Jasu Koskivirta

WORDPRESS-TEEMAN KEHITTÄMINEN

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2022

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Jasu Koskivirta
Työn nimi	WordPress-teeman kehittäminen
Toimeksiantaja	Mainostoimisto Groteski Oy
Vuosi	2022
Sivut	41 sivua, liitteitä 2 sivua
Työn ohjaaja(t)	Arto Väätäinen

TIIVISTELMÄ

Tämän opinnäytetyön aiheena on uuden WordPress-teeman kehittäminen toimeksiantajan web-kehityksen tueksi. Päätavoitteena on, että teeman uudistuksen avulla siirrytään WordPress-sisällönhallintajärjestelmän klassisen sisältöeditorin käytöstä uuteen Gutenberg-editoriin. Gutenberg eroaa vanhasta editorista siten, että se perustuu lohkoihin, jotka toimivat erilaisina sisältöelementteinä. Sivuston muokkausnäkymän on tarkoitus vastata lopullista sivuston ulkoasua, eikä täten erillistä esikatseluikkunaa tarvita. Toimeksiantajana työssä on Mikkelissä sijaitseva Mainostoimisto Groteski Oy, joka toteuttaa perinteisten mainostoimiston palveluiden lisäksi näyttäviä ja visuaalisia digitaalisia ratkaisuja.

Opinnäytetyön aluksi esitellään sisällönhallintajärjestelmän perusteet, minkä jälkeen teoriaosuus keskittyy WordPressin ominaisuuksiin kuten lisäosiin. Työssä käsitellään myös keskeisimmät teeman kehittämiseen käytetyt webtekniikat, kuten PHP, SCSS ja jQuery. Tarkoituksena on esitellä kehitysprosessi sellaisille henkilöille, jotka haluavat perehtyä WordPress-sivujen kehityksen ytimeen eli teemaan, joka määrittää verkkosivuston rakenteen, toiminnallisuuden sekä ulkoasun.

Ennen varsinaista kehitystyön aloittamista uutta teemaa suunniteltiin ja määriteltiin yhdessä toimeksiantajan kanssa. Gutenberg-editorin yhteiskäyttöä Advanced Custom Fields -lisäosan kanssa testattiin lokaalissa kehitysympäristössä, minkä jälkeen teeman rakentaminen ja siihen luotavien lohkojen kehitys alkoi. Lohkon rakentaminen aina rekisteröinnistä ohjelmointiin saakka esitellään työssä kattavasti vaihe vaiheelta, minkä jälkeen niiden ulkoasu ja responsiivisuus testattiin opinnäytetyötä varten rakennetulla demosivulla.

Tuloksena on aiempaa helppokäyttöisempi ylläpitonäkymä asiakkaille verkkosivustojen sisällönsyöttöä varten sekä monipuolinen ja selkeä kustomoitava pohja toimeksiantajan web-kehityksen käyttöön. Uusi teema toimii ensimmäisenä julkaistuna versiona, minkä jälkeen sen jatkokehitys alkaa kehittäjiltä saatujen kokemusten sekä asiakkaiden palautteiden perusteella.

Asiasanat: sisällönhallintajärjestelmä, web-ohjelmointi, WordPress, tietojenkäsittely

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Business Administration
Author (authors)	Jasu Koskivirta
Thesis title	Development of a WordPress theme
Commissioned by	Mainostoimisto Groteski Oy
Time	February 2022
Pages	41 pages, 2 pages of appendices
Supervisor	Arto Väätäinen

ABSTRACT

The purpose of this thesis was to develop a theme using the content management system WordPress. The main objective was that the former classic content editor would be replaced with the new Gutenberg editor based on block type content. Blocks are content elements that users can add to their website at edit screen to create different content layouts. The site edit screen view will match the final visual look of the website so there is no need for a separate preview window with the Gutenberg editor.

The theoretical part of this thesis introduced the basics of a content management system before focusing on the update history and features of WordPress. This thesis also explained technologies used for developing the theme, such as PHP, SCSS and jQuery. The purpose was to present the development process to people who wanted to become more familiar with WordPress development and themes. The theme is like core for a WordPress website, because it defines the structure, functionality and visual layout.

Before the actual development, the new theme was designed and defined together with the client company. The usage of the Gutenberg editor with the Advanced Custom Fields plugin was tested before developing blocks and other elements for the new theme. The theme was developed in a local development environment and its responsiveness was tested on a demo website compiled for this thesis.

The main result of this thesis was an easy-to-use edit screen for customers to update content on their websites. The new theme also works as customizable and a versatile base for web developers at Mainostoimisto Groteski Oy. It will be the first published version and its further development will continue based on the information gained from the developers and customer feedback.

Keywords: content management system, web programming, WordPress, data processing

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on kehittää Mainostoimisto Groteski Oy:lle uusi täysin kustomoitu teema käyttäen WordPress-sisällönhallintajärjestelmää. Nykyinen teema on tarkoitus korvata uudella kevään 2022 aikana. Toimin yrityksessä työharjoittelijana ennen toimeksiantoa, joten pääsin kehittämään verkkosivuja nykyisillä tekniikoilla ja sain täten hyvät lähtövalmiudet opinnäytetyön tekoon.

Opinnäytetyön aihe tukee Groteskin tavoitetta kasvaa yrityksenä sekä pysyä aallon harjalla modernin web-kehittämisen parissa. Uusitun teeman avulla yritys voi tarjota asiakkailleen entistäkin laadukkaampia ja helppokäyttöisempiä ratkaisuja esimerkiksi verkkosivustojen sisältöjen päivittämiseen. Työ itsessään sisältää teorian ja käytettyjen tekniikoiden esittelyn lisäksi erilaisten ohjelmointikielien soveltamista teeman kehittämiseen.

WordPressin ylläpitonäkymä on perinteisesti täysin erinäköinen verrattuna itse sivustoon, joten sisällönhallinta voi tuntua loppukäyttäjästä monimutkaiselta ja sekavalta. Lähtökohtaisesti uusi teema ja hallintapaneeli ratkaisevat tämän ongelman mahdollistamalla sen, ettei loppukäyttäjän tarvitse selata erillisiä valikoita ja esikatselutiloja päivittääkseen sisältöjä verkkosivuilleen. Sivuston muokkausnäkymän on siis tarkoitus vastata varsinaisen sivuston ulkoasua, eikä erillistä esikatseluikkunaa täten tarvita.

Opinnäytetyön toisessa luvussa on esitelty WordPress-sisällönhallintajärjestelmän perusteita ja tekniikoita, kuten oletuseditorin korvaavaa Gutenberg-editoria. Luvussa kolme yritysesittelyn jälkeen on kuvattu lokaalin kehitysympäristön käyttöönotto sekä uuden lohkoeditorin testaus ennen varsinaisen kehitystyön aloitusta. Luku neljä sisältää toteutuksen vaihe vaiheelta sekä lopputuloksen testauksen demosivun avulla. Työn päättävä viides luku summaa kehittämistehtävän toteutuksen. Käyn luvussa läpi esimerkiksi sen, miten valmista toteutusta voisi jatkokehittää tulevaisuudessa.

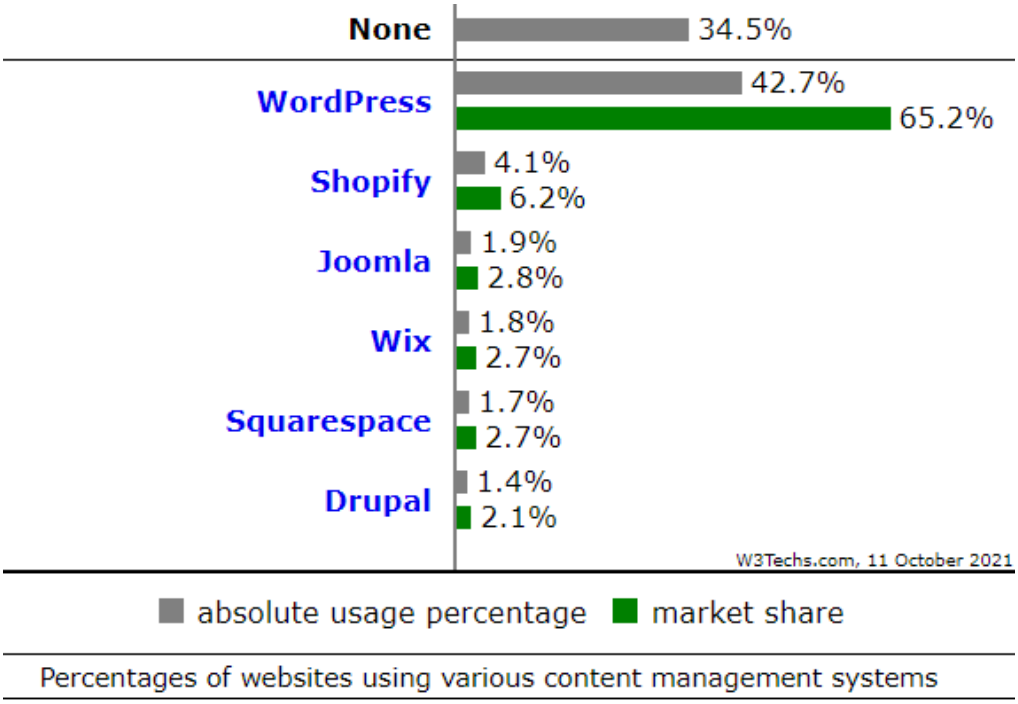
2 TYÖSSÄ KÄYTETYT TEKNIIKAT

Tässä luvussa esitellään opinnäytetyön kannalta keskeisiä tekniikoita kuten sisällönhallintajärjestelmän määritelmä, WordPressin tärkeimpiä ominaisuuksia sekä erilaisia web-tekniikoita, joiden avulla kehitystyö on toteutettu. Luvussa 3 kerrotaan tarkemmin, mitä teknisiä vaatimuksia toimeksiantaja määritti uudelle teemalle.

2.1 Sisällönhallintajärjestelmä (CMS)

Normaalisti verkkosivusto on kirjoitettu HTML-, CSS- ja JavaScript-ohjelmointikielillä. Jotta käyttäjä voi luoda uuden verkkosivuston, tarvitsee hänen hallita em. kielet. Sisällönhallintajärjestelmä (engl. *Content Management System, CMS*) on alusta, joka ratkaisee tämän ongelman. Sen avulla käyttäjä voi luoda uusia sivustoja sekä hallinnoida ja päivittää niitä ilman koodausosaamista. (Wpbeginner 2021.)

Nykyisin markkinoilta löytyy lukusia erilaisia ja eritasoisia sisällönhallintajärjestelmiä. Suosituimpia ovat mm. *WordPress*, *Drupal* sekä *Joomla*, jotka kaikki ovat ilmaisia avoimeen lähdekoodiin perustuvia järjestelmiä. Kaikille kolmelle edellä mainitulle järjestelmälle on olemassa laaja yhteisö, joka kehittää uusia ominaisuuksia sekä tukee käyttäjiä ongelmatilanteissa. (Wpbeginner 2021.)



Kuva 1. Sisällönhallintajärjestelmien markkinaosuuksia lokakuussa 2021 (W3techs 2021)

Johtavan CMS-tietolähteen *W3techs*:in (2021) mukaan WordPress on kuitenkin sisällönhallintajärjestelmistä käytetyin sekä myös markkinaosuudeltaan selvästi suurin. Lähes 43 prosenttia kaikista maailman verkkosivuista on toteutettu WordPressin avulla, ja näin ollen se on markkinaosuudeltaan kymmenkertainen verrattuna lähimpiin kilpailijoihin (kuva 1).

2.2 WordPress

WordPress on avoimen lähdekoodin ilmainen ohjelmisto, jolla käyttäjät ympäri maailmaa voivat helposti luoda verkkosivustoja erilaisiin käyttötarkoituksiin. Se sai alkunsa vuonna 2003, kun perustajat Mike Little ja Matt Mullenweg havaitsivat tarpeen hyvin rakennetulle ja visuaalisesti tyylikkäälle henkilökohtaiselle julkaisujärjestelmälle. Nykyisin WordPress perustuu PHP-ohjelmointikieleen sekä MySQL-tietokantaan ja on lisensoitu GPLv2-lisenssillä. (WordPress 2019.)

2.2.1 Versiot

On hyvä huomioida, että WordPressistä on olemassa kaksi erillistä versiota: WordPress.com ja WordPress.org. Näistä ensiksi mainittu on WordPressin omalla palvelimella käytettävä versio, kun taas jälkimmäinen on omalle palvelimelle asennettava versio. (Hakukonemestarit 2020.)

WordPress.com mahdollistaa alkeellisten kotisivujen teon, joten sitä käytetään esimerkiksi oman henkilökohtaisen blogisivuston ylläpitoon. Sen peruskäyttö on ilmaista eikä käyttäjän tarvitse itse huolehtia päivitysten asentamisesta. Haittapuolena voidaan pitää ilmaisversion suppeutta. Lisäosien asentaminen tai verkkokaupan pyörittäminen vaatii maksullisia paketteja, eikä valmiita teemoja ole kuin rajallinen määrä. Käyttäjän on myös rekisteröidyttävä ja hyväksyttävä käyttöehdot, eikä sivun koodiin ole mahdollista tehdä mitään muutoksia. Hakukonemestarit (2020) muistuttaakin, että jos kotisivut tehdään WordPress.comin palvelulla, tekijä ei omista blogia tai sen sisältöjä. (Hakukonemestarit 2020.)

