Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2024

Liisa Kotilainen

WordPress-teeman laatuun vaikuttavat tekniset ratkaisut

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2024 | 39 sivua

Liisa Kotilainen

WordPress-teeman laatuun vaikuttavat tekniset ratkaisut

WordPress on avoimen lähdekoodin sisällönhallintajärjestelmä, jonka avulla voi luoda ja hallinnoida verkkosivuja. WordPress-teema on esitysmalli, joka määrittelee verkkosivun ulkoasun ja käyttäjäkokemuksen, mahdollistaen sivuston räätälöinnin ja muokkaamisen.

Työn tavoite oli tutkia, millaisia teknisiä ratkaisuja suositellaan WordPress-teeman toteutuksessa, miksi niitä suositellaan ja miten ne vaikuttavat laatuun. Lisäksi suositeltuja ratkaisuja tarkasteltiin käytännössä tutkimalla kolmannen osapuolen teemoja sekä soveltamalla joitakin ratkaisuja omassa teemassa.

Työssä selvisi, että WordPress-teeman tekniseen laatuun vaikuttavat useat osa-alueet, kuten koodin selkeys ja ylläpidettävyys, suorituskyky, turvallisuus, optimointi, responsiivisuus, saavutettavuus sekä yhteensopivuus WordPressin ydinkoodin, lisäosien ja laajennusten kanssa. Pohjatyö teknisesti laadukkaan teeman luomiseen on työlästä, mutta parantaa teeman ylläpidettävyyttä ja laajentamista, joka taas johtaa menestykseen pidemmällä aikavälillä.

Työtä voisi jatkaa tutkimalla muiden teknistä laatua parantavien teknologioiden ja kehitystyökalujen, kuten Node.js ja React, yhdistämistä WordPressiin.

Asiasanat:

WordPress, verkkosivukehitys, web-ohjelmointi

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2024 | 39 pages

Liisa Kotilainen

Technical solutions affecting a WordPress theme

WordPress is an open-source content management system that allows for the creation and management of websites. A WordPress theme is a design template that defines the appearance and user experience of a website, enabling customization and modification.

The objective of this work was to investigate the technical solutions recommended for implementing a WordPress theme, why they are recommended, and how they affect quality. Additionally, the recommended solutions were examined in practice by studying third-party themes and applying some solutions to our own theme.

The study revealed that several aspects influence the technical quality of a WordPress theme, such as code clarity and maintainability, performance, security, optimization, responsiveness, accessibility, and compatibility with WordPress core, plugins, and extensions. The groundwork for creating a technically high-quality theme is labor-intensive, but it improves the theme's maintainability and scalability, leading to long-term success.

Future work could involve exploring the integration of other technologies and development tools that enhance technical quality, such as Node.js and React, with WordPress.

Keywords:

WordPress, web-development, web-design

Sisältö

Kuvat

Taulukot

Ohjelmat

1 Johdanto

WordPress on helppokäyttöinen sisällönhallintajärjestelmä, jolla voi luoda ja hallinnoida verkkosivuja. Sivustoa voi räätälöidä ja muokata käyttämällä teemoja, jotka määrittelevät verkkosivujen ulkoasun. Kolmannen osapuolen teemoja on saatavilla lukuisia, mutta teemaa voi kehittää myös itse joko tiettyyn tarpeeseen tai WordPress-yhteisölle jaettavaksi. WordPress-teeman kehittäminen vaatii tietämystä vähintään PHP-, HTML- ja CSS-kielistä, mutta myös JavaScriptin osaaminen on hyödyksi.

Vaadittujen ohjelmointikielten hallitsemisen lisäksi teeman laatuun vaikuttaa tekniseen toteutukseen liittyviä seikkoja, kuten tiedostohierarkia ja modulaarisuus WordPressissä, turvallisen ja saavutettavan koodin kirjoittaminen sekä optimointi ja mukautettavuus. Modernien verkkosivujen ja siten myös WordPress-teeman tulee olla responsiiviset ja sopia käytettäviksi myös mobiililaitteilla. Työssä perehdytään yleisellä tasolla parhaisiin käytäntöihin, joita suositellaan teeman kehittämisessä. Opinnäytetyön huomio on yksinomaan teknisissä toteutuksissa ja niiden soveltamisessa käytännössä.

