, johonka varastoidaan sivustoon liittyviä tiedostoja; kuvia, dokumentteja, äänitiedostoja ja videota. (WordPress Codex 2015b) Media-kirjaston lisäksi tiedostoja voidaan lisätä myös suoraan julkaisuun sivun sisällönsyöttönäkymässä. vaikka kuva lisätään suoraan julkaisuun näkyy se lisäyksen jälkeen myös mediakirjastossa.

Mikäli tiedostoja ladataan ensimmäistä kertaa WordPress-asennukseen luo WordPress kansion uploads `wp-content`-kansion sisään. Tämän sisään WordPress luo oletuksena kansioita sen mukaan, koska tiedosto on ladattu asennukseen. `Uploads`-kansion alikansiot on nimetty automaattisesti vuoden ja kuukauden mukaan (Damstra, Stern & Williams 2015, 42).

Usein suomalaisilla sivustoilla sivusto on käännetty eri kielille käyttämällä käännöslisäosaa. Polylang-lisäosan kanssa on huomioitava, että tiedostot eivät näy suoraan kaikille kielille, jos media-tiedostot on asetettu käännetäväksi sisällöksi.



Kuvio 12. Media-kirjasto tiedoston maksimikoko näkyy latausnäkymässä.

Yksi huomioitava asia tiedostojen lataamisessa on tiedostojen maksimikoko. Kuviossa 12 nähdään ympyröidyssä kohdassa, että tiedostojen maksimikooksi on määritetty 2 megatavua. WordPressin PHP-koodissa arvoa voidaan kysyä funktiolla `wp_max_upload_size()`, joka palauttaa maksimikoon tavuina. (WordPress Codex 2015c) Asiakkaalle ei kannata antaa liian korkeaa latauskokoa tiedostoille, vaan arvo tulee säätää tarpeiden mukaan oikeaksi. Arvoa voidaan teoriassa vaihtaa ini_set-komennoilla `functions.php`-tiedostossa, mutta toimittaessa oikean webhotellin kanssa tämä harvemmin on mahdollista. Useimmiten palveluntarjoaja ei halua että PHP-asetuksia voidaan vaihtaa asiakkaan puolelta tietoturvasyistä. Muutokset latauskokoon tehdään palvelimen puolelta muokkaamalla `php.init`-tiedoston arvoja `max_upload_size` ja `post_max_size`.

7 SISÄLLÖN NÄYTTÄMINEN TEEMASSA

7.1 The loop

The loop on WordPressin sisällön näyttämisen perusta. The loop on PHP-koodi, jonka avulla voidaan näyttää admin-näkymässä määriteltyä sisältöä sivupohjassa. The loop käy lävitse jokaisen julkaisun ja näyttää sisältöä sen mukaan, miten sitä on kutsuttu loopin sisällä. Kaikki the loopin sisällä määritelty PHP- ja html-koodi prosessoidaan jokaisen julkaisun kohdalla. The loopin sisällä voidaan käyttää paljon hyödyllisiä Wordpressin tarjoamia sivupohja-funktioita. Niiden avulla voidaan hakea esimerkiksi sivun otsikko `the_title()`, aika `the_time()` tai kategoria `the_category()`. (WordPress Codex 2015d) The loop määritellään sivupohjassa yleensä kuvion-”The loop määritteleminen sivupohjassa” mukaisesti.

```
 8
 9  <!-- Aloitetaan loop -->
10  <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
11
12      <!-- Otsikko -->
13      <?php the_title('<h1>', '</h1>'); ?>
14
15      <!-- Sisältö -->
16      <div class="content-wrap">
17          <?php the_content(); ?>
18      </div>
19
20
21  <!-- Suljetaan loop -->
22  <?php endwhile; else : ?>
23      <p><?php _e( 'Ei löytynyt haluamaasi sisältöä.' ); ?></p>
24  <?php endif; ?>
25
```

Kuvio 13. The loop määritteleminen sivupohjassa.

Aluksi katsotaan, onko sivulla yhtään julkaisua. Jos julkaisuja löytyy näytetään jokainen löydetty julkaisu ja niistä kutsuttu sisältö while-loopin sisällä. Kuvion 13 mukainen määrittely tulostaa otsikon ja WordPressin Wysiwyg-kentän sisällön. The loopin sisällä voitaisiin kutsua myös yksilöityjä kenttiä custom fieldsejä, näitä käsitellään paremmin luvussa 7.3. Advanced Custom Fields.

The looppia on mahdollista käyttää muokattuna myös näyttämään sisältöä tiettyjen suodattimien mukaisesti. On mahdollista näyttää sisältöä esimerkiksi tietyn

kategorian tai sisältötyypin mukaisesti. Luvussa 8.1 Custom post type sisällön näyttäminen käsitellään *custom post type* sisältötyypin sisällön näyttämistä sivupohjassa.

7.2 Sivupohja-funktiot

WordPress tarjoaa kattavan määrän valmiita sivupohja-funktioita, joilla voidaan näyttää julkaisuun liittyvää dataa helposti pienellä määrällä koodia. Sivupohja-funktiot ovat PHP-funktioita, joten niitä tulee kutsua PHP-tagien sisällä. Sivupohja-funktiot näyttävät vain tarvittavan datan halutussa kohdassa. Kaikelle datalle voidaan siis määritellä tarkka sijainti sivupohjassa.