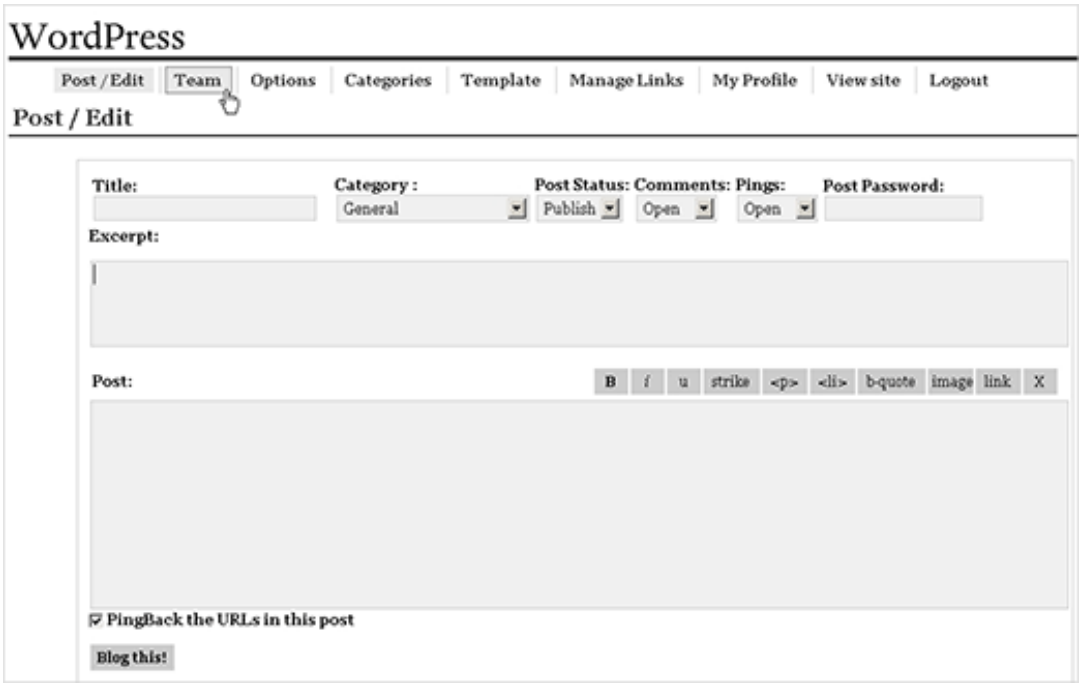
Jos tarkoituksena on tehdä juuri sellaiset verkkosivut kuin haluaa, niin valinta versioiden välillä kohdistuu WordPress.orgiin. Sen käyttöönotto onnistuu täysin ilmaiseksi lataamalla ja asentamalla uusimman version omalle

palvelimelle. Käyttäjä saa täyden vapauden esimerkiksi lisäosien asentamiseen tai koodin muokkaamiseen. Käyttäjä omistaa sivunsa ja kaiken sisällön eikä rekisteröitymistä tarvita. Täytyy kuitenkin muistaa, että varmuuskopiointi sekä päivitysten asentaminen täytyy hoitaa itse. (Hakukonemestarit 2020.)

Opinnäytetyössä on käytetty ladattavaa WordPress.org-palvelua. Tämä johtuu siitä syystä, että se mahdollistaa toimeksiantajalle räätälöidä asiakasprojektit ilman rajoitteita, kun taas WordPress.com on soveltuvampi yksityishenkilöiden tarpeisiin. Ilman mahdollisuutta vaikuttaa sivujen koodiin on käytännössä raskas ammattimainen käyttö mahdotonta WordPressillä. Kun jatkossa tässä työssä puhutaan WordPressistä, tarkoitetaan nimenomaan asennettavaa WordPress.org-versiota.

2.2.2 Päivityshistoria

WordPressin edeltäjänä toimi *b2/cafelog*-niminen julkaisujärjestelmä, joka toimi lähinnä blogien kirjoitusta varten. Vuonna 2003 kaksi kyseisen julkaisujärjestelmän käyttäjää Matt Mullenweg ja Mike Little päättivät rakentaa uuden alustan b2/cafelogin päälle. Tästä syntyi WordPress, jonka ensimmäinen virallinen versio julkaistiin 27.5.2003 ja se sai hyvän vastaanoton yhteisöltä. Ensimmäinen versio (kuva 2) sisälsi mm. uusitun admin-käyttöliittymän sekä uusia templaatteja eli sivupohjia. (The History of WordPress 2021.)



Kuva 2. WordPressin ensimmäinen versio vuodelta 2003 (The History Of WordPress 2021)

Toukokuussa 2004 julkaistiin WordPressin ensimmäinen suuri päivitys (versio 1.2), joka toi mukanaan lisäosat (engl. *plugins*), joiden avulla käyttäjät voivat lisätä verkkosivuihinsa erilaisia ominaisuuksia ja toimintoja. Lisäosat ts. laajennukset ovat edelleen merkittävässä roolissa WordPress-kehityksessä, ja niitä löytyy nykypäivänä reilusti yli 50 000 kappaletta. (Miller 2021.)

Seuraava tärkeä ominaisuus eli ylläpitäjälle tarkoitetun käyttöliittymän saapuminen osaksi WordPressiä tapahtui joulukuussa 2005 (versio 2.0). Päivityksen myötä ylläpitäjä pystyi lisäämään kategorioita ja tunnisteita kirjoituksiin poistumatta editorista. Kyseistä päivitystä pidetään yhtenä suurimmista parannuksista WordPressin historian aikana. (Miller 2021.)

Vuosina 2010–2013 WordPress sai useita isoja versiopäivityksiä. Ne toivat mukanaan valtavan määrän uusia työkaluja sivuston ulkoasun räätälöintiä varten, kuten täysin kustomoitavat taustakuvat, otsikot sekä valikkorakenteet. Isoimpina muutoksina voidaan pitää vuoden 2012 versiopäivitystä 3.4, jolloin mukaan tuotiin teemat (engl. *themes*) sekä vuoden 2013 päivitystä, jolloin automaattiset päivitykset tulivat osaksi WordPressiä (versio 3.7). (Miller 2021.)

Viimeisimpänä suurena päivityksenä pidetään loppuvuoden 2018 versiopäivitystä 5.0. Tällöin vuodesta 2003 asti ollut oletuseditori korvattiin Gutenberg-

nimisellä “lohkoeditorilla”, joka mullisti tavan rakentaa verkkosivuja WordPressillä. Gutenbergin avulla käyttäjä voi jakaa sisältöjä, kuten tekstiä, kuvia sekä listoja omiin lohkoihinsa, joita voi tarvittaessa muokata haluamallaan tavalla. (Miller 2021.)

2.2.3 WordPress-teemat ja -lisäosat

Valmiiksi rakennettu ulkoasupohja eli teema vastaa sivuston ominaisuuksista, rakenteesta sekä visuaalisesta ilmeestä. WordPress sisältää sekä ilmaisia teemoja että maksullisia premium-teemoja, joita käyttäjät voivat muokata toiveidensa mukaan. Helpoimmillaan teema toimii siten, että käyttäjän ei tarvitse kuin lisätä omat sisältönsä valmiiksi suunnitelluille sivupohjille saadakseen verkkosivuston kasaan. (WP-kotisivut 2021a.)

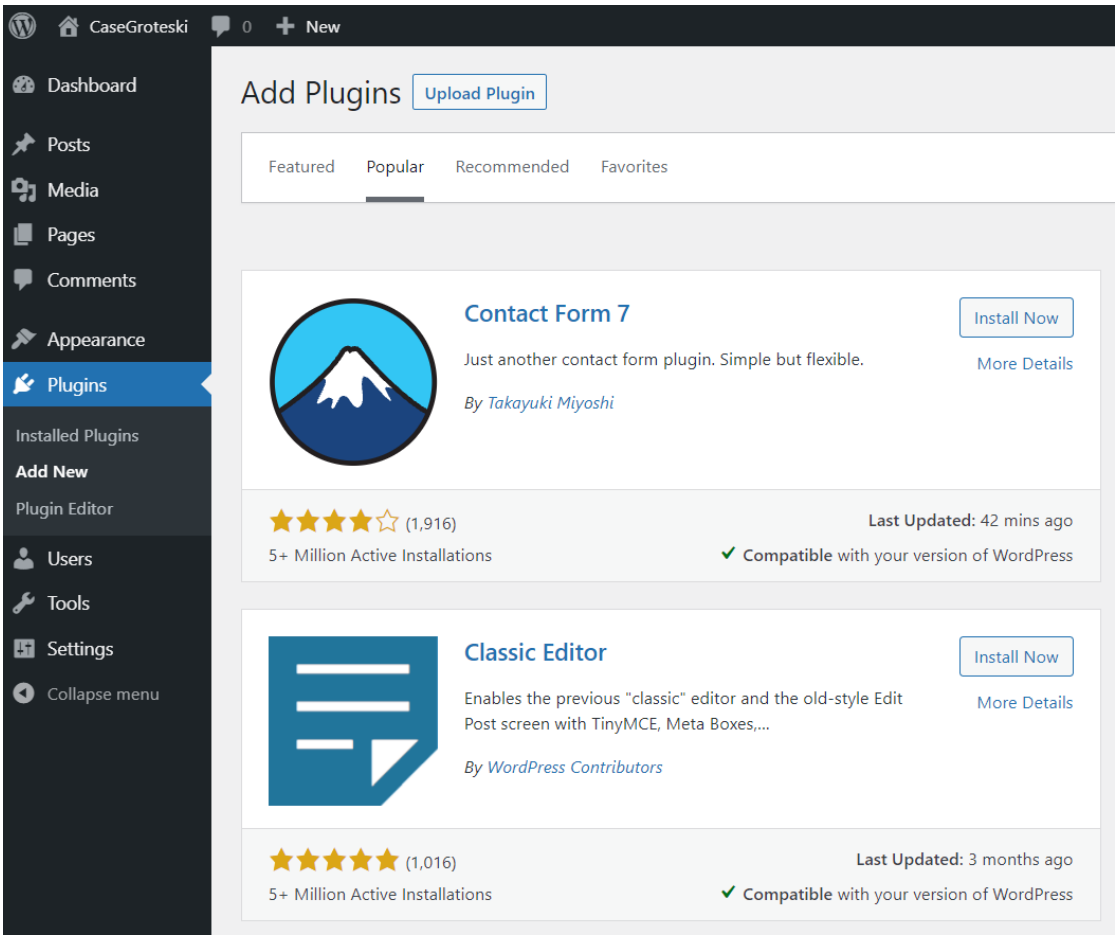
Tuhansien erilaisten teemojen valikoimasta WordPress käyttäjän voi olla hankalaa löytää sopiva vaihtoehto sivustolleen. On kuitenkin hyvä muistaa, että vaikka teema olisi suunniteltu johonkin tiettyyn toimialaan, niin sillä voi kuitenkin tehdä verkkosivut kaikenlaiseen tarkoitukseen. WP-kotisivut (2021a) on listannut käyttäjien avuksi kriteerejä, joita on hyvä ottaa huomioon sopivan teeman etsinnässä:

- Käyttäjien arvostelut ja kommentit
- Teeman suosio ja lataus-/myyntimäärä
- Päivityshistoria valmistajan tai kehittäjän toimesta
- Teeman ominaisuudet ja ulkonäkö
- Responsiivisuus eli mobiiliystävällisyys
- Teeman nopeus
- Hakukoneoptimointi

Teeman tekniseen rakenteeseen sekä yksittäisiin tiedostoihin tutustutaan enemmän luvussa *4.1 Teeman rakenne ja käyttöönotto*.

WordPressin lisäosa eli laajennus mahdollistaa uusien ominaisuuksien ja toimintojen käyttöönoton verkkosivustolle. Laajennusten avulla voidaan lisätä toiminnallisuuksia ilman koodausosaamista, joten siksi niistä on tullut erittäin suosittuja käyttäjien keskuudessa. WordPressiin on saatavilla tuhansia ilmaisia sekä maksullisia lisäosia, joiden laatu vaihtelee merkittävästi. Käyttäjän kannattaa siis selvittää ennen lisäosan asennusta sen luotettavuus sekä yhteensopivuus ja toimivuus oman WordPress-verkkosivuston kanssa. Lisäosien

asennus, päivitys ja käyttöönotto tapahtuu WordPressin hallintapaneelin kautta (kuva 3). (WP-kotisivut 2021b.)



Kuva 3. Lisäosien asennus WordPressin hallintapaneelissa

Käyttäjän kannattaa asentaa sivustolleen vain tarvittavat lisäosat. Tämä siksi, että turhat asennukset voivat hidastaa sivuston toimintaa. Myös vanhojen tarpeettomien lisäosien poistaminen on syytä tehdä, sillä vanhentuessaan ne voivat aiheuttaa tietoturvariskin ja avata hakkereille pääsyn sivustolle. (WP-kotisivut 2021b.)