Työn tavoitteena on saavuttaa ymmärrys teeman laadusta ja millaisista asioista laatu kokonaisuudessaan koostuu. Tavoitteena on myös paneutua perusteluihin suositusten taustalla ja teknisiin toteutuksiin käytännössä opettelemalla soveltamaan niitä omassa teemassa. Lisäksi tavoitteena on tarkastella, onko suositusten noudattamisesta hyötyä teemankehittäjille, ja oppia millaista työtä suositusten noudattaminen käytännössä vaatii.

Luvussa 2 esitellään WordPress-teeman perusteet yleisellä tasolla. Luvussa 3 tavoitteena on tutustua keskeisimpiin parhaisiin teknisiin suosituksiin tutkimalla niin WordPressin omia tietopankkeja kuin WordPress-ammattilaisten julkaisuja. Lisäksi tutkitaan parhaiden käytäntöjen toteutumista joissakin kolmannen osapuolen teemoissa. Luvussa 4 sovelletaan harjoitustyön kannalta olennaisia teknisiä ratkaisuja esimerkkiteemassa ja päätösluvussa koostan opinnäytetyön keskeisimmät asiat ja havainnot.

Opinnäytetyö tukee projektitoimisto theFIRMAlle toteutettavaa harjoitustyötä, jossa luodaan manuaali ja malliteema sekä yksinkertaiset verkkosivut oppilaiden apuvälineeksi WordPress-verkkosivujen räätälöimisessä ja teemankehityksen opettelussa. Opinnäytetyössä esiteltäviä teknisiä ratkaisuja sovelletaan harjoitustyön teemassa, joten ratkaisut valikoidaan sen mukaan, mikä on olennaista harjoitustyön kannalta.

Rajauksina opinnäytetyössä ei käydä läpi koodi- tai merkintäkielten perusteita. PHP:n, JavaScriptin, HTML:n ja CSS:n merkintätavoille on omat suositukset myös WordPress Developer Resourcesin mukaan, ja ne huomioidaan soveltavassa osuudessa, mutta näihin käytäntöihin ei perehdytä erikseen. Teemoihin liittyvää perusteoriaa käydään läpi toisessa luvussa, mutta teemankehityksen prosessia ei käydä opinnäytetyössä läpi. Soveltavassa osuudessa WordPress on asennettu paikalliselle Apache HTTP -palvelimelle avoimen lähdekoodin ohjelmistolla XAMPP, mutta asennusprosessia ei huomioida opinnäytetyössä. Opinnäytetyö ei myöskään käy saavutettavuuteen liittyviä yksityiskohtia tai lisäosia.

2 Teeman perusteet

WordPress-teema hallitsee, miten WordPress-sivuston sisältöä näytetään. Teemat koostuvat joukosta tiedostoja, jotka yhdessä muodostavat sivuston ulkoasun ja toiminnallisuuden. Teemoja voidaan luoda henkilökohtaiseen käyttöön, asiakkaille, tai niitä voidaan tarjota WordPress-yhteisön käyttöön. Teemoissa voidaan hyödyntää erilaisia malleja (eng. templates), merkintöjä (eng. tags) ja WordPress Loopia luomaan uniikkeja ratkaisuja. Niitä voidaan myös käyttää omien mallien luomiseen tiettyjä sivuja tai sivutyyppejä varten sekä vaihdella eri ulkoasuvaihtoehtojen välillä. (WordPress Codex n.d.)

Rakentamalla teeman itse, voi saavuttaa täysin räätälöidyn ja ainutlaatuisen lopputuloksen. Lisäksi teeman voi luoda skaalattuna esimerkiksi brändin yksilöllisiin tarpeisiin. Koska tarpeeton koodi voidaan jättää pois, myös suorituskyky ja latausajat voidaan optimoida, jolloin käyttäjäkokemus paranee. Myös hakukoneoptimointia ja turvallisuutta voidaan parantaa tekemällä yksilöityjä ratkaisuja. (Mahfuz 2024.)