Sivupohja-funktiot tulostavat sisällön, joko suoraan sivulle siihen kohtaan johon kutsu on asetettu tai palauttavat halutun arvon sivupohjassa. Esimerkkinä voidaan antaa esimerkiksi otsikon näyttäminen sivupohjassa. `The_title()`-funktio tulostaa otsikon suoraan siinä kohdassa, johon kutsu on sivupohjassa toteutettu, kun taas `get_the_title()` palauttaa vain halutun julkaisun otsikon sisällön. `get_the_title()`-funktio ottaa vastaan yhden parametrin julkaisun ID:n. Jotta otsikko voidaan näyttää `get_the_title()`-funktiolla, tulee se tulostaa PHP:n echo-komennolla. Get-etuliitteellä alkavat sivupohja-funktiot palauttavat useimmiten vain dataa, kun taas the-alkuiset tagit näyttävät sisältöä suoraan kutsutussa kohdassa. Sivupohja-funktioita on satoja ja ne kaikki löytää WordPress codex-dokumentaatiosta.

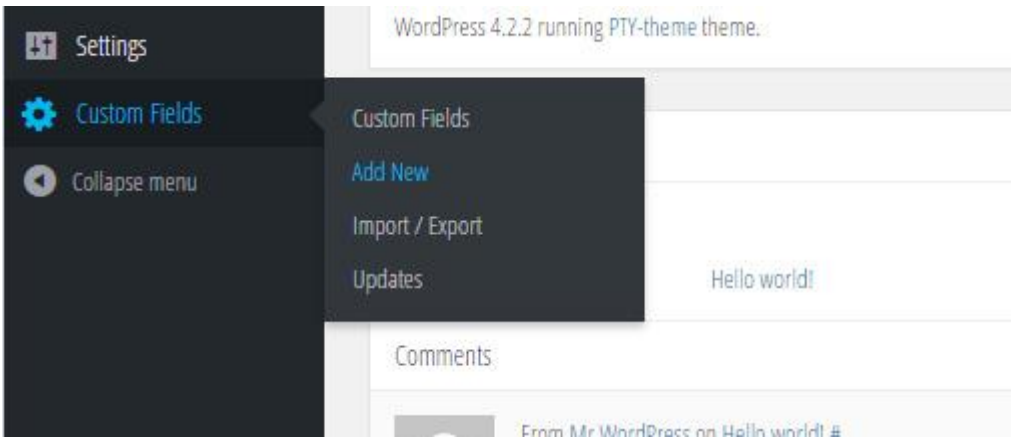
7.3 Advanced Custom Fields

Advanced custom fields on suosittu WordPressin lisäosa, joka tarjoaa valmiita sisällönsyöttökenttiä admin-näkymään. Tällä hetkellä Advanced Custom Fieldsistä on yli 900 000 aktiivista asennusta. (WordPress Plugins 2015) Oletuksena Wordpress tarjoaa Wysiwyg-sisällönsyöttökentän. Wysiwyg-kenttään voidaan lisätä niin tekstiä, linkkejä kuin kuviakin. Jos asiakkaalle annetaan ohjeet wysiwyg-kentän käyttöön, on hyvin mahdollista, että jossain kohtaa sisältöä syö-

tetään väärin. Advanced custom fields poistaa tämän ongelman. Advanced custom fieldsiin voidaan määritellä eri kentät eri sisältötyypeille ja kutsua osioita juuri niille varatuissa kohdissa sivupohjassa. Tällöin minimoidaan mahdolliset syötön aikana tapahtuvat inhimillisetvirheet. Kun kentät on jäsennelty intuitiivisesti, asiakkaan käyttökokemus parantuu merkittävästi.

7.3.1 Advanced Custom Fields kentän määrittely

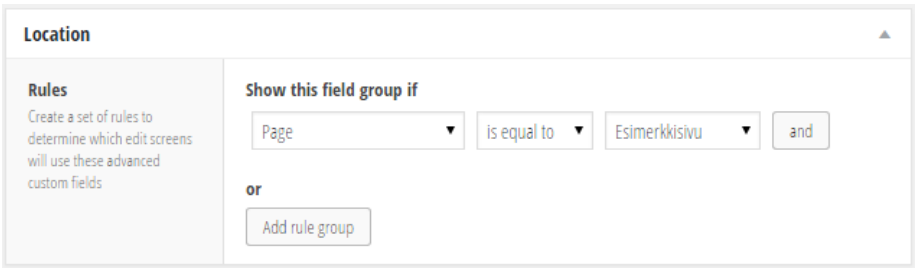
Advanced Custom Fields -kentät määritellään admin-näkymässä menemällä valikossa kohtaan Custom fields ja Add new.



Kuvio 14. Advanced Custom Fields admin-valikossa.

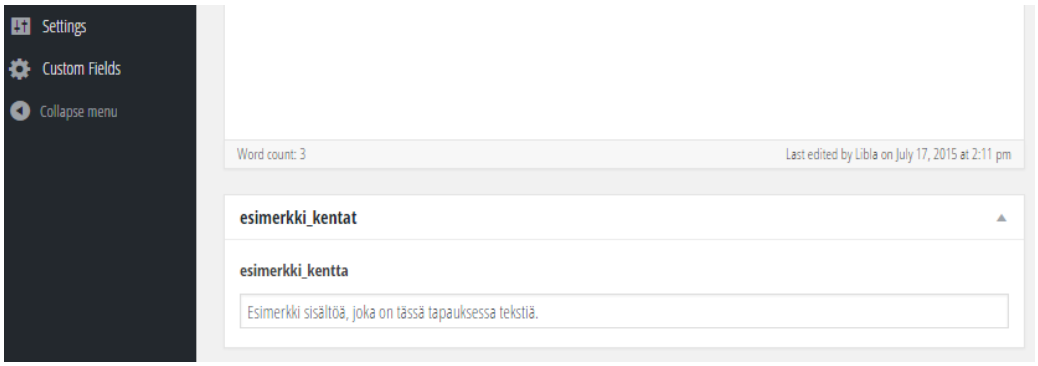
Uuden custom fieldsin luontinäkymässä määritellään kentälle nimi, joka näkyy käyttäjälle ja nimi jolla kenttää kutsutaan sivupohjissa. Kenttiä on hyvin monenlaisia ja niille on runsaasti eri vaihtoehtoja missä muodossa ne palauttavat dataa sivupohjassa.