Alla oleva listaus sisältää poimintoja WP-kotisivujen (2021b) suosittelemista lisäosista. Lista on jaettu kategorioihin ja suluissa on mainittu nimeltä esimerkkejä kyseisen kategorian lisäosista:

1. Turvallisuus (*Wordfence, iThemes Security*)
2. Varmuuskopiointi (*All in one WP migration*)
3. SEO Hakukoneoptimointi (*Yoast SEO, All in one seo pack*)
4. Cache välimuisti (*WP Fastest cache*)
5. Verkkokauppa (*WooCommerce*)

6. Google Analytics kävijäseuranta (*GA Google Analytics*)
7. Kuvien optimointi (*Shortpixel*)
8. Page builder, sivunrakentaja (*Elementor, Divi*)
9. Yhteydenotto- ja kyselylomake (*Contact form 7*)
10. Kalenteri (*Setmore*)

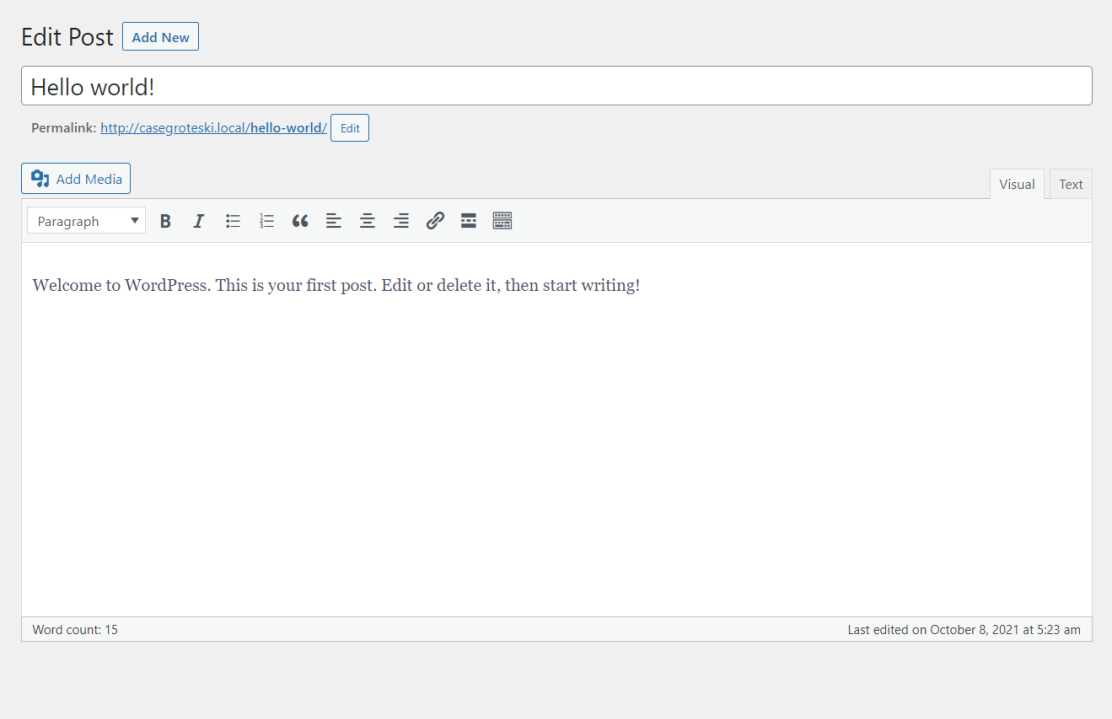
Tämän opinnäytetyön lopputuotokseen sisältyy muutamia lisäosia, jotka toimeksiantaja on erikseen määritellyt. Nämä lisäosat ovat asennettu osaksi uutta teemaa, mutta niiden käytöstä ei tässä työssä ole erikseen kerrottu. Asennetut lisäosat on lueteltu luvussa *3.3 Määrittely ja suunnittelu*.

2.2.4 Gutenberg-editori

Gutenberg-niminen WordPressin oma sisältöeditori on keskeisessä roolissa tässä opinnäytetyössä. Sillä on tarkoitus korvata vanha WordPressin sisältöeditori (tunnetaan myös nimellä ”klassinen editori”), joka on ollut tähän asti käytössä toimeksiantajan web-kehityksessä. Tämä luku perustuu kokonaisuudessaan Jalosen (2020) artikkeliin Gutenbergin ominaisuuksista ja käyttökokemuksista.

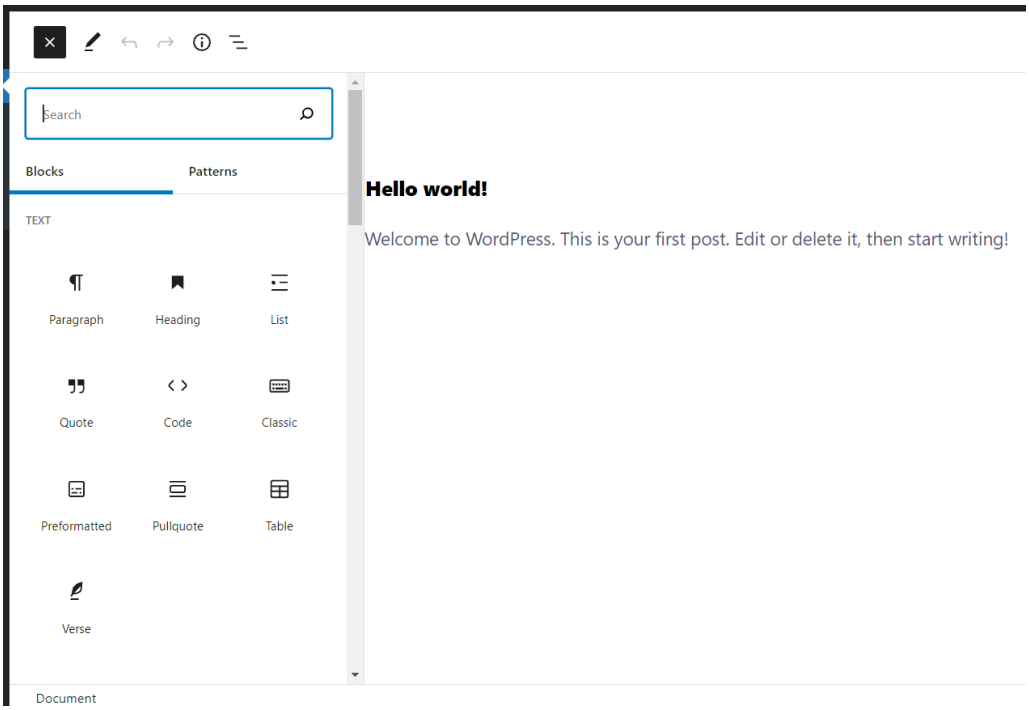
Gutenberg-editori tuli laajempaan käyttöön, kun joulukuussa 2018 WordPress-versio 5 julkaistiin. Editori on nimetty Johannes Gutenbergin mukaan, joka tunnettiin kirjapainon keksijänä. Gutenberg perustuu lohkoihin (engl. *blocks*), jotka toimivat sisältöelementteinä kuten tekstinä, otsikoina tai kuvina. Tästä seuraa myös yleisesti käytetty nimitys ”lohkoeditori” puhuttaessa Gutenberg-editorista.

Klassiseen editoriin (engl. *classic editor*) nähden merkittävin ero Gutenbergissä on se, miten sisältö rakennetaan uudelle sivulle, koska klassinen editori sisältää ainoastaan yhden tyhjän sisältöalueen (kuva 4). Klassinen editori muistuttaakin tekstinkäsittelyohjelmista tuttua muokkausmahdollisuutta, sisältäen esimerkiksi tekstin lihavoinnin, tasauksen tai kuvien lisäämisen tekstin joukkoon. Nämä ominaisuudet ovat käytössä myös Gutenberg-editorissa.



Kuva 4. WordPressin klassinen sisältöeditori

Gutenberg sisältää oletuksena esimerkiksi kappale-, otsikko-, kuva- ja painikkeet-lohkot, joiden käytön voi aloittaa heti editorin avautuessa (kuva 5). Uusia lohkoja voi rakentaa halutuista elementeistä WordPressin käyttöliittymässä, jolloin yksittäistä lohkoa voi käyttää uudelleen sivuston eri paikoissa. Myös HTML-ankkureiden tai CSS-luokkien lisäys onnistuu esimerkiksi ulkoasutyylien määrittämiseksi, mutta tämä kuitenkin vaatii koodiin perehtyneisyyttä.



Kuva 5. Oletuslohkojen lisäys Gutenberg-editorissa

Vaikka Gutenberg kuuluu nykyään WordPressin ydinversioon automaattisesti, voi käyttäjä halutessaan palata klassisen editorin käyttöön. Tämä onnistuu asentamalla *Classic Editor* -nimisen lisäosan, jolloin Gutenberg ohitetaan kokonaan. Täytyy kuitenkin huomata, että kyseiselle lisäosalle on luvattu tuki varmuudella vain vuoden 2021 loppuun asti.

2.3 Web-tekniikat

Seuraavissa luvuissa esitellään kehitystyön kannalta keskeisiä web-tekniikoita. On hyvä huomioida, että tässä opinnäytetyössä ei opeteta ohjelmoinnin perusteita tai käydä läpi esimerkiksi perinteistä HTML-merkintäkieltä (engl. *Hypertext Markup Language*) tai CSS-tyylikieltä (engl. *Cascading Style Sheets*). Tässä luvussa keskitytään siis ainoastaan WordPressin sekä kustomoidun teeman ja hallintapaneelin toteutukseen olennaisesti liittyviin tekniikoihin.

2.3.1 PHP

PHP (engl. *Pre-Processor Hypertext*) on yleisesti käytetty ohjelmointikieli erilaisten dynaamisten verkkosivustojen ja -sovellusten kehittämiseen. Se on myös saanut vuosien varrella paljon suosiota palvelinpuolen skriptauskielenä. PHP on helppokäyttöinen ja tehokas kieli, joka toimii useissa eri käyttöjärjestelmissä. (Carr. & Gray 2018, 1–2.)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>
<body>

<h1>PHP syntax demo page</h1>

<?php
//PHP code goes here between PHP tags
echo "Hello World!";
?>

</body>
</html>
```

Kuva 6. Esimerkki PHP:n syntaksista osana HTML-koodia

PHP:n syntaksi on todella tärkeä hallita, jotta palvelin tietää, mistä PHP-koodin lukeminen ja parsiminen on tarkoitus aloittaa. Käytettäessä PHP:n aloitus- sekä lopetustunnisteita (engl. *tags*) voi koodin kirjoittaa käytännössä mihin tahansa kohtaan tiedostossa (kuva 6). Tämä tarkoittaa sitä, että PHP-koodi voidaan tunnisteiden avulla syöttää toimivasti esimerkiksi keskelle HTML-verkkosivuston koodia. (Carr. & Gray 2018, 1–2.)

Koska WordPress perustuu PHP-koodiin, toimii se opinnäytetyössä toteutuksen osalta isossa roolissa. Teema itsessään sekä sen sisältämät osakokonaisuudet kuten Gutenberg-editorille luodut kustomoidut lohkot ovat koodattu PHP:lla. Lopputulosta kuvakaappauksineen käsitellään kokonaisuudessaan luvussa *4. Toteutus ja testaus*.

2.3.2 SASS, SCSS

SASS (lyhenne sanoista *Syntically Awesome Style Sheets*) on CSS-tyylikielen laajennus. Sitä voidaan kutsua myös CSS-esiprosessoriksi. SASS sisältää CSS-syntaksista puuttuvia ominaisuuksia, kuten muuttujat ja sisäkkäiset säännöt (engl. *nested rules, nesting*). SASS:n avulla voidaan vähentää toistoa, joka puolestaan säästää aikaa ja lyhentää lopullista koodia. (Sass 2021.)

SASS:n muuttujien avulla voidaan tallentaa tietoa, kuten värikoodeja tai kirjasintyyppejä eli fontteja (kuva 7). Tehtäessä esimerkiksi yrityksen brändin mukaista väritystä verkkosivuille, voi kehittäjä pitää muuttujien avulla värikoodin ympäri sivustoa samana sekä muuttaa sitä tarpeen tullen yhdellä muutoksella koodissa. (Sass 2021.)

Kuva 7. SASS:n muuttujat vertailussa CSS-syntaksiin (Sass Basics 2021)

Sisäkkäisillä säännöillä voidaan noudattaa HTML-koodin visuaalista hierarkiaa, joka perinteisestä CSS-koodista on puuttunut. SASS tukee myös tiedostojen osittamista. Tämä tarkoittaa sitä, että SASS-tiedostot voidaan jakaa osiin (engl. *partials*), jotka sisältävät vain pienen osuuden CSS-koodin kokonaisuudesta. Tämä tapa helpottaa koodin lukemista sekä ylläpitämistä. (Sass 2021.)

Kun opinnäytetyössä mainitaan kirjainyhdistelmä SCSS, tarkoitetaan nimenomaan SASS:ia. Näiden kahden ero on siinä, että SCSS on SASS:in syntaksi, joka rakentuu olemassa olevan CSS-syntaksin päälle. Tämä tarkoittaa sitä, että käytössä ovat puolipisteet ja hakasulkeet, kuten CSS-syntaksissa.

2.3.3 jQuery

jQuery on avoimen lähdekoodin JavaScript-kirjasto eli kokoelma uudelleenkäytettävää JavaScript-koodia. Se on luotu helpottamaan kehittäjien arkea, ettei samoja ongelmia tarvitsisi ratkaista uudestaan ja uudestaan. Sisällyttämällä jQuery-kirjaston osaksi mitä tahansa projektia, saa käyttöönsä esimerkiksi yleishyödyllisiä funktioita, jotka on kirjoitettu selkeään ja helppolukuiseen muotoon. (MacLees 2014.)

jQueryn helppokäyttöisyyttä tukee se, että se toimii yhdessä HTML-merkintäkielen elementtien sekä CSS-valitsimien (engl. *selectors*) kanssa. Taustalla oleva yhteisö sekä todella laaja lisäosien kirjasto takaavat sen, että kehittäjän ei tarvitse rakentaa kaikkea tarvitsemaansa alusta asti. Myös jQueryn käyttöönotto on tehty helpoksi. Vaihtoehtoina on joko ladata kirjastosta oma kopio, tai sisällyttää se osaksi projektia käyttämällä ns. sisällönjakeluverkkoa (engl. *CDN, Content Delivery Network*). (MacLees 2014.)

Kuvassa 8 on havainnollistettu, kuinka jQueryn syntaksi eroaa JavaScriptistä. Esimerkkinä kuvassa on toiminto, jossa painiketta painamalla piilotetaan HTML-elementti ruudulta. Vaikka kyseessä on hyvin yksinkertainen ja lyhyt toiminto voidaan kuitenkin huomata, että koodin määrä putoaa merkittävästi käyttämällä jQuerya.