2.1 Tematyypit

WordPressissa on pääasiassa kaksi tematyyppiä: klassinen teema ja vuonna 2022 käyttöön otettu lohkoteema. Lisäksi olemassa on erilaisia hybriditeemoja, joissa klassiseen teemaan yhdistellään lohkoteemalle tyypillisiä ominaisuuksia, ja käytännössä teeman voi rakentaa millaiseksi tahansa. Taulukossa 1 on vertailtu klassisen teeman ja lohkoteeman pääpiirteisiä eroja. (Alfredo Navas 2023.)

Taulukko 1. Klassisen ja lohkoteeman erot.

Klassinen teema	**Lohkoteema**

PHP, HTML, CSS ja mahdollisesti JavaScript.	HTML, CSS ja JavaScript, PHP dynaamiseen dataan.
PHP-mallit sivuille sekä sivujen osille ja CSS-tyylitiedosto.	HTML-tiedostot sisältävät mallit lohkojen asetteluun.
WordPress Customizer tyylimuutoksille. Isommat muutokset voivat vaatia koodaustaitoja tai lisäosien käyttöä.	Lohkoeditori koko sivun mallin editoimiseen. Kustomointi on helppoa ilman koodaustaitoja.
Ylä- ja alatunnisteet, menut ja logot rekisteröidään functions.php-tiedostossa.	Ylä- ja alatunnisteet, menut ja logot kustomoidaan lohkoissa.
Käyttää vimpaimia.	Käyttää lohkoja.

Klassinen teema on Wordpressin alkuperäinen tematyyppi ja edelleen laajalti käytössä. Klassinen teema käyttää PHP-pohjaisia malleja verkkosivujen rakenteen luomiseen ja sisällön näyttämiseen. Klassisessa teemassa sisällöntuottajalle voidaan antaa mahdollisuus muokata erilaisia elementtejä Customizer-työkalulla, mutta mahdollisuudet sivumallien suurempaan muokkaukseen ovat rajalliset. (Alfredo Navas 2023.)

Lohkoteema tuli käyttöön kun WordPressin versio 5.9 julkaistiin vuonna 2022. Kyseinen tematyyppi käyttää HTML-mallitiedostoja, joiden avulla verkkosivut rakennetaan lohkoista. Lohkot ovat klassista teemaa vapaammin muokattavissa WordPress-editorissa ja erilaisia sivuja voi rakentaa yhdistelemällä lohkoja. Räätälöitävät tyylit ja ominaisuudet määritellään tiedostossa theme.json. Sisällöntuottajalla on siis enemmän vapauksia päivittää teemaa suoraan editorissa koskematta koodiin. (Marcus Kazmierczak 2022.)

Hybriditeemassa käytetään klassisen teeman PHP-pohjaisia malleja, mutta teemassa voidaan lisäksi hyödyntää esimerkiksi tiedostoa theme.json ja antaa sisällöntuottajan muokata osioita eli lohkoja verkkosivujen sisällöstä

lohkoeditorin avulla, mutta ei teemaa itsessään. WordPressin versiosta 6.1 lähtien klassisiin teemoihin on voinut lisätä lohkotyyppisiä malleja, joita sisällöntuottaja voi muokata ja hallita hallintapaneelin kautta. (Alfredo Navas 2023.)