Yksi kriittinen asia on valita, mihin luotu custom field liitetään. Kuviossa 15 Advanced Custom Fields -kentän liittäminen tiettyyn sivuun demonstroidaan, kuinka liittää Advanced custom fields -kenttä tiettyyn sivuun (tässä tapauksessa sivuun nimeltä Esimerkkisivu).



Kuvio 15. Advanced Custom Fields kentän liittäminen sivuun.

Liitoksen tekemisen ja muutosten tallentamisen jälkeen luotu kenttä löytyy sivun admin-näkymästä kuvion 16 Advanced custom fields -kenttä sivun admin-näkymässä mukaisesti. Advanced Custom Field voidaan liittää mihinkä tahansa sisältötyyppiin tai sivuun kaikki eri vaihtoehdot löydät kohdasta *show this field group if*.



Kuvio 16. Advanced Custom Fields-kenttä sivun admin-näkymässä.

7.3.2 Advanced Custom Fields sisällön näyttäminen sivupohjassa

Advanced Custom Fields -kenttiä kutsutaan sivupohjissa PHP-tagien sisällä, kuten WordPressin sisäänrakennetuja sivupohja-funktioita. Kuviossa 16 luotua `esimerkki_kentta`:aa voidaan kutsua Esimerkkisivun sivupohjassa kuvion 17 mukaisesti funktiolla `the_field()` ja kirjoittamalla sulkumerkkien sisään parametriksi kutsutun kentän nimi. Advanced Custom Fields -kenttää voidaan kutsua myös WordPressin omalla funktiolla `get_post_meta()`. `Get_post_meta()` ottaa vastaa kaksi parametriä meta-keyn eli kentän nimen ja toisena parametrina post_ID:n joka on sivun ID, jolla `get_post_meta()`-

funktiota kutsutaan. `Get_post_meta()`-funktio on nopeampi kuin Advanced Custom Fieldsin tarjoama `get_the_field()` -funktio. Kappaleessa 7.4 kerrotaan lisää siitä, kuinka Advanced Custom Fields tallentaa dataa tietokantaan.

```
 9
10  <!-- Aloitetaan loop -->
11  <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
12
13      <!-- Otsikko -->
14      <?php the_title('<h1>', '</h1>'); ?>
15
16      <!-- Sisältö -->
17      <div class="content-wrap">
18          <!-- kutsutaan ACF-kentän sisältöä -->
19          <p><?php the_field('esimerkki_kentta'); ?></p>
20      </div>
21
22
23  <!-- Suljetaan loop -->
24  <?php endwhile; else : ?>
25      <p><?php _e( 'Ei löytynyt haluamaasi sisältöä.' ); ?></p>
26  <?php endif; ?>
27
```

Kuvio 17. Advanced Custom Fields-kentän kutsuminen sivupohjassa.

7.3.3 Advanced Custom Fields repeater field

Yksi Advanced Custom Fieldsin voimakkaimmista kentistä on repeater field. Reapeater field mahdollistaa sisällön toistorakenteen sivupohjassa. Repeater-kentään voi lisätä admin-näkymässä rajattoman määrän kenttiä ja poistaa kenttiä tarpeen mukaisesti. Jos siis tarvitaan monta samanlaista kenttää peräkkäin kannattaa käyttää repeater-kenttää. Näin voidaan välttää toistoa koodissa.

```
7
8   <?php
9       // Tarkistetaan onko repeater_field kentällä yhtään kenttää
10      if( have_rows('repeater_field') ):
11
12          while ( have_rows('repeater_field') ) : the_row(); ?>
13
14              <div class="col-xs-12 col-sm-6">
15                  <h4><?php echo get_sub_field('ala_otsikko'); ?></h4>
16                  <div><?php the_sub_field('ala_sisalto'); ?></div>
17              </div>
18
19          <?php
20          endwhile;
21
22      else :
23      // ei löytynyt yhtään riviä
24      endif;
25  ?>
26
```

Kuvio 18. Advanced Custom Fields repeater field kutsuminen sivupohjassa.

Kuviossa 18 määritetään if-lause, joka tarkistaa onko `repeater_field`:llä yhtään alakenttää. Mikäli if-lausetta ei määritellä ja kenttä on tyhjä palauttaa koodi PHP-virheen. Tämä johtuu siitä, että koodi yrittää prosessoida dataa, jota ei ole olemassa. If-lausekkeen sisällä määritellään while loop, joka muistuttaa vahvasti WordPressin omaa The loop-silmukkaa. `Have_post()`-funktion sijasta silmukkaan määritetään `have_rows([repeater_kentän_nimi])`. While-silmukan sisässä voidaan kutsua repeater-kentän sisäisiä kenttiä samaan tapaan kuin muitakin advanced custom fields-kenttiä. Tärkeää on huomata, että sisäistä kenttää kutsutaan etuliitteellä `get_sub_field([alakentän_nimi])`, vertaa normaalia `get_field([kentän_nimi])`. Myös alakenttien osalta get-etuliite palauttaa arvon ja the-etuliite tulostaa arvon. **Kuviossa 18** on käytetty molempia tapoja ja tulostettu `get_sub_field()` PHP-komennolla echo. (Advanced Custom Fields 2015a)