```
1   //JavaScript
2   var button = document.getElementById("button");
3   var image = document.getElementById("img");
4   button.addEventListener("click", () => {
5       image.style.display = "none";
6   });
7
8   //jQuery
9   $("#button").click(() => {
10      $("#img").hide();
11  });
12
```

Kuva 8. JavaScriptin ja jQueryn eroja havainnollistava esimerkki

MacLees:n (2014) mukaan on tärkeä huomioida, että jQuery ei ole erillinen ohjelmointikieli. Se sisältää samat säännöt ja syntaksin kuin JavaScript, mutta tekee kirjoittamisesta helpompaa. jQueryn virallinen slogan kuuluukin, että: *"write less, do more"*. Tämä tiivistää hyvin jQueryn idean, koska sen avulla vain muutama rivi koodia voi tehdä monipuolisia asioita, jotka olisivat työläitä kirjoittaa pelkällä JavaScriptillä.

2.3.4 Responsiivisuus

Responsiivisuudella tarkoitetaan sitä, että verkkosivusto suunnitellaan mukautumaan käyttäjän päätelaitteen mukaisesti. Sivusto skaalautuu erilaisille näytöille sopivaksi sekä painikkeet ja valikot toimivat myös esimerkiksi mobiililaitteilla. Verkkosivustojen responsiivisuuden toteutukseen käytetään nykyisin erilaisia ohjelmistokehyksiä (engl. *framework*), joista suosituin kehys eli *Bootstrap* toimi hyvänä vaihtoehtona teeman responsiivisuuden toteuttamiseksi.

Bootstrap on avoimen lähdekoodin ilmainen CSS-ohjelmistokehys, jonka avulla voidaan toteuttaa mm. sivuston responsiivisuutta. Bootstrapin käyttö perustuu erilaisiin komponentteihin, joita käytetään HTML-rakenteessa. Tällaisia komponentteja ovat esimerkiksi *container*, *column* ja *row*, joiden avulla voidaan rakentaa täysin responsiivinen ruudukkomainen (engl. *grid*) rakenne sivustolle. Bootstrapin käyttöönotto tapahtuu lisäämällä CDN-linkki HTML-tiedostoon tai vaihtoehtoisesti lataamalla tiedostot osaksi projektia. (Getbootstrap 2021.)

Toinen suosittu tapa toteuttaa sivuston responsiivisuutta on käyttää CSS:n omaa kaksiulotteista ruudukkomaista rakennetta, *CSS Grid Layoutia*. Se eroaa Bootstrapista siten, että toteutus tehdään pääasiassa CSS-koodissa, kun taas Bootstrapia käytetään HTML-elementtien class-arvoina. Lopulliseen toteutukseen otimme CSS Grid Layoutin käyttöön Bootstrapin sijasta. Tästä kerrotaan lisää seuraavassa luvussa.

3 TOIMEKSIANTO

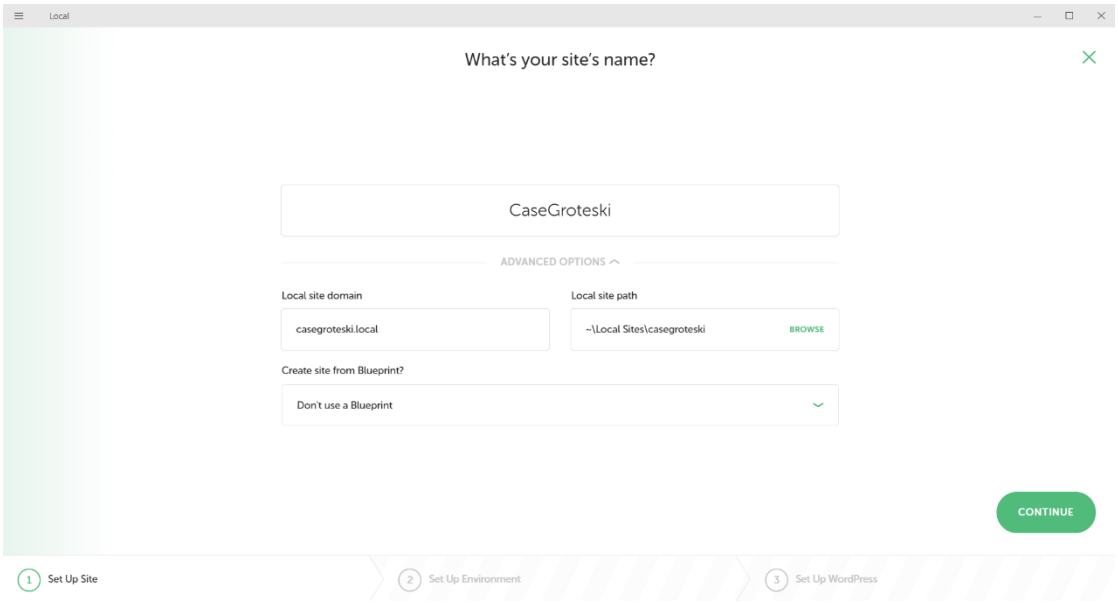
Mainostoimisto Groteski Oy on vuonna 2012 perustettu mikkeliläinen mainostoimisto. Groteskin palveluvalikoimaan kuuluvat brändi- ja kampanjasuunnittelu, visuaalinen suunnittelu sekä digitaaliset ratkaisut. Nykyisin Groteski työllistää 11 ammattilaista, joiden joukossa on visuaalisia suunnittelijoita sekä web-kehittäjiä. Yritys suunnittelee ja toteuttaa projekteja, jotka räätälöidään täysin asiakkaan toiveiden ja tarpeiden mukaan.

WordPress-sivustojen luonnista Groteskilla on vuosien kokemus, jonka aikana tuotantoprosessi on kehittynyt valtavasti. Heidän ajatusmalliinsa kuuluu se, että alusta asti hyvin ja laadukkaasti rakennettu sivusto pysyy helppona ylläpitää. Tämä takaa myös sen, että sivuja on mahdollista laajentaa ja skaalata asiakasyrityksen liiketoiminnan mukaan. Siksi sivujen jatkokehitys ja ylläpitopalvelut kuuluvat olennaisena osana Groteskin palveluihin. Myös erilaiset konsultointipalvelut liittyen esimerkiksi hakukoneoptimointiin, sivujen analytiikkaan tai saavutettavuuteen ovat osana palvelukokonaisuutta.

3.1 Lokaalin kehitysympäristön käyttöönotto

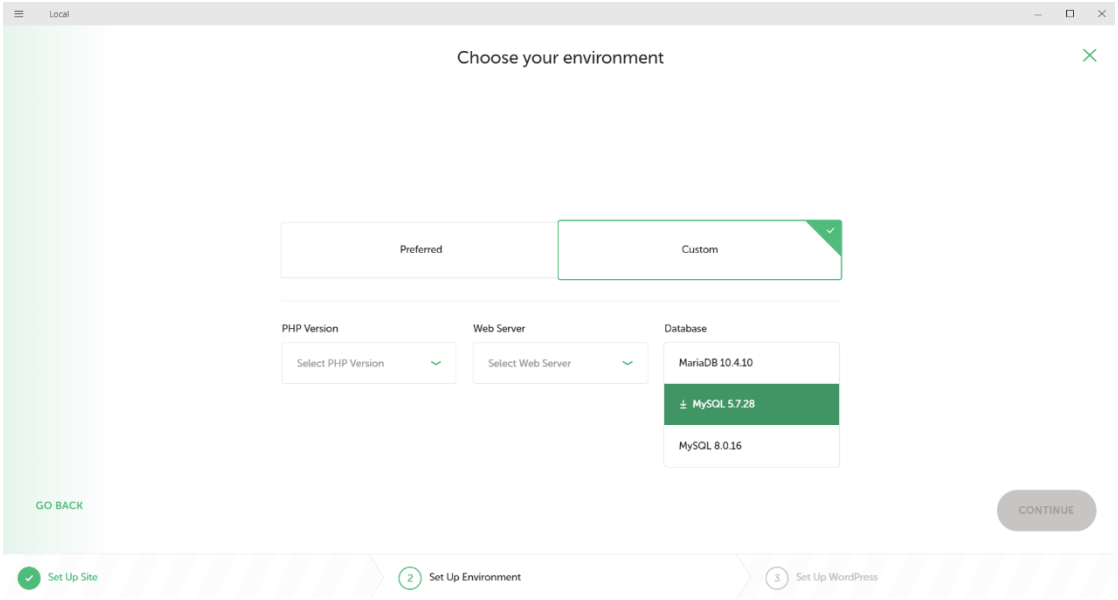
Työn toteutusta varten tarvitsee omalle koneelle asentaa paikallisesti WordPress. Tähän tarkoitukseen paras vaihtoehto on *Local*-niminen työkalu, joka on ilmaiseksi ladattavissa osoitteesta https://localwp.com/. Kyseisen työkalun avulla käyttäjä voi perustaa useita lokaaleja WordPress-sivustoja omalle koneelleen erilaisilla asetuksilla.

Localin lataamisen ja asentamisen jälkeen oman sivuston perustaminen onnistuu vaivattomasti. Etusivun ”Add Local Site” -painikkeella siirrytään kolme kohtaa sisältävään määritykseen. Aluksi sivusto nimetään halutulla tavalla, josta saadaan myös verkkotunnus (engl. *domain*) omalle sivustolle (kuva 9).



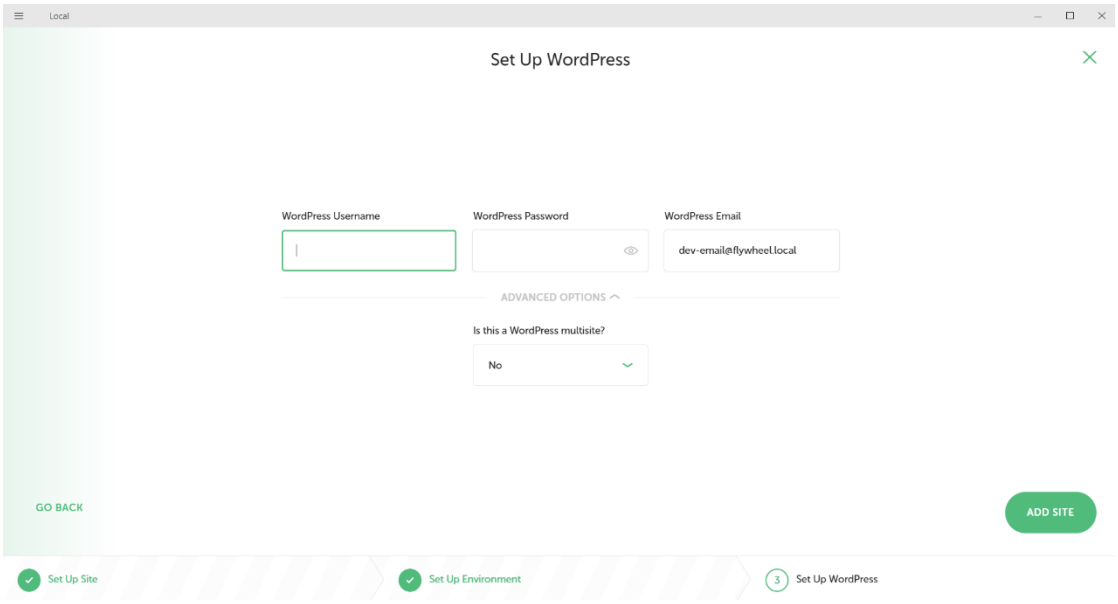
Kuva 9. Lokaalin sivuston perustaminen

Tämän jälkeen sivustolle annetaan oletusasetukset. Tässä vaiheessa käyttäjä voi valita suositellut asetukset (engl. *preferred*) tai muuttaa halutessaan esimerkiksi PHP:n versiota tai käytettävää tietokantaa (kuva 10).



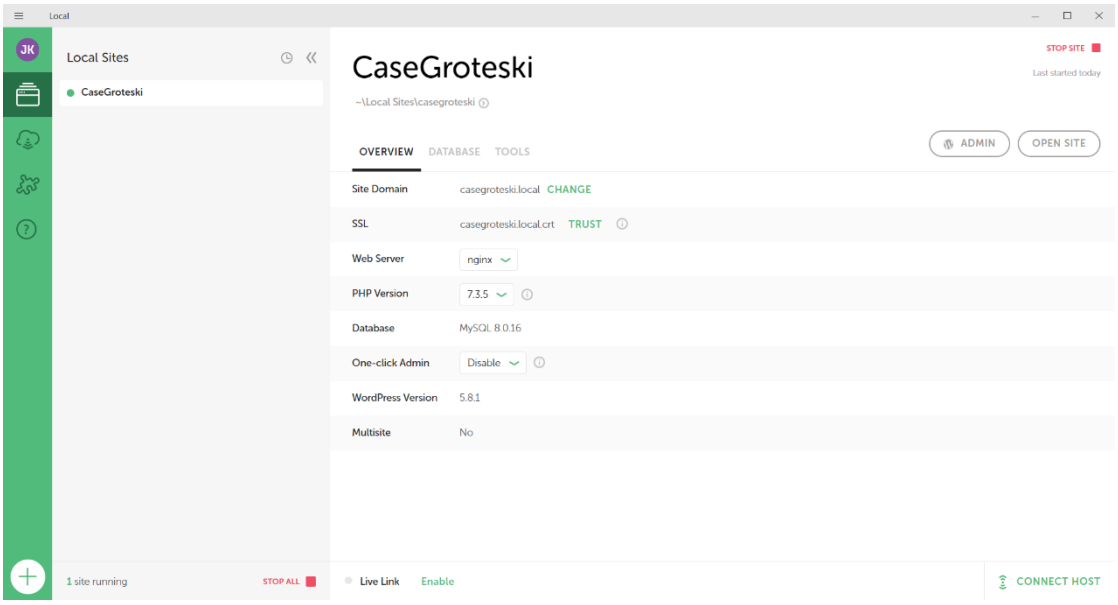
Kuva 10. Lokaalin sivuston asetusten valinta

Kolmantena ja viimeisenä kohtana luodaan käyttäjä tulevalle WordPress-sivustolle (kuva 11).



Kuva 11. Käyttäjätilin luominen lokaalille sivustolle

Tämän jälkeen Local luo sivuston halutuilla asetuksilla ja lisää sen *Local Sites* -listaan. Kuvassa 12 näkyvä Localin etusivu sisältää myös painikkeet mm. uuden sivuston käynnistämiselle, sammuttamiselle sekä ylläpitäjän (engl. *admin*) näkymään kirjautumiselle. Sivuston yleisnäkymässä huomautetaan esimerkiksi vanhentuneesta WordPressin versiosta sekä tiettyjä muutoksia on mahdollista tehdä myös jälkikäteen.



Kuva 12. Local-kehitysympäristön etusivunäkymä

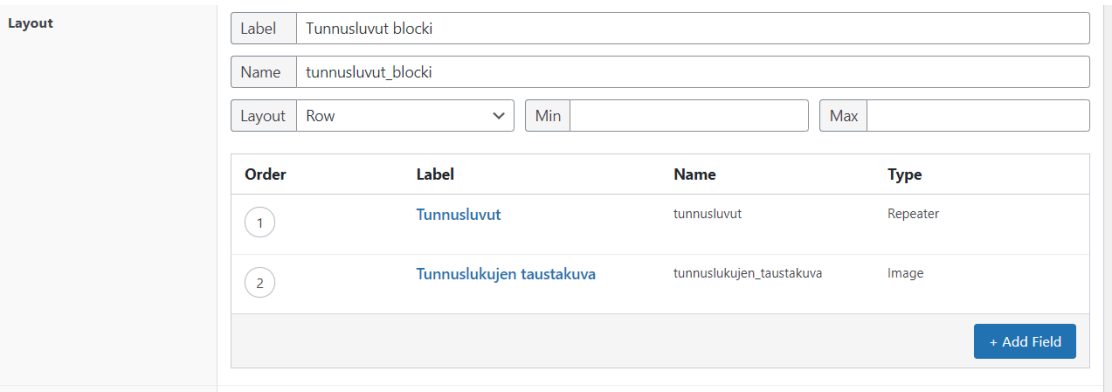
Kun ensimmäisen kerran käynnistää ja avaa uuden WordPress-sivustonsa, voi huomata, että se sisältää ainoastaan yhden ”Hello world!” -nimisen artikkelin (engl. *post*) sekä ”Sample page” -nimisen sivun. Sivustossa on oletuksena

asennettu kolme WordPressin omaa teemaa: *Twenty Nineteen*, *Twenty Twenty* sekä *Twenty Twenty-One*, joista viimeisin on käytössä. Mitään lisäosia ei kuulu oletuksena uuden WordPressin asennukseen.

3.2 Gutenberg ja Advanced Custom Fields

Ennen varsinaisen kehitystyön käynnistymistä tutustuin Gutenberg-editoriin ja sen eroavaisuuksiin klassiseen editoriin verrattaessa. Asensin tässä vaiheessa sivustolle *Classic Editor* -nimisen lisäosan, jotta pääsisin tarvittaessa tekemään vertailua ja testausta. Toimeksiantajan kehotuksesta päätin rakentaa muutamia yksinkertaisia lohkoja käyttäen Advanced Custom Fields PRO (ACF) -lisäosaa yhdessä Gutenberg-editorin kanssa. Kyseisen lisäosan avulla pystyy rakentamaan kustomoituja elementtejä esimerkiksi teksteille, kuville ja valintakentille. Näitä elementtejä on mahdollista asettaa sivustolleen mihin tahansa sisältötyypistä riippumatta. Tämän jälkeen loppukäyttäjän on helppo toteuttaa sisältöjen lisäykset ja muutokset WordPressin editorissa ilman minkäänlaista koodausosaamista.

ACF:ää käytetään WordPressin hallintapaneelista, jonka kautta siirrytään luomaan kenttäryhmiä (engl. *field groups*). Kenttäryhmän sisälle on mahdollista rakentaa useita yksittäisiä elementtejä, joihin voi asettaa erilaisia syöttö- ja lomaketietoja. Rakensin tällä tapaa elementin (kuva 13 ja kuva 14), jossa käyttäjältä kysytään taustakuva sekä kolme kappaletta tunnuslukuja (sis. teksti ja numero). Kyseinen elementti kuvastaa sitä, että yrityksen sivustolla olisi näkyvillä erilaisia tunnuslukuja kuten työntekijöiden määrä tai liikevaihdon suuruus.

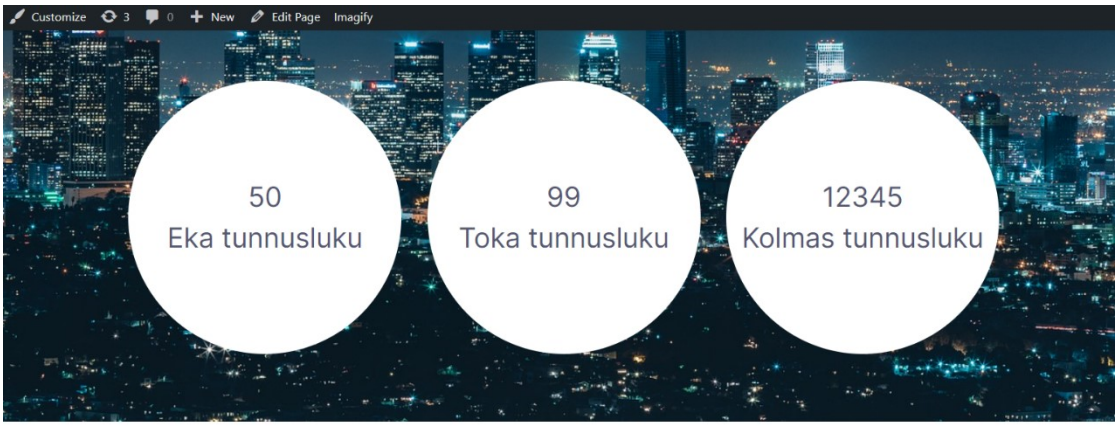


Kuva 13. Elementin luonti ACF-lisäosalla