2.2 Teeman rakenne

2.2.1 Tiedostot

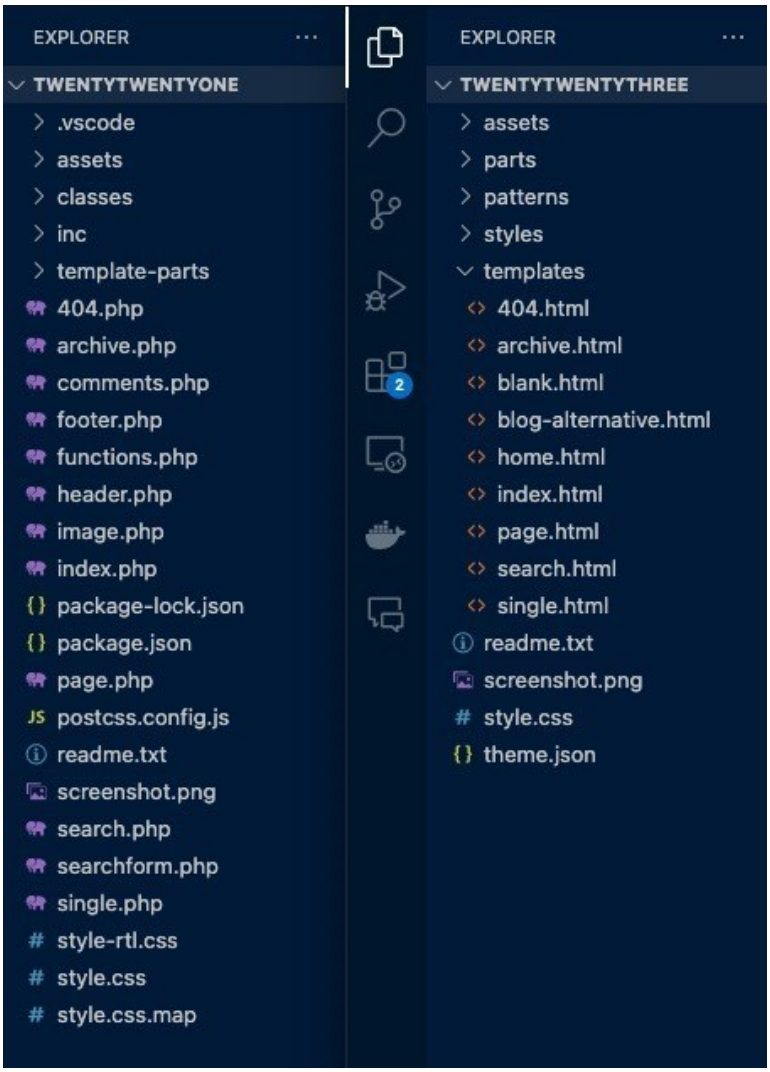
Toimiakseen teema tarvitsee vähintään tyylitiedoston style.css ja mallitiedoston index.php tai index.html. Index-tiedosto toimii oletusmallina etusivulle sekä muille sivuille, jos muita malleja ei löydy. Teeman mallitiedostot sisältävät yleensä myös globaaleja osia verkkosivuista, kuten mallin ylätunnisteelle (eng. header) sekä alatunnisteen (eng. footer). Lisäksi teema voi sisältää sivupalkin (eng. sidebar) ja erilaisia sivutyyppikohtaisia mallitiedostoja, kuten eri mallit artikkeleille (eng. posts) ja sivuille (eng. pages). (Joost de Valk 2011.)

Lisäksi teeman toiminnallisuutta voidaan hallita tiedostolla functions.php. WordPress ei vaadi tiedostoa, mutta se sisältyy kaikkiin tarjolla oleviin teemoihin, sillä se on olennainen osa teeman käyttäytymisen mukauttamisen kannalta. Tiedostossa voidaan muun muassa rekisteröidä menuja sekä alueita vimpaimille (eng. widgets), lisätä tyylejä ja komentosarjoja ja määrittää teemalle ominaiset asetukset. Lisäksi tiedostossa voidaan hyödyntää WordPressin tarjoamia koukkuja (eng. hooks) ja suodattimia (eng. filters), joilla voidaan mukauttaa WordPressin ja lisäosien ydintoimintoja tai aktivoida mukautettuja toimintoja. (WordPress Developer Resources 2023.)

WordPress itsessään sisältää JavaScript-ohjelmointikieltä, joten myös teemassa voidaan lisätä toimintoja JavaScriptillä. Esimerkiksi kolmannen osapuolen tarjoamien työkalujen, kuten upotetun kyselylomakkeen lisäämiseen voidaan tarvita JavaScriptiä. (Nazaryan 2024.)

Teemakansion sisältö näyttää erilaiselta riippuen onko kyseessä klassinen vai lohkoteema, kuten nähdään Alfredo Navasin (2023) esimerkistä, jossa

vertaillaan WordPressin klassista teemaa Twenty Twenty One ja lohkoteemaa Twenty Twenty Three. (Kuva 1.) Klassisessa teemassa PHP-mallit sijaitsevat teeman juurikansiossa, kun taas lohkoteemassa vastaavat HTML-mallit tulee sijoittaa kansioon templates. Teeman Twenty Twenty Three kansiosta uupuu myös tiedosto functions.php, sillä lohkoteemassa kyseistä tiedostoa ei vältämättä tarvita. Tiedosto voitaisiin kuitenkin lisätä, mikäli teemassa haluttaisiin ottaa käyttöön lisää tyylitiedostoja tai skriptejä. (Navas 2023.)