7.4 Advanced Custom Fields WordPressin tietokannassa

Advanced Custom Fields tallentaa datan WordPressin tietokantaan. Kentät tallentuvat asennuksen `post_meta`-tauluun. Jokaista advanced custom fields-kenttää varten luodaan tauluun kaksi riviä. Ensimmäiseen tallenetaan `meta_key`, joka on

kentän nimi etuliitteellä alaviiva. Toiseen kenttään tallentuu `meta_value`, joka on kentän arvo. Kentän arvoja voidaan hakea tietokannasta myös suorilla SQL-kyselyillä. Custom fieldsin arvoja voidaan käyttää myös esimerkiksi suodattamaan WP_Query:n palauttamia arvoja. Esimerkissä Kuvio 19 käytetään etunimeä John suodattamaan WP_Query:n palauttamia arvoja. Kappaleessa 8 käsitellään tarkemmin `$args`-arrayta ja sen määrittämistä.

```
34
35  // args
36  $args = array(
37      'post_type'     => 'ihmiset',
38      'meta_key'      => 'etunimi',
39      'meta_value'    => 'John'
40  );
41
```

Kuvio 19. Advanced Custom Fields kenttä suodattimena.

8 CUSTOM POST TYPE MÄÄRITTÄMINEN

Custom post type luodaan käyttämällä `register_post_type()`-funktiota. Funktio ottaa vastaan kaksi parametriä, `$post_type` ja `$args`. `$post_type` on luotavan sisältötyypin nimi. Se pitää syöttää kokonaan pienillä kirjaimilla ilman välilyöntejä ja sen maksimimerkkimäärä on 20. `$args` on array, joka sisältää sisältötyypin määrityksiä ja useita asetuksia WordPressissä. (WordPress Codex 2015e) *Custom post type* määritellään `functions.php`-tiedostossa, koska `functions.php` sisältö voi nopeasti paisua hyvin suureksi on hyvä erotella sisältöä eri tiedostoihin. *Custom post type* -määritykset voidaan tehdä esimerkiksi tiedostossa function-custom-post-type, joka voidaan liittää osaksi functions-php-tiedostoa PHP-komennolla include `include_once('[tiedoston-polku]');`

`Register_post_type()`-funktioon voidaan liittää `$args`-arrayn avulla useita eri argumentteja, joiden avulla saadaan määriteltyä sisältötyypille eri ominaisuuksia. Käyttökokemuksen intuitiivisuutta saadaan lisättyä lisäämällä adminnäkymään ikoni *custom post typelle*. Menu_icon argumentille voidaan osoittaa jokin WordPressin tarjoamista ikoneista, joista löytää täydellisen listan osoitteesta

https://developer.wordpress.org/resource/dashicons/ tai menu_icon argumentin voi laittaa osoittamaan omaan 16x16 pikseliä suureen kuvatiedostoon. Kuviossa 20 menu_icon osoittaa WordPressin tarjoamaan ikoniin dashicons-calendar-alt.

```php

// CUSTOM POST TYPE
add_action( 'init', 'create_post_type' );

function create_post_type() {

    $labels = array(
        'name' => __( 'ajankohtaista' ),
        'singular_name' => __( 'ajankohtainen' )
        );

    $args = array (
        'labels'        => $labels,
        'public'        => true,
        'has_archive'   => true,
        'menu_icon'     => 'dashicons-calendar-alt',
        'rewrite'       => array( 'slug' => 'ajankohtaista' ),
        'menu_position' => 5,
        'supports'      => array( 'title', 'editor', 'thumbnail', 'author', 'excerpt' ),
        'taxonomies'    => array( 'category', 'post_tag' )
        );

    register_post_type( 'ajankohtaista', $args);
}

```

Kuvio 20. Custom post type määrittäminen functions.php-tiedostossa.

Jotta *custom post typelle* voidaan luoda listausnäkymä käyttäen WordPress sivupohjajärjestelmää, tulee sille määrittää `has_archive`-argumentti arvoon `true`. Tämän jälkeen voidaan käyttää `archive.php`-tiedostoa samaan tapaan, kuin oletussisältötyypin post kanssa. Tästä enemmän luvussa 8.2 Archive.php.

8.1 Custom post type sisällön näyttäminen

Custom post typen sisältöä voidaan näyttää monella tapaa. WP_Query luokalla voidaan hakea tietokannasta dataa erilaisten argumenttien mukaisesti. WP_Query luokalle voidaan asettaa argumentiksi ’post_type’ ja osoittaa sille arvoksi merkkijonona *custom post typen* nimi. Tämän jälkeen WP_Query-luokka palauttaa tietyn sisältötyypin mukaisen datan WordPressin the loopin sisällä. Kuviossa 21 havainnollistetaan Custom post type sisällön näyttäminen WP_Queryn avulla the loopissa.

```php

$args = array('post_type' => 'ajankohtaista',
        'orderby' => 'date',
        'order' => "$order"
    );

$ajankohtaista = new WP_Query( $args );


// Käytetään custom post type WP_Querya loopissa
if($ajankohtaista->have_posts()) : while($ajankohtaista->have_posts()) : $ajankohtaista->the_post(); ?>

    <!-- Otsikko -->
    <?php the_title('<h1>', '</h1>'); ?>

    <!-- Sisältö -->
    <div class="content-wrap">
        <?php the_content(); ?>
    </div>

    <!-- Suljetaan loop -->
    <?php endwhile; else : ?>
    <p><?php _e( 'Ei löytynyt haluamaasi sisältöä.' ); ?></p>
<?php endif; ?>


```

Kuvio 21. Custom post type sisällön näyttäminen WP_Queryn avulla the loopissa.