```php
<?php elseif ( get_row_layout() == 'tunnusluvut_blocki' ) : ?>

<div class="tunnusluvut-block-wrapper" style="background-image:
url('<?php echo get_sub_field('tunnuslukujen_taustakuva')['url']; ?>')">

    <?php if( have_rows('tunnusluvut') ): ?>

        <div class="tunnusluvut-block row">

        <?php while( have_rows('tunnusluvut') ) : the_row(); ?>

            <div class="single-tunnusluku">
                <div class="luvut">
                    <span class="count"><?php echo get_sub_field('eka_luku') ?></span></br>
                    <span class="text"><?php echo get_sub_field('eka_teksti') ?></span>
                </div>
            </div>
            <div class="single-tunnusluku">
                <div class="luvut">
                    <span class="count"><?php echo get_sub_field('toka_luku') ?></span></br>
                    <span class="text"><?php echo get_sub_field('toka_teksti') ?></span>
                </div>
            </div>
            <div class="single-tunnusluku">
                <div class="luvut">
                    <span class="count"><?php echo get_sub_field('kolmas_luku') ?></span></br>
                    <span class="text"><?php echo get_sub_field('kolmas_teksti') ?></span>
                </div>
            </div>

        <?php endwhile; ?>

        </div>

    <?php endif; ?>
```

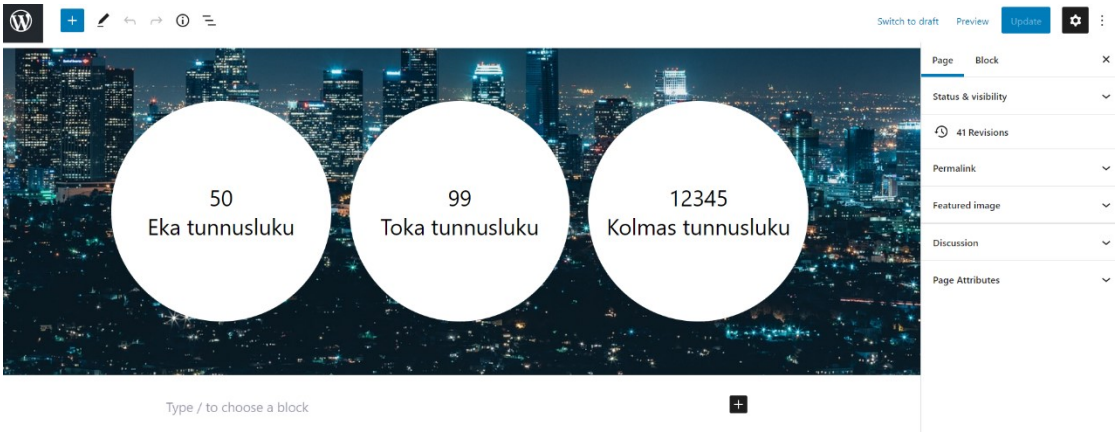
Kuva 14. ACF-elementin rakentaminen PHP-koodilla

Lopuksi tein yksinkertaisen ulkoasumuotoilun kyseiselle elementille CSS-koodin avulla. Kuvassa 15 on näkymä elementistä selaimessa silloin, kun käyttäjä on antanut syöttötietoja.



Kuva 15. Valmis ACF-elementti selaimessa

Käytössäni olleessa teemassa (*Air-light*) CSS oli jaettu kahteen erilliseen tiedostoon, joista toista käytettiin nimenomaan Gutenberg-editorin näkymään. Tein muokkauksia myös tähän CSS-tiedostoon, jotta saisin editointinäkymän vastaamaan selaimessa ollutta ulkoasua (kuva 16).



Kuva 16. ACF-elementti Gutenberg-editorissa

Tämän harjoituksen myötä huomaa heti Gutenbergin vahvuudet klassiseen editoriin verrattuna. Loppukäyttäjän on todella helppo nähdä muutokset samalla, kun hän syöttää uutta sisältöä tai lisää kokonaan uusia elementtejä sivustolleen. Erillistä esikatseluikkunaa ei käytännössä tarvita vaan muutokset näkyvät reaaliajassa jo editorinäkymässä.

3.3 Määrittely ja suunnittelu

Projektin alussa pidimme muutamia yhteisiä workshop-tyylisiä palavereita toimeksiantajan kanssa, joissa ideoimme tulevan teeman sisältöä. Päätimme käyttää pohjana jatkokehitykseen tarkoitettua starter-teemaa nimeltä *Underscores*, josta Groteskilla oli jo aiempaa kokemusta. Underscoresista on riisuttu ulkoasu käytännössä kokonaan pois, mutta se sisältää kuitenkin WordPressin toiminnallisuuden kannalta kaikki tärkeät elementit. Palavereissa luotiin lista asioista, jotka on tarkoitus huomioida uuden teeman toteutuksessa:

- **SCSS**
  - Käytetään SCSS-syntaksia toteutuksessa.
  - Kootaan SCSS-tiedostot yhdeksi CSS-tiedostoksi *Gulp*-nimisen työkalun avulla.

- **Responsiivisuus**
  - Aiemmin käytössä on ollut Bootstrap, mutta totesimme, että CSS Grid Layoutilla saadaan haluttu lopputulos aikaiseksi joustavammin. Myös tulevien lohkojen HTML-rakenne saadaan pidettyä mahdollisimman yksinkertaisena ilman Bootstrapia.
  - Pidetään syntaksi johdonmukaisena tulevissa elementeissä.
  - Käytetään muuttujia esimerkiksi ”breakpointeissa” responsiivisuutta ajatellen.

- **Saavutettavuus**
  - Tehdään saavutettavuutta varten helpot ja toistuvat toimenpiteet valmiiksi teemaan (esim. asetetaan HTML <img> tageihin alt-attribuutti valmiiksi).

- **Lisäosat**
  - Asennetaan teemaan valmiiksi lisäosat, jotka ovat käytössä jokaisessa tulevassa sivustossa: *Advanced Custom Fields Pro, Imagify, Contact Form 7, Easy WP SMTP ja Yoast SEO*.

- **Gutenberg-lohkot**
  - Luodaan valmiita lohkoja, joita tarvitaan käytännössä jokaisessa asiakasprojektissa.
  - Tarvittavia lohkoja ovat ainakin tekstilohko, kahden ja kolmen kolumnin lohkot, teksti- ja kuvalohko, karusellilohko sekä lohkot logoille ja henkilöille.
  - Lohkoihin tehdään PHP- ja HTML-rakenne sekä tarvittavat CSS-muotoilut, kuten marginaalit ja responsiivisuus.

Tässä vaiheessa projektille luotiin myös versionhallintaa varten *Github*-nimiseen palveluun ns. ohjelmavarasto (engl. *repository*). Versionhallinta on hyvä tapa varsinkin silloin, kun projektin parissa työskentelee useita eri henkilöitä. Sen avulla voidaan esimerkiksi palata projektin aiempaan versioon, jos nykyiseen versioon on päässyt ei-toivottuja muutoksia. Versionhallinta toimii myös varmuuskopiona projektille sekä hyvänä tallennuspaikkana koodeille jatkokehitystä ajatellen. Tämän opinnäytetyön ajan Github repository toimi linkkinä toimeksiantajan web-kehittäjän kanssa, joka aika ajoin tarkasteli koodiani ja pyysi lisäyksiä sekä antoi korjausehdotuksia tulevaa valmista teemaa ajatellen.

4 TOTEUTUS JA TESTAUS

Tässä opinnäytetyön luvussa on kuvattu teeman tekninen toteutus. Projekti aloitettiin yhteistyössä toimeksiantajan kanssa, minkä jälkeen siirryin itsenäisesti kehittämään mm. teemaan tulevia lohkoja sekä niiden ulkoasua ja responsiivisuutta. Totesimme myös yhdessä, että työn laajuudesta johtuen on

vaikea rajata valmista lopputulosta selkeästi. Tästä syystä otimme tavoitteeksi saada teeman sellaiseen vaiheeseen opinnäytetyön puitteissa, että sillä voidaan aloittaa web-kehitys tulevissa asiakasprojekteissa.

4.1 Teeman rakenne ja käyttöönotto

Tässä luvussa tutustutaan teeman rakenteeseen ohjelmointiteknisesti sekä tarkastellaan teemassa olevien tiedostojen toimintoja ja merkitystä kokonaisuuden kannalta. Tarkoitus ei ole esitellä jokaista teemaan kuuluvaa tiedostoa, vaan käydä läpi ne, jotka ovat keskeisimmässä roolissa uuden teeman kehityksessä. Kuvassa 17 näkyy pohjaksi valitun Underscores-teeman kansiorakenne tiedostoineen.

Nimi	Muokkauspäivä	Tyyppi	Koko
inc	20.1.2022 10.11	Tiedostokansio	
js	20.1.2022 10.11	Tiedostokansio	
languages	20.1.2022 10.11	Tiedostokansio	
template-parts	20.1.2022 10.11	Tiedostokansio	
.eslintrc	20.1.2022 10.11	ESLINTRC-tiedosto	1 kt