Kuva 1. Klassisen ja lohkoteeman rakenne.

2.2.2 Mallit

WordPress-mallit määrittelevät sivuston eri osien rakenteen. Ne voivat koskea yksittäistä sivua tai kokonaisen sivutyyppin, kuten esimerkiksi tuotesivun mallia. Tyypillisessä mallissa pyydetään WordPressiä noutamaan ensin ylätunniste, sitten sivun pääsisältö ja viimeiseksi alatunniste. Itse sisällön näyttämistä hallitsee niin kutsuttu WordPress Loop, joka on WordPressin standardi tapa noutaa ja näyttää verkkosivujen sisältöä dynaamisesti teemassa. (WordPress Developer Resources 2015.)

Klassisessa teemassa käytetään sisäänrakennettuja funktioita sisällön näyttämiseen ja eri mallien hakemiseen. Esimerkiksi ylätunniste haetaan käyttämällä sisäänrakennettua funktiota get_header(). WordPress sisältää paljon erilaisia funktioita, joilla mallin toiminnallisuutta voidaan muokata. The Loopin tapaa näyttää sisältöä voidaankin hallita pitkälti WordPressin omilla funktioilla. (WordPress Developer Resources 2015.)

Esimerkissä nähdään klassisen teeman Twenty Twenty One index-tiedoston koodi sisältäen ylätunnisteen, Loopin sekä alatunnisteen. Malli noudattaa tyypillistä järjestystä sisällön näyttämisessä. (Kuva 2.)

```php
get_header(); ?>

<?php if (
    is_home() && !is_front_page() && !empty (single_post_title('', false))
): ?>
    <header class="page-header alignwide">
        <h1 class="page-title">
            <?php single_post_title(); ?>
        </h1>
    </header><!-- .page-header -->
<?php endif; ?>

<?php
if (have_posts()) {

    // Load posts loop.
    while (have_posts()) {
        the_post();

        get_template_part(
            'template-parts/content/content',
            get_theme_mod('display_excerpt_or_full_post', 'excerpt')
        );
    }

    // Previous/next page navigation.
    twenty_twenty_one_the_posts_navigation();

} else {

    // If no content, include the "No posts found" template.
    get_template_part('template-parts/content/content-none');

}

get_footer();
```

Kuva 2. Klassisen teeman index.php.

Lohkoteemassa PHP ja funktiot on korvattu HTML-merkintäkielellä ja parametreilla oikeiden mallien tai osien hakemiseen. The Loopin korvaa The Query Loop, joka määrittää mitä julkaisuja ja lohkoja sivulla näytetään. Lohko toimii samalla tavalla kuin perinteinen The Loop, mutta mahdollistaa visuaalisen käyttöliittymän sisällön näyttämisen määrittämiseen. Lohkot tallennetaan HTML-kommentteihin, joiden avulla WordPress osaa hakea dataa dynaamisesti. (Alfredo Navas 2023.)

Esimerkissä nähdään lohkoteeman index.html, jonka rakenne poikkeaa merkittävästi klassisen teeman vastaavasta tiedostosta index.php. (Kuva 3.)

```
<!-- wp:query {"query":{"pages":0,"offset":0,"postType":"post","order":"desc",
    "orderBy":"date","author":"","search":"","exclude":[],"sticky":"",
    "inherit":true,"taxQuery":null,"parents":[]},"displayLayout":{"type":"flex",
    "columns":3},"align":"wide","layout":{"type":"default"}} -->
<div class="wp-block-query alignwide">
    <!-- wp:post-template {"align":"wide"} -->
        <!-- wp:post-featured-image {"isLink":true,"width":"100%",
            "height":"clamp(15vw, 30vh, 400px)","align":"wide"} /-->
        <!-- wp:post-title {"isLink":true,"align":"wide"} /-->
        <!-- wp:post-excerpt /-->
        <!-- wp:post-date {"isLink":true} /-->

        <!-- wp:spacer {"height":"var(--wp--preset--spacing--70)"} -->
        <div style="height:var(--wp--preset--spacing--70)" aria-hidden="true"
        class="wp-block-spacer"></div>
        <!-- /wp:spacer -->
    <!-- /wp:post-template -->

    <!-- wp:query-pagination {"paginationArrow":"arrow","align":"wide",
        "layout":{"type":"flex","justifyContent":"space-between"}} -->
        <!-- wp:query-pagination-previous /-->
        <!-- wp:query-pagination-next /-->
    <!-- /wp:query-pagination -->
</div>
<!-- /wp:query -->
```

Kuva 3. Lohkoteeman Twenty Twenty Three index.html ja The Query Loop.