Kun käytetään WP_Query-luokkaa näyttämään Custom post typen sisältöä voidaan *custom post type* näyttää missä tahansa sivupohjassa. Esimerkiksi voidaan listata uutiset tiettyyn osaan etusivua.

Mikäli rekisteröitäessä *custom post typea*, sille on määritetty argumentiksi has_archive, on custom post typellä käytössä hierarkinen sivupohjamalli. Custom post typen hierarkinen sivupohjamalli mahdollistaa listausivun luomisen sivupohjalla `archive.php` ja yksittäisen julkaisun `single.php`. (Damstra, Stern & Williams 2015, 136)

8.2 Archive.php

Custom post typelle voidaan luoda oma listaussivu, joka näyttää kaikki sisältötyypin mukaiset julkaisut. Tyypillistä on ettei sivulla näytetä kaikkea tietoa julkaisusta, vaan vain tarvittavat tiedot, kuten otsikko ja rajattu määrä sisältöä. Julkaisun ingressi voidaan toteuttaa `wp_trim_excerpt()`-funktiolla. `Wp_trim_excerpt()-funktio` palauttaa 55 ensimmäistä merkkiä julkaisun sisällöstä ja se kuuluu funktioihin, joita pitää käyttää the loopin sisällä (WordPress Codex 2015f).

Sivupohjatiedosto nimetään kirjoittamalla alkuun archive, väliviiva ja sen jälkeen sisältötyypin nimi. `Arcive.php`-tiedosto listaa julkaisuja. Kun halutaan linkata

julkaisuja listausnäkymässä yksittäiseen julkaisuun, voidaan linkki kutsua funktiolla `get_post_permalink()`. Funktiota `get_post_permalink()` voidaan kutsua html-ankkurielementin `href`-attribuutissa PHP-tagien sisällä. Linkistä sivusto siirtyy yksittäiseen julkaisuun, joka käyttää sivupohjaa `single.php`. Kuviossa 22 on luotu linkki h1-html-elementtiin riveillä 20—22. Rivillä 28 kutsutaan funktiota `edit_post_link()`, joka näkyy pelkästään sisään kirjautuneelle käyttäjälle. Funktio tulostaa linkin, josta käyttäjä pääsee suoraan adminnäkymään ja muuttamaan julkaisun sisältöä.

```
15              <?php if(have_posts()) : while(have_posts()) : the_post();  ?>
16
17                  <div class="col-xs-12 col-md-8">
18
19                          <!-- otsikko joka toimii linkkinä yksittäiseen postaukseen -->
20                          <a href="<?php echo get_post_permalink(); ?>" class="post-type-title">
21                              <?php the_title('<h1>', '</h1>'); ?>
22                          </a>
23
24                          <hr>
25                          <!-- sisältö -->
26                          <?php the_content(); ?>
27
28                          <div>
29                              <?php edit_post_link( 'Muuta postausta tästä', '<i>', '</i>'); ?>
30                          </div>
31
32                  </div><!-- /col -->
33
34              <?php endwhile; endif; ?>
```

Kuvio 22. Esimerkki Archive.php-tiedostosta.

8.3 Single.php

`Single.php` näyttää yksittäisen julkaisun sisällön. `Single.php` on käytössä oletus sisältötyypin mukaisille artikkeleille kuin myös *custom post type* artikkeleille. Ainoana erona *custom post type* -dataa näyttäessä tulee tiedostonimeen lisätä sisältötyypin nimi `single-[sisältötyypin_nimi].php` WordPress tietää automaattisesti mistä julkaisusta on kyse, jos sivuston sivupohjahierarkiaa on noudatettu. Tieto sisältötyypistä kulkee globaalin `$wp_query`-objektin mukana. Erillisiä styypin mukaisia kyselyjä ei siis tarvita, vaan sivulla voidaan näyttää dataa suoraan The loopin avulla.

`Single.php`-tiedostolla on joitakin erikoissivupohja-funktioita `previous_post_link()` palauttaa linkin edelliseen julkaisuun ja

`next_post_link()`-funktio palauttaa linkin seuraavaan julkaisuun. Näitä funktioita kutsutaan myös linkki-elementin href-attribuutissa, kuten tehtiin kuvi-ossa 22 `get_post_permalink()`-funktion kanssa (WordPress Codex 2015g).

9 WORDPRESS JA TIETOTURVA

Kuten todettua WordPress on äärimmäisen suosittu alusta sivustojen julkaisuun. Mitä suositumpia palveluista tulee, sitä enemmän hakkerit kiinnostuvat niiden murtamisesta. Näin on tapahtunut myös WordPressin kohdalla. WordPress toimii alustana niin harrastelijoiden kuin myös pitkänlinjan ammattilaisten julkaisemille sivustoille. Verkkosivukehittäjien kirjo on siis laaja ja sivustoilla näkee hyvin erilaisia lähestymistapoja sivustojen toteutukseen. Eritasoinen tietotaito ja eri käytänteet luovat haasteita WordPressin tietoturvan kannalta. Tässä luvussa avataan perusperiaatteita ja käytänteitä siitä kuinka toteutetaan sivusto turvallisesti ja pidetään se myös turvallisena.