.stylelintrc	20.1.2022 10.11	JSON-tiedosto	1 kt
404	20.1.2022 10.11	PHP-tiedosto	2 kt
archive	20.1.2022 10.11	PHP-tiedosto	2 kt
comments	20.1.2022 10.11	PHP-tiedosto	3 kt
composer	20.1.2022 10.11	JSON-tiedosto	2 kt
footer	20.1.2022 10.11	PHP-tiedosto	1 kt
functions	20.1.2022 10.11	PHP-tiedosto	5 kt
header	20.1.2022 10.11	PHP-tiedosto	2 kt
index	20.1.2022 10.11	PHP-tiedosto	2 kt
LICENSE	20.1.2022 10.11	Tiedosto	18 kt
package	20.1.2022 10.11	JSON-tiedosto	2 kt
page	20.1.2022 10.11	PHP-tiedosto	1 kt
phpcs.xml.dist	20.1.2022 10.11	DIST-tiedosto	4 kt
README	20.1.2022 10.11	MD-tiedosto	5 kt
readme	20.1.2022 10.11	Tekstitiedosto	2 kt
screenshot	20.1.2022 10.11	PNG-tiedosto	1 kt
search	20.1.2022 10.11	PHP-tiedosto	2 kt
sidebar	20.1.2022 10.11	PHP-tiedosto	1 kt
single	20.1.2022 10.11	PHP-tiedosto	1 kt
style	20.1.2022 10.11	CSS-tiedosto	17 kt
style-rtl	20.1.2022 10.11	CSS-tiedosto	17 kt

Kuva 17. Underscores-teeman rakenne oletuksena

Teeman yksi pakollisista tiedostoista on index.php, jota käytetään oletuksena verkkosivuston etusivun muodostuksessa. Index.php-tiedostoon haetaan WordPressin omalla *get*-funktiolla header.php sekä footer.php. Näistä header.php toimii sivun otsakkeena, joka sisältää HTML-sivun head-osion (kuten html, head ja meta-tag:it sisältöineen). WordPressissä tähän osioon ei lisätä suoria viittauksia esimerkiksi CSS-tyylitiedostoihin. Viittaukset hoidetaan wp_head-funktiolla ennen viimeistä head-tagia. Samalla tavalla index.php-tiedostoon liitetään myös footer.php eli sivuston alatunniste. Se kutsutaan wp_footer-funktiolla, joka lisätään viimeiseksi ennen footer-elementin sulkemista.

Teeman toiminnallisuus sijoitetaan functions.php-tiedostoon. Sen sisällä voidaan erilaisten funktioiden avulla määrittää mm. sivuston asetuksia sekä linkittää CSS- ja JavaScript-tiedostoja tai ulkoisia kirjastoja, kuten esimerkiksi fontteja tai aiemmin työssä esitelty jQuery-kirjasto.

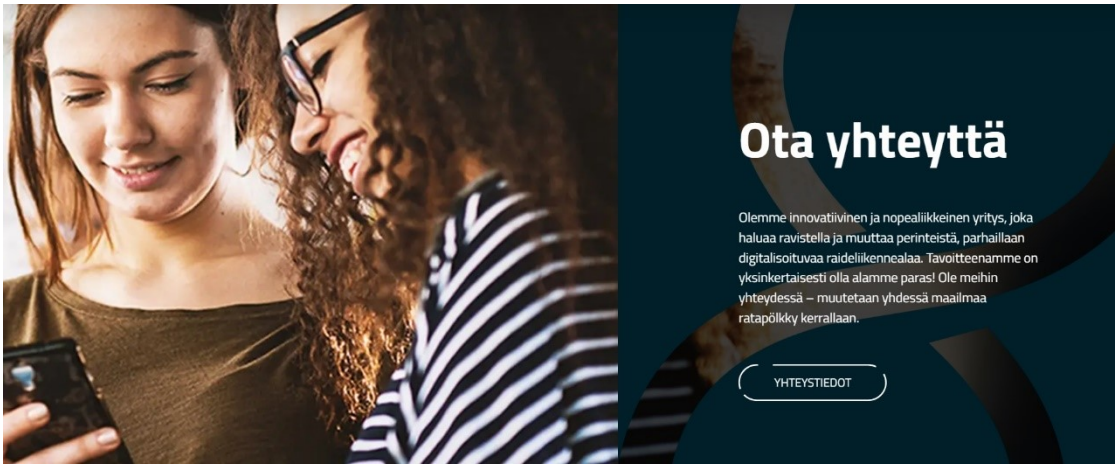
Teeman kehittäminen alkoi uuden lokaalin Wordpressin asentamisella. Lisäsimme toimeksiantajan valmiiksi pohjustetun teeman uuteen Wordpressin asennukseen ja otimme sen käyttöön hallintapaneelista. Otimme tässä vaiheessa käyttöön myös valmiiksi konfiguroidun *Gulp*-nimisen työkalun, joka kokoaa ositetut SCSS-tiedostot minimoituun (engl. *minified*) muotoon yhdeksi CSS-tiedostoksi. Tämä tarkoittaa sitä, että lopullisessa CSS-tiedostossa, jota selain käyttää, on poistettuna kaikki ylimääräiset merkit, kuten välilyönnit, kommentit sekä rivinvaihdot. Tällöin tiedostokoko pienenee, mikä taas nopeuttaa verkkosivuston toimintaa.

Teemaan luotiin uusi kansiorakenne, jossa Gutenberg-lohkojen PHP-, SCSS- ja JavaScript-tiedostot ositettiin omiin kansioihinsa. Yksittäisten lohkojen SCSS-tiedostojen lisäksi myös esimerkiksi muuttujille on varattu oma kansionsa. Kyseinen kansio sisältää mm. väreille sekä fonteille omat SCSS-tiedostot. Lopuksi kaikki ositetut tyylitiedostot kootaan yhdeksi globaaliksi tiedostoksi ja Gulp hoitaa pakkaamisen *global.min*-nimiseksi CSS-tiedostoksi, jota verkkosivusto lopulta lukee ja käyttää ulkoasua varten.

4.2 Lohkojen luominen

Tämän opinnäytetyön keskeisin ja tärkein yksittäinen uudistus on klassisesta editorista siirtyminen Gutenberg-editoriin. Täysin kustomoitavien lohkojen luonti onnistuu laajentamalla Gutenbergia ACF-lisäosan kanssa. Tässä luvussa näytetään yksittäisen lohkon rakentaminen alusta alkaen, vaikka valmis opinnäytetyö sisältää useita erilaisia lohkoja. Lohkojen rakenne ja syöttötietojen tyyppi toki vaihtuu, mutta niiden luominen ja rekisteröinti tapahtuu samalla tyylillä kuin seuraavaksi on esitetty.

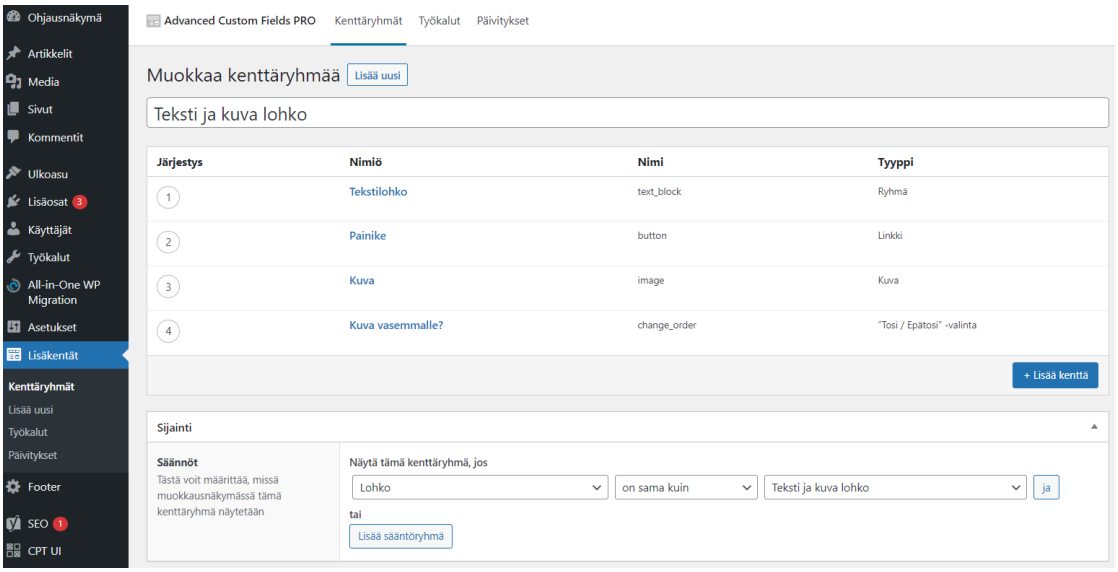
Teemaa varten haluttiin luoda lohko, joka on jaettu kahteen kolumniin. Toinen kolumni sisältäisi otsikon, tekstikentän sekä painikkeen, jolle voidaan antaa linkki esimerkiksi alasivulle oman verkkosivuston sisällä. Toinen kolumni olisi kokonaisuudessaan kuva ja näiden kahden kolumnin paikkaa voisi loppukäyttäjä halutessaan vaihtaa. Kuvassa 18 on esimerkki elementistä, joka löytyy Groteskin kehittämästä Proxion Oy:n verkkosivustosta. Kyseisessä elementissä on visuaalisuutta lisäävinä tekijöinä taustakuvan parallax-efekti ja painikkeen animaatio vietäessä hiiren kursori sen päälle. Nämä efektit ovat sellaisia, jotka räätälöidään kohteena olevan asiakasprojektin mukaan eikä täten ole järkevää toteuttaa teemaan tuleviin lohkoihin. Seuraavaksi esitelty kahden kolumnin lohko voisi siis toimia pohjana kuvan 18 valmiin elementin kehittämisessä, koska molemmat sisältävät samanlaisen rakenteen (otsikko, tekstielementti, painike ja kuva kahteen kolumniin jaettuna).



Kuva 18. Kuvakaappaus Proxion Oy:n verkkosivustolta

Ensimmäisenä asiana em. lohko rakennetaan Wordpressin hallintapaneelista luomalla uusi kenttäryhmä. Kenttäryhmä nimetään halutulla tavalla, minkä

jälkeen siihen tehdään tarvittavat syöttötiedot. Kuvassa 19 on kenttäryhmä nimeltä ”Teksti ja kuva lohko”, johon on tehty neljä erilaista elementtiä. Ensimmäisenä on ”Tekstilohko”, joka on kenttätyyppiä ”Ryhmä” eli se sisältää useita alakenttiä. Alakentät ovat tässä tapauksessa tekstikenttä ”Otsikko” sekä *Wysiwyg*-editoria käyttävä kenttä ”Teksti”. Tämän lisäksi lohkosta löytyy paikka painikkeelle, kuvalle sekä valintaruutu sille, haluaako kuvan vaihtaa oikealta puolelta vasemmalle. Lopuksi lohkolle luodaan sääntö, jonka avulla voidaan määritellä, missä näkymässä kenttäryhmä näytetään.



Kuva 19. Uuden kenttäryhmän luominen hallintapaneelissa

Lohko kutsutaan functions.php-tiedostossa acf_register_block-funktiolla, jonka sisällä määritetään mm. lohkon nimi ja sivupohja, jota halutaan käyttää lohkon esittämiseen (kuva 20). Lohkojen rekisteröintiin otettiin käytännöksi, että nimeämisessä käytetään apuna muuttujia. Esimerkiksi render_template-parametrille annetaan tässä tapauksessa arvoksi polku, joka muuttujan $block_slug avulla on kokonaisuudessaan muotoa: ”blocks/text-and-image-block.php”. Teeman hakemistosta löytyy siis *blocks*-niminen kansio, jossa on *text-and-image-block*-niminen PHP-tiedosto, jolla esitetään kyseinen rekisteröity lohko.