3 Laatuun vaikuttavat tekijät

WordPress-teeman laatua voidaan tarkastella käyttäjäkokemuksen, turvallisuuden ja suorituskyvyn suhteen. Nykystandardien mukaan verkkosivujen tulee olla täysin responsiiviset, yhteensopivat eri selaimien kanssa, noudattaa saavutettavuuden vaatimuksia ja tukea eri kielille kääntämistä. Lisäksi hyvältä teemalta odotetaan hakukoneoptimointia, joka sisältää esimerkiksi HTML-elementtien oikeanlaisten merkintöjen käyttämisen, sekä nopeuden optimoinnin. (Elkatan 2023.)

Teknisestä näkökulmasta teeman koodin tulee olla hyvin dokumentoitua ja selkeää ja noudattaa kunkin ohjelmointi- tai merkintäkielen omia parhaita käytäntöjä. Koodin tulee myös olla ajantasaista ja käyttää ohjelmointi- ja merkintäkielten viimeisimpiä versioita. (Elkatan 2023).

Tässä luvussa käsitellään aiheita ja teknisiä seikkoja, jotka vaikuttavat teeman laatuun. Lähestyn asiaa tutkimalla millaisia toteutuksia ja ratkaisuja suositellaan ja pyrin löytämään perustelut suosituksille. Samalla tuon esille esimerkkejä joistakin suosituimmista, laadukkaina pidetyistä ilmaisista teemoista ja niiden toteutuksista.

3.1 Tematyypin valinta

Tutkimuksessa tematyypin vaikutuksesta teemojen käytettävyyteen ja virheiden määrään havaittiin, että valtaosa kokeeseen osallistuneista henkilöistä suoriutuivat annetuista tehtävistä paremmin lohkoteemaa käyttäessä. Kyselyn perusteella todettiin valtaosan osallistujista pitävän lohkoteemaa kokonaisuudessaan parempana ratkaisuna. (Joona Vesa 2022: 21-23.)

Lohkoteemoja pidetään WordPress-teemankehityksen tulevaisuutena niiden helpon muokattavuuden johdosta. Lisäksi lohkoteema voi tarjota nopeampia latausaikoja, sillä se vaatii usein vähemmän lisäosia kuin klassinen teema. Toisaalta lohkoteeman laaja muokattavuus voi olla tarpeetonta verkkosivujen

omistajalle ja klassinen teema voi tarjota parempaa ylläpidettävyyttä yksinkertaisilla mukautusvaihtoehdoilla. Lisäksi klassista teemaa kehittäessä voidaan päästä lohkoteeman kanssa samoihin latausaikoihin tekemällä räätälöityjä ratkaisuja ja ominaisuuksia kolmannen tahon lisäosien sijaan. (Ilyas 2024.)

3.2 Tekninen toteutus

3.2.1 Teeman tiedostot ja modulaarisuus

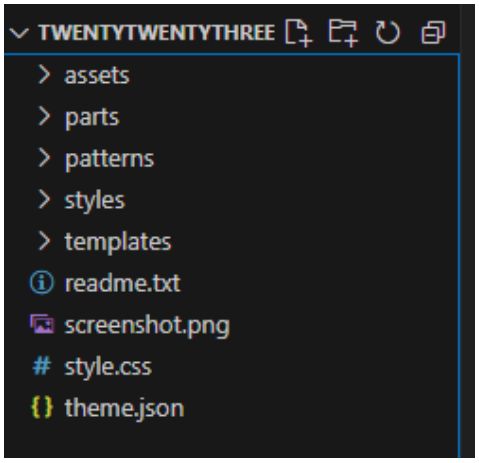
WordPress-teeman tiedostot sijaitsevat teemakansiossa, tyypillisesti useisiin alikansioihin jaettuna. Tiedostojen selkeä järjestys ja alikansioihin jakaminen tekee teeman toiminnallisuuden ymmärtämisestä helpompaa sekä ylläpitäjille että kehittäjille. Hyvin järjestetty teemakansio mahdollistaa myös paremman laadunvalvonnan.