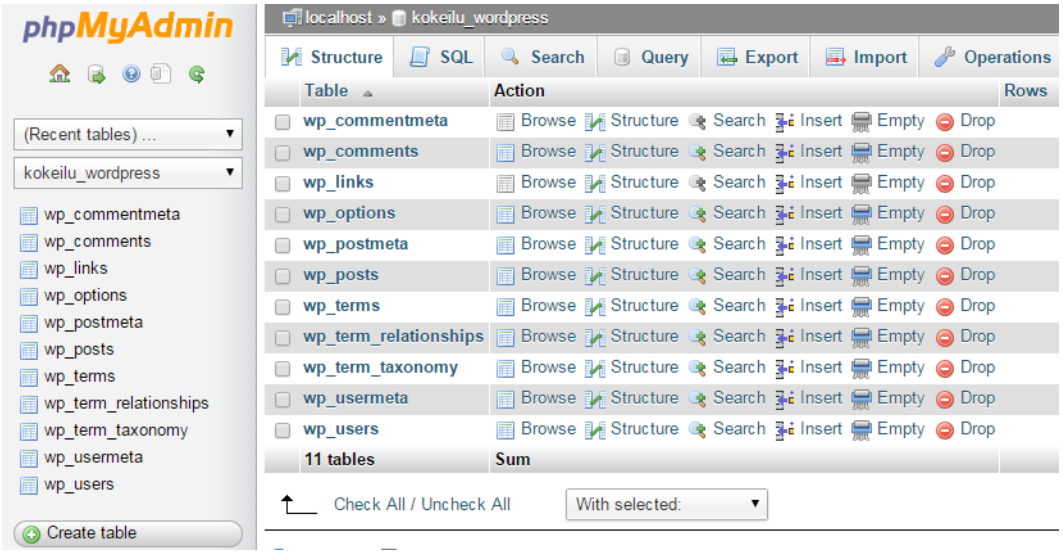
9.1 Päivitykset

WordPress-sivustoja kehittää lukematon määrä kehittäjiä päivittäin. Kaikki heistä haluavat, että sivusto on turvallinen. Mikäli kehittäjä huomaa tietoturvauhan WordPressissä on hyvin yleistä, että hän lähettää tiedon WordPress kehitystiimille ja näin tietoturva-aukko on pääsee laajempaan tietoisuuteen.

Päivityksiä tulee WordPressiin usein. Päivittäminen on onneksi hyvin nopeata ja helppoa. Kun avataan WordPressin admin-näkymä näkyvät päivitykset heti ensimmäisellä sivulla. WordPress tarjoaa myös automaattiset päivitykset. Ne vaativat serveriltä oikeuden päivittää tiedostoja WordPress-kansiossa. Oikeuksien jakaminen koko WordPress-kansioon ei ole suotavaa. Tämä avaa enemmän mahdollisuuksia hakkeroijille manipuloida tiedostoja, joihin ei ole tarkoitus koskea ilman lupaa.

9.2 Table prefix

Kun asennetaan uusi WordPress-asennus, WordPress generoi automaattisesti taulurakenteen MySQL-tietokantaan. Jokaisen taulun nimen eteen WordPress laittaa oletuksena etuliitteen `wp_` eli table prefixin. Kuviossa 23 nähdään käytännössä table prefix phpMyAdmin-näkymässä.



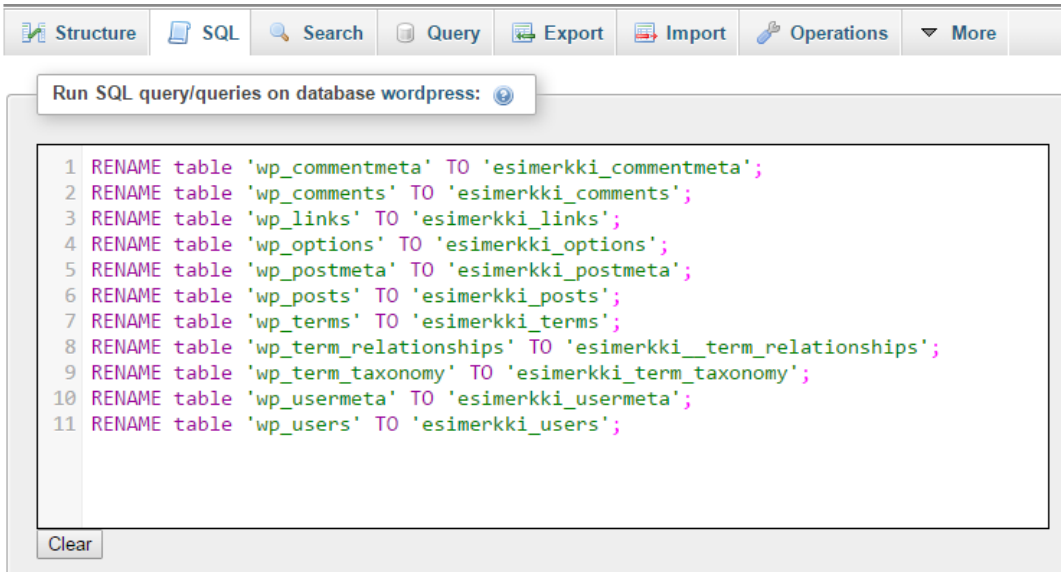
Kuvio 23. WordPress taulurakenne ja table prefix.