```php
/**
 * ACF Blocks
 */
function groteski_acf() {
    if ( function_exists( 'acf_register_block' ) ) {
        $js_dir = get_template_directory_uri() . '/js/dist';

        $block_name  = 'Teksti ja kuva lohko';
        $block_slug  = 'text-and-image-block';
        $description = 'Lohko tekstipaikalla ja kuvalla';

        acf_register_block_type(
            array(
                'name'            => $block_name,
                'title'           => $block_name,
                'description'     => $description,
                'enqueue_script'  => "$js_dir/blocks/$block_slug.min.js",
                'render_template' => "blocks/$block_slug.php",
                'keywords'        => array( $block_name ),
            )
        );
```

Kuva 20. ACF-lohkon rekisteröinti functions.php-tiedostossa

Seuraavaksi lohkolle tehdään uusi vastaavalla tavalla nimetty PHP-tiedosto, johon luodaan haluttu HTML-rakenne sekä käydään läpi tulevat syöttötiedot. Kuvassa 21 on esitetty rakenne aiemmin rekisteröidylle ”Teksti ja kuva” lohkolle. Aluksi otetaan syöttötiedot ACF-kentistä muuttujille, minkä jälkeen luodaan HTML-rakenne sekä tulostetaan muuttujien sisällöt.

```php
<?php
    $text_block = get_field('text_block');
    $button = get_field('button');
    $image = get_field('image');
    $order = get_field('change_order');
    if( $order ) :
        $class = 'left';
    else:
        $class = '';
    endif;
?>

<section class="text-and-image-block">
    <div class="grid">

        <div class="text-block <?php echo $class ?>">

            <h2><?php echo $text_block['title']; ?></h2>

            <?php echo $text_block['text']; ?>

                <?php if($button) :?>
                    <div class="buttons">
                        <a class="button" target="<?php echo esc_attr($button['target']); ?>"
                          title="<?php echo $button['title']; ?>" href="<?php echo $button['url']; ?>">
                           <?php echo $button['title']; ?>
                        </a>
                    </div>
                <?php endif; ?>
        </div>

        <div class="image">
            <img src="<?php echo esc_url( $image['sizes']['medium'] ); ?>" alt="<?php echo esc_attr( $image['alt'] ); ?>"/>
        </div>

    </div>
</section>
```

Kuva 21. Text-and-image-block.php-tiedoston rakenne

Lopuksi valmiille lohkolle luodaan samalla tavalla nimetty tyylitiedosto, jota käytetään ainoastaan tähän yksittäiseen lohkoon (kuva 22). Lohko jaetaan kahteen kolumniin antamalla *grid*-nimiselle luokalle ominaisuus (engl. *property*) ”grid-template-columns”. Tämän ominaisuuden arvolla määritetään gridin kolumnien määrä ja koko eli tässä tapauksessa kaksi kappaletta samankokoisia kolumneja. Lohkolle on määritelty myös ns. *breakpoint*, eli näyttökoko pikseleissä, jolloin käytetään tiettyjä ulkoasumäärittelyitä. Tässä tapauksessa näyttökoon ollessa maksimissaan $max-s-muuttujan arvo (767 pikseliä), kolumnijako poistuu eli sekä kuvasta että tekstikentästä tulee täysleveitä ja ne asettuvat allekkain verkkosivustolla.

```scss
.text-and-image-block {
    .grid {
        grid-template-columns: repeat(2, 1fr);
        .image {
            text-align: center;
        }
        .buttons {
            text-align: center;
            margin: 15px 0;
        }
        .left {
            grid-area: 1 / 2;
        }
        @media (max-width: $max-s) {
            grid-template-columns: 1fr;
            .left {
                grid-area: 1;
            }
        }
    }
}
```

Kuva 22. Text-and-image-block.scss-tiedoston rakenne

Valmiin lohkon ulkoasun muotoiluksi riittää käytännössä CSS Grid Layoutin käyttöönotto ja responsiivisuuden määrittely, koska loput ulkoasumäärittelyt vaihtuvat kohteena olevan asiakasprojektin mukaan. Siksi lohko onkin tässä vaiheessa toiminnoiltaan valmis käytettäväksi web-kehityksessä. Kuvassa 23 näkyy valmis ”Teksti ja kuva lohko”, johon on syötetty esimerkin vuoksi sisältöä.

Kuva 23. Teksti ja kuva -lohko selainikkunassa

Valmiiseen teemaan rakentui lopulta yhteensä kymmenen kappaletta erilaisia lohkoja. Lohkoista kaksi on varattu sivuston hero-elementiksi, jolla tarkoitetaan suurta visuaalista aluetta sivuston ylälaidassa. Sillä luodaan yleensä ensivaikutelma sivuston kävijälle esimerkiksi kuvan tai videon avulla. Muutaman kolumneihin jaetun tekstilohkon lisäksi koettiin tarve mm. ”yhteystietojen lohkolle” (*engl. contacts block*) sekä ns. ”nostolohkolle” (*engl. lift block* tai *post block*), johon loppukäyttäjä voi tarvittaessa laittaa esimerkiksi blogikirjoituksia, artikkeleita tai työpaikkailmoituksia näkyville.

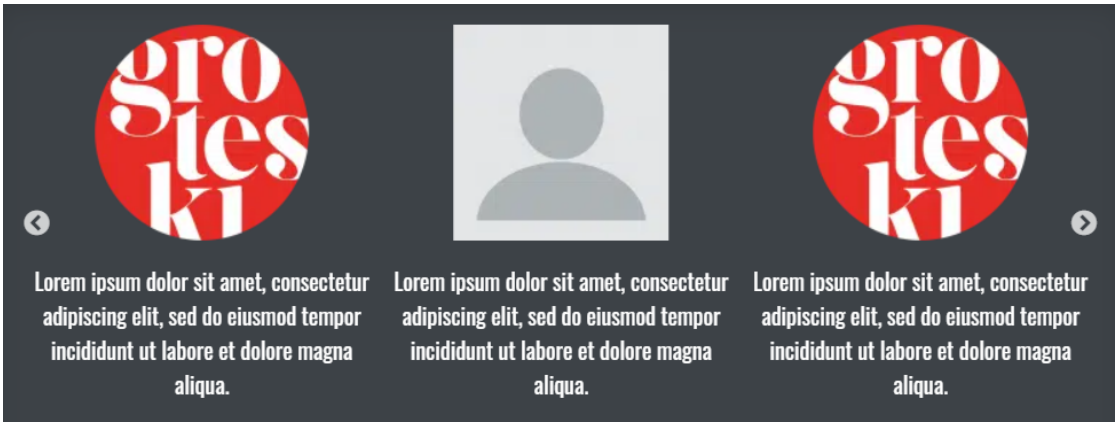
Eräs yleisesti käytetty elementti verkkosivuistoissa on nimeltään ”karuselli” (engl. *slider*), joka myös luotiin osaksi uutta teemaa omana lohkonaan. Sen avulla käyttäjä voi selata horisontaalisesti painikkeiden avulla kuvia tai muuta sisältöä. Karuselliin voi laittaa kuinka paljon sisältöä tahansa, mutta esimerkiksi vain kolme niistä näkyy kerrallaan ruudulla.

Teemaan otettiin käyttöön *Slick*-niminen jQuery-kirjasto. Sen avulla pystyy rakentamaan kohtuullisen helposti muokattavia ja responsiivisia karuselleja eri käyttötarkoituksia varten. Slick saadaan käyttöön lataamalla siihen kuuluvat JavaScript- ja SCSS-tiedostot osaksi teemaa tai vaihtoehtoisesti lisäämällä CDN-linkit. Karusellilohko rekisteröitiin samalla tapaa kuten aiemmin on esitetty. Sille määritettiin syöttötiedoiksi kuva, otsikko ja tekstikenttä, jotka PHP-koodissa käydään läpi ja tulostetaan HTML elementtiin, joka on luokkaa ”slick-carousel”.

```
import { breakpoints } from '/js/src/variables/breakpoints';

jQuery( document ).ready( function() {
    jQuery('.slick-carousel').not('.slick-initialized').slick({
        infinite: true,
        speed: 300,
        slidesToShow: 4,
        slidesToScroll: 1,
        arrows: true,
        responsive: [
            {
                breakpoint: breakpoints.max_m,
                settings: {
                    slidesToShow: 3,
                    slidesToScroll: 1,
                },
            },
            {
                breakpoint: breakpoints.max_s,
                settings: {
                    slidesToShow: 2,
                    slidesToScroll: 1,
                },
            },
```

Kuva 24. Slick-toiminnon määrittely slider-block.js-tiedostossa

Slickin toiminta perustuu HTML-elementtiin kohdistuvaan *slick*-funktioon, jolle voidaan määrittää erilaisia asetuksia. Lohkon omaan slider-block.js-tiedostoon lisättiin em. funktio (kuva 24), jolle määriteltiin esimerkiksi, kuinka monta sisältöelementtiä halutaan näyttää kerrallaan (*slidesToShow*) ja monta elementtiä liikutaan yhdellä painalluksella (*slidesToScroll*). Muita määrityksiä ovat karusellin loputon pyöriminen, nopeus ja nuolinäppäinten asettaminen näkyviksi. Myös responsiivisuuden määrittely onnistuu samassa yhteydessä, eikä täten erillisiä SCSS-määrityksiä tarvita.



Kuva 25. Valmis karusellilohko selainikkunassa

Kuvassa 25 on valmis karusellilohko, johon on syötetty esimerkin vuoksi sisältöä. Laidoilla olevista nuolista karuselli liikkuu yhden sisältöelementin verran joko oikealle tai vasemmalle. Kaikki muut teemaan kuuluvat lohkot ovat esillä liitteen 1 kuvakaappauksissa.

4.3 Sivuston toistuvat elementit

Edellisessä luvussa esiteltiin lohkojen luominen alusta alkaen aina syöttötietojen asettamisesta koodiin saakka. Näistä lohkoista on tarkoitus koostaa suurin osa teemalla kehitettävien verkkosivustojen sisältöalueista, oli sitten kyseessä yksinkertainen blogisivu tai esimerkiksi laajempi yrityksen kotisivu. Kyseisten lohkojen lisäksi verkkosivustot sisältävät lähes poikkeuksetta sivujen välillä toistuvia elementtejä, kuten aiemmin työssä mainitun hero-alueen ja alatunnisteen tai valikkorakenteen sivuston ylälaitaan. Tässä luvussa esitellään teemaan toteutettu alatunniste eli tuttavallisemmin *footer*.

Footer-osioon asetetaan monesti sisältöä, jota ei muuteta tai päivitetä kuin poikkeustapauksissa. Tämän kaltaista sisältöä voi olla yrityksen yhteystiedot, sosiaalisen median linkit tai tietosuojaseloste. Tämän vuoksi footeria on teknisesti hankala suunnitella soveltuvaksi moneen erilaiseen tarpeeseen, joten yksi vaihtoehto olisi luoda useita erilaisia footer-elementtejä. Päätimme kuitenkin toimeksiantajan kanssa, että teema sisältäisi vain yhden footerin. Tähän lisättäisiin useita erilaisia rakenteita, joita voitaisiin tarpeen tullen kopioida tai poistaa kokonaan. Lopputuloksena olisi mahdollisimman yleiskäyttöinen kokonaisuus, jotta se toimisi pohjana monipuolisiin asiakastarpeisiin.

Footerin rakentaminen on toteutettu teemaan pitkälti samalla kaavalla ACF-lisäosan kanssa kuten edellisessä kappaleessa näytettiin. Lisäkenttiin on luotu uusi kenttäryhmä nimeltä ”Footer”, jonne on tehty rakenne sisältötyypeille. Koska footereita löytyy teemasta vain yksi kappale, on sille luotu oma asetussivu (engl. *options page*). Asetussivu lisätään WordPressin ohjausnäkymään functions.php-tiedostossa kuvan 26 tapaan sekä antamalla sitä vastaava sääntö kenttäryhmää luodessa. Tällöin saadaan ohjausnäkymän valikkoon oma hallinta footerin sisällönsyötölle. Tästä syystä footeria ei myöskään näy em. sivujen muokkausnäkymässä muiden lohkojen tapaan.

```php
/**
 * ACF Options to admin menu
 */
if ( function_exists( 'acf_add_options_page' ) ) {
    acf_add_options_page(array(
        'page_title' => 'Footer',
        'menu_title' => 'Footer',
        'menu_slug'  => 'footer-settings',
        'redirect'   => false,
    ));
}
```

Kuva 26. Footerin asetussivun lisääminen

Koska footereiden rakenne ja käyttötarkoitus vaihtelee todella paljon erilaisten verkkosivustojen välillä, päätettiin sille luoda useita erilaisia sisältötyyppejä. Perinteisten taustakuvan, logon, otsikon ja tekstikentän lisäksi footerista tulisi löytyä valikkorakenne sivuston omille alasivuille, että ulkopuolisille sivuille kuten sosiaalisen median kanaville. Tästä syystä footerin yksi grid-rakenne sisältää kolme kolumnia, joista jokaisen sisältö on tyypiltään erilainen. Jos esimerkiksi jotain näistä kolumneista ei tarvita, voi kyseisen kolumnin HTML- ja PHP-koodin poistaa kokonaan ja muuttaa gridin asettelua tekemällä pienen muutoksen SCSS-koodiin. Tämä tapa on helpompi ja nopeampi kehittäjälle kuin se, että esimerkiksi sivuston sisäinen navigaatio täytyy rakentaa joka kerta uudestaan, kun sille tulee tarve uuden verkkosivuston kehittämisessä.

Footerina toimiva template-tiedosto footer.php löytyy oletuksena Underscores-teeman kansiosta. Tähän tiedostoon on luotu tarvittava rakenne <footer> HTML-tagien sisään. Tässä tapauksessa on käytetty kolmea ruudukkoa, joista ylin sisältää kaksi kolumnia (logo ja otsikko sekä teksti), keskimmäinen kolme kolumnia (yhteystiedot, sivuston valikko sekä sosiaalisen median linkit) ja alimmainen yhden kolumnin (linkki Groteskin omille verkkosivuille).