Mikäli oletusetuliite jätetään kantaan, on hakkereiden helpompi päästä käsiksi tauluihin. Oletusarvo voidaan muuttaa `wp_config.php`-tiedostossa vaihtamalla `$table_prefix` muuttujan arvoa. Kuviossa 24 table prefix arvon muuttaminen näytetään esimerkki, jossa arvoksi on vaihdettu `omaprefix_` oletuksen `wp_` sijasta.

```
62
63  // muutettu table_prefix
64      $table_prefix  = 'omaprefix_';
65
```

Kuvio 24. Table prefix arvon muuttaminen.

Mikäli prefix vaihdetaan vastaavalla tavalla pitää muutokset tehdä ennen asentamista. Prefix voidaan vaihtaa myös valmiille tai kehityksessä olevalle projektille. Tällöin pitää muuttaa prefix esimerkiksi käyttämällä phpMyAdminia. Kuviossa 25 ajetaan SQL-komento, jolla päivitetään taulujen prefix uuteen haluttuun muotoon. (WpBeginner 2015a)



Kuvio 25. PhpMyAdmin taulujen prefixin vaihtaminen SQL-komennolla.

Lopuksi pitää vielä tarkistaa, onko options-taulussa kenttiä, jotka käyttävät vanhaa prefixiä. Kentät voidaan hakea yksinkertaisella SQL-kyselyllä, jota havannoillistetaan kuviossa 26. Tämän jälkeen kenttien prefix voidaan vaihtaa, joko graafisessa käyttöliittymässä tai SQL-komennolla.

```
1 SELECT * FROM `esimerkki_options` WHERE `option_name` LIKE '%wp_%'
```

Kuvio 26. Options taulun prefix SQL-kysely.

Myös usermeta-taulussa voi olla kolumneja, jotka on nimetty `wp_`-etuliitteellä. Voidaan suorittaa vastaava haku kuin kuviossa 27 ja tehdä samat toimenpiteet myös tähän tauluun.

```
1 SELECT * FROM `esimerkki_usermeta` WHERE `meta_key` LIKE '%wp_%'
```

Kuvio 27. Usermeta taulun prefix SQL-kysely.

9.3 Wp-config-tiedoston siirtäminen

`Wp-config`-tiedosto pitää sisällään paljon arkaluontoista tietoa WordPress-asennuksesta, kuten salasanan ja käyttäjänimen tietokantaan. Näiden tietojen ei haluta päätyvän hakkereiden käsiin. Mikäli PHP lakkaa toimimasta sivustolla, on mahdollista, että tiedoston sisältö näkyy selkokielisellä tekstillä sivustolla.

`Wp-config`-tiedosto voidaan siirtää kansiorakenteessa yhden pykälän ylöspäin WordPress-asennuksen juuresta. Aluksi WordPress katsoo löytyykö `wp-config.php`-tiedosto WordPress-asennuksen juuresta, jos näin ei ole, se etsii sitä pykälää ylempää. Tämä kaikki tapahtuu täysin automaattisesti, eikä mitään asetuksia tarvitse muuttaa. Tämä on helppo ja nopea keino nostaa WordPress-asennuksen tietoturvan tasoa.

9.4 Käyttäjäoikeudet

WordPress tarjoaa sisäänrakennettuna mahdollisuuden jakaa käyttäjile erilaisia oikeuksia eli rooleja. Rooleja on kuutta eri tyyppiä; super admin, administrator, editor, author, contributor ja subscriber. (Damstra, Stern & Williams 2015 43/507) Taulukko 1 antaa selkeän kuvan siitä, mitä oikeuksia eri käyttäjäroolit antavat eri käyttäjille. Uusia käyttäjärooleja voidaan luoda myös itse koodissa.

CAPABILITY	SUPER ADMIN	ADMINISTRATOR	EDITOR	AUTHOR	CONTRIBUTOR	SUBSCRIBER
manage net-work	X					
manage themes	X	X				
manage plu-gins	X	X				
manage users	X	X				
manage site options	X	X				
moderate comments	X	X	X			
manage cate-gories	X	X	X			
Manage links	X	X	X			

Manage all posts	X	X	X			
Manage all pages	X	X	X			
Manage others posts	X	X	X			
Read and manage private posts	X	X	X			
Read and manage private pages	X	X	X			
Upload files	X	X	X	X		
Publish posts	X	X	X	X		
Delete own published posts	X	X	X	X		
Edit own posts	X	X	X	X	X	
Delete own unpublished posts	X	X	X	X	X	
Read	X	X	X	X	X	X

Taulukko 1. WordPress käyttäjän roolit. (WordPress Codex 2015h).

Yksi riski sivuston turvallisuudelle on jakaa liikaa oikeuksia käyttäjille. Mikäli käyttäjän ei tule tehdä muuta kuin päivittää tiettyä osiota sivustosta tulee hänelle jakaa oikeudet vain näihin toimintoihin. Tämä estää tietämätöntä käyttäjää tuhoamasta sivustolle tärkeätä materiaalia tai muuttamasta sivustolle tärkeitä asetuksia, jotka voivat hajoittaa koko sivuston. Tutkailemalla taulukkoa oikeuksista voidaan hyvin nopeasti nähdä, että editor on oiva vaihtoehto asiakkaalle. Editor ei pääse vaikuttamaan WordPress-asetuksiin vaan pääsee vaikuttamaan lähinnä sivujen ja artikkelien sisältöön. Tulee aina miettiä tarkkaan oikeuksien tarve ja jakaa niitä vain sen verran kuin tarve vaatii.