```
73          <div class="footer-right-block">
74              <ul>
75                  <?php while( have_rows('footer_some_block', 'option') ) : the_row();
76                      if( get_sub_field('some_link', 'option')) : ?>
77                      <?php
78                          $some_link = get_sub_field('some_link','option');
79                          $some_icon = get_sub_field('some_icon', 'option');
80                      ?>
81                          <li>
82                              <a href="<?php echo $some_link['url'] ?>" title="<?php echo $some_link['title']; ?>">
83                                  <span>
84                                      <img src="<?php echo esc_url($some_icon['sizes']['thumbnail']); ?>" />
85                                      <?php echo $some_link['title']; ?>
86                                  </span>
87                              </a>
88                          </li>
89                      <?php endif;
90                  endwhile; ?>
91              </ul>
92          </div>
93
94      </div> <!--CENTER GRID END-->
```

Kuva 27. Sosiaalisen median linkkien kolumni footer.php-tiedostossa

Kuvassa 27 on esitetty, kuinka footerin sosiaalisen median linkeille luotu kenttä tyypiltään ”toista rivejä” (engl. *repeater*) on toteutettu koodissa. Aluksi koodissa käydään PHP:n ns. *while*-silmukalla läpi kentälle annetut sisällöt. Silmukka toistaa koodia niin kauan kuin sille annettu ehto on voimassa. Tässä tapauksessa tulostetaan HTML-listaan linkkejä, joille haetaan syöttötietoina url, kuva sekä otsikko.

Kuva 28. Valmis footer-elementti selaimessa

Lopuksi footerin grid-ruudukoille on tehty tarvittavat SCSS-muotoilut, jotta rakenne pysyy kasassa myös erikokoisilla näytöillä. Kuvassa 28 näkyy valmis footer testisisältöjen kanssa selainikkunassa.

4.4 Testaus demosivustolla

Koska opinnäytetyön aikataulun vuoksi ei ollut mahdollisuutta testata teemaa oikeassa asiakasprojektissa, päädyin kokoamaan lohkoista sivun testaamista varten. Se toimisi teeman etusivuna ja sisältäisi yhden kappaleen jokaista luotua elementtiä sekä footer-osion sivun alalaidassa. Tällöin lohkojen responsiivisuutta ja sisällönsyöttöä voisi testata selaimessa. Demosivu toimisi myös kehittäjille helppona ja nopeana tapana tarkastaa teemaan kuuluvat lohkot ennen varsinaisen projektin aloittamista.

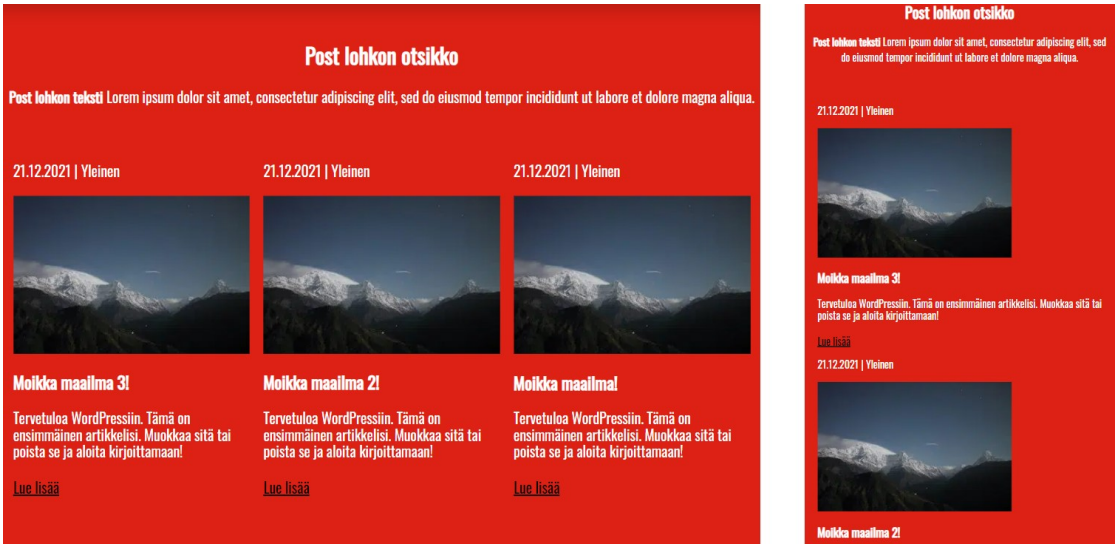
```
1  section[class*='-block']{
2      box-shadow: 0px 20px 30px -20px rgba(0,0,0,0.5) inset;
3  }
4  section[class*='-block']:nth-child(even){
5      background-color: #de2115;
6      color: white;
7  }
8  section[class*='-block']:nth-child(odd){
9      background-color: #3D4247;
10     color: white
11 }
12 //--------------------------------------------------------------------------
13 // Yllä olevat demosivun muotoilut voi poistaa kun aloittaa uutta projektia!
```

Kuva 29. Demosivun SCSS-muotoilut

Päätin määrittää demosivustolle ulkoasun siten, että joka toinen lohko olisi taustaväriltään erilainen. Tällöin sivun avattuaan on helpompi havaita yksittäiset lohkot ja niiden sisältämän rakenteet. Värisävyiksi valitsin Groteskin omaan verkkosivustoon perustuvat harmaan sekä punaisen, joita käytin koko sivustoa koskevassa layout.scss-tiedostossa lohkojen taustavärinä (kuva 29). Käytin valitsimina HTML <section> -elementtejä, jotka sisältävät block-luokan. Määritin jokaiselle lohkolle varjon (*box-shadow*) ja taustavärin sen mukaan onko kyseessä parillinen (*even*) vai pariton (*odd*) elementti. Teemaa käyttävän kehittäjän on hyvä muistaa, että edellä mainittuja SCSS-muotoiluja käytetään vain demosivun kanssa, joten ne voi poistaa heti uuden verkkosivuston kehitysprojektin alussa.

Jokaiselle lohkolle on määritetty responsiivisuutta varten muotoiluja, jotka testattiin käyttäen Google Chrome -selaimen kehittäjän työkaluja (engl. *developer tools*). Teemassa käytössä olleet breakpoint-muuttujat alkoivat 360 pikselistä, joten sen alle jäävät näyttökoot eivät vaikuttaneet lohkojen ulkoasun

suunnitteluun. Yleisin määritelmä jokaiselle lohkolle oli se, että elementtien kolumnijako vähenee sitä mukaa, kun näyttökoko pienenee. Lopuksi kolumnijako poistetaan kokonaan, kun näyttökoko saavuttaa 480 pikselin eli tällöin elementit asettuvat allekkain näytölle.



Kuva 30. Post-lohkon responsiivisuuden testaus

Kuvassa 30 on havainnollistettu responsiivisuuden toimintaa. Vasemmalla puolella kuvaa on *Post*-niminen lohko kannettavan tietokoneen näytöllä. Lohko sisältää kolme kappaletta testiartikkeleita vierekkäin grid-rakenteen sisällä. Oikealla puolella kuvaa on sama lohko, mutta näyttökoko on leveydeltään vain 400 pikseliä, joka voisi vastata mobiililaitteiden näyttökokoa. Tällöin yksittäiset artikkelinostot asettuvat allekkain, eli kolumnijako poistuu käytöstä.

5 PÄÄTÄNTÖ

Sovimme toimeksiantajan kanssa opinnäytetyöstä kesällä 2021 työharjoitteluni loppuvaiheessa. Vaihtoehtoja aiheeseen oli muutamia, mutta päädyimme lopulta teeman uusimiseen, koska se oli ajankohtaista ja jopa pakollista asiakkaiden halusta siirtyä Gutenberg-editorin käyttöön. Pidin myös itse tärkeänä, että aihe sekä siitä seurannut lopputulos olisi jollain tapaa merkittävä ja hyödyllinen toimeksiantajalle. Löimme aiheen lukkoon ja asetimme työlle aikatauluksi kevään 2022, jolloin teema otettaisiin käyttöön.

Tiesin työn olevan laaja ja aikaa vievä kokonaisuus, joten käytin ensimmäiset kuukaudet teorian ja tekniikoiden opetteluun, ennen varsinaisen kehitystyön

käynnistämistä. Koin mielenkiintoiseksi selvittää pohjaa ja historiaa WordPress-kehitykselle, koska olin juuri edellisenä kesänä itse ollut web-kehittäjänä työharjoittelussa kymmenen viikon ajan. Tästä syystä Groteskin aiempi teema oli jollain tapaa tuttu, joka auttoi hieman suunnittelua sekä uuden teeman ideointia.

Yhteistyö toimeksiantajan kanssa sujui mielestäni erinomaisesti ja joustavasti vallitsevan tilanteen huomioon ottaen osaksi myös etänä. Yhteiset palaverit toimivat hyvänä tilaisuutena jakaa ajatuksia ja tietoa parhaan lopputuloksen saavuttamiseksi. Kesken opinnäytetyön eteen tuli myös yksi projekti, jossa Groteskin kehittäjät joutuivat käyttämään Gutenberg-editoria, vaikka uuden teeman kehittäminen oli kesken. Emme antaneet tämän kuitenkaan vaikuttaa esimerkiksi työn valmistumiseen, vaan pysyimme alkuvuoden 2022 julkaisuaikataulussa.

Opinnäytetyö kokonaisuutena antoi todella paljon uutta oppia sekä pääsin myös soveltamaan jo aiemmin hankittuja taitoja monipuolisesti. Teeman kehitys vaati suunnittelun lisäksi myös sellaista teknistä osaamista, jota olin saanut ainoastaan työharjoittelun kautta. Syvempi oppiminen esimerkiksi WordPressin hierarkiaan ja teeman rakenteeseen tuli kuitenkin vasta tämän opinnäytetyön kautta. Valmiiseen työhön olen varsin tyytyväinen ja koen, että tavoiteltu lopputulos saavutettiin. Teemalle määritetyt asiat kuten saavutettavuutta ja responsiivisuutta koskevat toimenpiteet sekä tarvittavat Gutenberg-editorin lohkot valmistuivat osaksi teemaa.

Aiheesta johtuen teema itsessään ei ole täydellisesti valmis, vaan enemmänkin ensimmäinen toimiva julkaistu versio. Teeman käyttöönoton jälkeen sitä tullaan jatkokehittämään ja laajentamaan projekteista saatujen tulosten perusteella. Omasta mielestäni jatkokehityksen kannalta onkin tärkeää saada kokemuksia esimerkiksi lohkojen soveltumisesta erilaisiin käyttötarkoituksiin. Uskoisin myös, että teeman jatkokehitykseen tulee vaikuttamaan ajankohtaiset trendit sekä uudet visuaaliset ratkaisut verkkosivustojen ulkoasun suunnittelussa.

LÄHTEET

Carr, D. & Gray, M. 2018. Beginning PHP: master the latest features of PHP 7 and fully embrace modenr PHP development. E-kirja. Birmingham: Packt Publishing. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 12.10.2021].

Getbootstrap. 2021. Introduction. WWW-dokumentti. Saatavissa: https://getbootstrap.com/docs/5.0/getting-started/introduction/ [viitattu 10.1.2022].

Hakukonemestarit. 2020. Miten WordPress.com ja WordPress.org eroavat toisistaan. Blogi. Saatavissa: https://www.hakukonemestarit.fi/blogi/miten-eroaa-wordpress-com-ja-wordpress-org/ [viitattu 25.9.2021].

Jalonen, L. 2020. Gutenberg – WordPressin oma lohkoeditori. WWW-dokumentti. Saatavissa: https://www.zoner.fi/wordpress/gutenberg/ [viitattu 11.10.2021].

MacLees, N. 2014. jQuery for Designers Beginner's Guide Second Edition. E-kirja. Birmingham: Packt Publishing. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 18.10.2021].

Miller, A. 2021. WordPress Updates: The Biggest Changes Over the Years. Creative Minds. WWW-dokumentti. Saatavissa: https://www.cminds.com/wordpress-5-0-ensure-maximum-compatibility/ [viitattu 19.9.2021].

Sass. 2021. Sass Basics. WWW-dokumentti. Saatavissa: https://sass-lang.com/guide [viitattu 12.10.2021].

The History of WordPress. 2021. WPBeginner. WWW-dokumentti. Päivitetty 19.8.2021. Saatavissa: https://www.wpbeginner.com/news/the-history-of-wordpress/ [viitattu 19.9.2021].

W3techs. 2021. Usage statistics of content management systems. WWW-dokumentti. Saatavissa: https://w3techs.com/technologies/overview/content_management [viitattu 11.10.2021].

WordPress. 2019. About. WWW-dokumentti. Saatavissa: https://fi.wordpress.org/about/ [viitattu 19.9.2021].

WP-kotisivut. 2021a. Mikä on WordPress teema. WWW-dokumentti. Saatavissa: https://www.wp-kotisivut.com/wordpress/teemat/ [viitattu 11.10.2021].

WP-kotisivut. 2021b. Parhaat WordPress lisäosat 2021. WWW-dokumentti. Saatavissa: https://www.wp-kotisivut.com/wordpress/plugins-lisaosat/ [viitattu 11.10.2021].

Wpbeginner. 2021. 15 Best and Most Popular CMS Platforms in 2021 (Compared). WWW-dokumentti. Päivitetty 14.8.2021. Saatavissa: https://www.wpbeginner.com/showcase/best-cms-platforms-compared/ [viitattu 24.9.2021].

KUVALUETTELO

Kuvakaappaukset demosivustosta

Kuvakaappaukset demosivustosta