10 TULEVAISUUDEN NÄKYMÄT TEEMAKEHITYKSESSÄ

Tämä opinnäytetyö käsitteli WordPressin omaa sivupohjajärjestelmää. Tällä hetkellä näyttää siltä, että WordPress alkaa siirtymään enemmän pelkäksi back end -vaihtoehdoksi kehitettäessä verkkosovelluksia ja sivustoja. Käytännössä jatkossa ei siis enää käytetä WordPressin omia sivupohja-funktioita datan näyttämäiseen, vaan voidaan kehittää WordPress-sovelluksia irroittamalla datan käsittely erikseen datan näyttämisestä. Tällöin dataa käsitellään mallissa ja näkymässä näytetään data käyttäjälle Javascript-kirjaston esimerkiksi Angular-JS avulla.

WordPress on laajentanut toiminnallisuuttaan WP REST API -rajapinnalla. WP REST API on rajapinta, joka palauttaa sivuston dataa json-muodossa, jota voidaan kutsua eri päätepisteistä sivustolta. Eri päätepisteet palauttavat eri dataa parametrien mukaisesti. Rajapinta mahdollistaa saman datan käytön eri sovelluksissa. Kuvitellaan tilanne, jossa yritys haluaa tehdä mobiilisovelluksen, joka käyttää samaa dataa kuin yrityksen internetsivusto. Tällöin voidaan käyttää samaa dataa, niin sivustolla kuin myös mobiilisovelluksessa. Koska internetsivun ja ulkopuolisen sovelluksen data on yhteistä, molemmat päivittyvät samalla, kun WordPress back endissä tehdään muutoksia. Ajax-rajapintakäsittelyllä on etuna sen modulaarisuus. Ajax-käsittelyn ansiosta koko sivua ei tarvitse ladata kokonaan, vaan sivua voidaan ladata osissa tarpeen mukaisesti. Onko järkeä ladata koko sivua uudestaan, jos halutaan päivittää pelkkä tekstiosio ja otsikko sovelluksesta?

Kokonainen WordPress-teema voidaan kehittää käyttäen WP REST APIa. Itse teemakehitys voidaan tehdä esimerkiksi AngularJS nimisellä javascript-ohjelmistokehyksellä. AngularJS perustuu malli-näkymä-käsittelijä-arkkitehtuuriin, jossa eroitetaan datan käsittelyn malliin ja datan näyttäminen näkymään. Vastaava arkkitehtuuri mahdollistaa logiikkakäsittelyn siirtämisen sivupohjista mallin sisään ja näin sivupohjista saadaan selkeämpiä kehittää. Opinnäytetyön jälkeen tein harjoittelun nimissä itselleni sivuston www.liblauniverse.com,

joka on toteutettu WP REST API -rajapinnalla ja sivupohjat on toteutettu AngularJS-ohjelmistokehyksellä.

Kokonaisuutena opinnäytetyö onnistui hyvin. Asiakas oli tyytyväinen projektin lopputulokseen. Valmis projekti löytyy osoitteesta http://spty.fi/. Opinnäytetyön tekijä pääsi syventämään omia taitojaan WordPress-kehittämisessä. Samalla syntyi opinnäytetyö, jonka avulla uusi WordPress-kehittäjä pääsee alkuun WordPress-maailmaan tutustumisessa.

LÄHTEET

Advanced Custom Fields 2015a Viitattu 1.9.2015
http://www.advancedcustomfields.com/resources/code-example

Bootstrap 2015a Css. Viitattu 1.6.2015
http://getbootstrap.com/css/

Damstra D., Stern H. & Williams B. Professional 2015 WordPress Design and Development 3. painos. Indiana. John Wiley & Sons, Inc.

Developer WordPress 2016a wp_enqueue_style() Viitattu 20.1.2016
https://developer.wordpress.org/reference/functions/wp_enqueue_style/

Developer WordPress 2016b add_action () Viitattu 20.1.2016
https://developer.wordpress.org/reference/functions/add_action/

Spurlock J. 2013 Bootstrap. 1. painos. Sebastopol. O'Reilly Media, Inc.

WordPress Codex 2015a Post Types. Viitattu 7.6.2015
https://codex.wordpress.org/Post_Types

WordPress Codex 2015b Function Reference / Uploading Files Viitattu 10.8.2015
http://codex.wordpress.org/Function_Reference/ Uploading_Files

WordPress Codex 2015c Function Reference / wp max upload size Viitattu 10.8.2015
http://codex.wordpress.org/Function_Reference/wp_max_upload_size

WordPress Codex 2015d The Loop. Viitattu 16.7.2015
https://codex.wordpress.org/The_Loop

WordPress Codex 2015e Function Reference / Register Post Type. Viitattu 2.6.2015
https://codex.wordpress.org/Function_Reference/register_post_type

WordPress Codex 2015f Function Reference / wp trim excerpt. Viitattu 18.07.2015
https://codex.wordpress.org/Function_Reference/wp_trim_excerpt

WordPress Codex 2015g Theme Development / Single Post single.php. Viitattu 29 03.08.2015
https://codex.wordpress.org/Theme_Development#Single_Post_.28single.php.

WordPress Codex 2015h Viitattu 7.6.2015
https://codex.wordpress.org/Roles_and_Capabilities

WordPress Plugins 2015 Advaced Custom Fields Viitattu 16.7.2015
https://wordpress.org/plugins/advanced-custom-fields/

WpBeginner 2015a http://www.wpbeginner.com/wp-tutorials/how-to-change-the-wordpress-database-prefix-to-improve-security/ Viitattu 07.08.2015