Tyylitiedoston style.css lisäksi klassisen teeman index.php sijoitetaan juurikansioon, mutta muuten klassisissa teemoissa ei toimivuuden kannalta vaadita tiettyä kansiorakennetta. Yleisenä käytäntönä on sijoittaa teeman eri osat, kuten ylä- ja alatunnisteet sekä sivupalkit erilliseen kansioon. Sen sijaan lohkoteeman index.html sijoitetaan templates-kansioon, jotta WordPress tunnistaa teeman tyypin lohkoteemaksi. Muutkin mallit tulee sijoittaa samaan kansioon ja osat kuten ylä- ja alatunnisteet kansioon nimeltä parts, jotta WordPress osaa löytää ne. (WordPress Developer Resources 2014.)

Vaadittujen kansioiden lisäksi teemaan on usein eritelty kansio assets, joka yleensä sisältää esimerkiksi kansiot fonteille, CSS-tiedostoille, kuville tai muulle medialle, monikielisyydelle tai JavaScript-tiedostoille. WordPress Developer Resourcesin mukaan myös kansiot patterns ja styles ovat vakiokäytäntö lohkoteemoissa. Patterns-kansio sisältää teeman lohkot, joihin pääsee käsiksi WordPressin lohkoeditorin kautta. WordPress rekisteröi kansion sisällön automaattisesti kansion nimen perusteella. Styles sisältää lohkoteeman

määrittelyt ja variaatiot teeman tyyleille ja asetuksille. (WordPress Developer Resources 2014.)

WordPress Developer Resources (2014) suosittelee tarkastelemaan Twenty-sarjan teemoja kansiorakenteen oppimiseen. Esimerkiksi teema Twenty Twenty Three noudattaa esimerkillistä kansiorakennetta. (Kuva 4.)



Kuva 4. Teeman Twenty Twenty Three kansiorakenne.

Teeman käyttötarkoitukset ja laajuus vaikuttavat kansiorakenteen hallinnoimiseen. OceanWP on eräs suosittu ilmaisteema, joka on toteutettu klassisen teeman tyyliin PHP-pohjaisin mallein. OceanWP on laaja, moderni yleisteema, joka mahdollistaa hyvin monenlaisten verkkosivujen luomisen ja laajan räätälöinnin. OceanWP muun muassa tukee verkkokauppalisäosaa WooCommerce ja sisältää korvaavia mallitiedostoja sekä tyylitiedostoja lisäosan vakiomallien tilalle. Lisäksi teema sisältää lukuisia eriteltyjä tyylitiedostoja ja laajennuksia. Näin suuressa projektissa on tärkeää eritellä koodi pienempiin osiin hallittavuuden säilyttämiseksi. Vertailuna OceanWP:n teemakansio sisältää 138 alikansiota ja 1 035 tiedostoa, kun taas Twenty Twenty Threen teemakansio sisältää 10 alikansiota ja 50 tiedostoa.

Modulaarinen lähestymistapa on suositeltu käytäntö yleisesti verkkosivuohjelmoinnissa, eikä WordPress poikkea asiassa. Kun sivuston osat erotellaan pienempiin osiin eli moduuleihin, yksittäisten, pienempien osuuksien ja ominaisuuksien muokkaaminen ja päivittäminen on helpompaa ja

riskittömämpää. Kun mallin muokkaamiselle syntyy tarve, voidaan keskittyä pieneen osaan koodia ja loput teeman koodista säilyy turvassa mahdollisilta virheiltä. (Elkatan 2023.)

3.2.2 Koodi

WordPressille on määritelty omat koodausstandardit, jotta koodi on johdonmukaista ja helposti ymmärrettävää. Nämä standardit liittyvät koodin syntaksiin, koodin kommentointiin ja nimeämiskäytäntöihin. Johdonmukainen koodi helpottaa ylläpidettävyyttä ja kehittäjien yhteistyötä. Noudattamalla standardeja varmistetaan myös yhteensopivuus WordPressin ydinjärjestelmän sekä muiden lisäosien kanssa. Myös laadunvalvonta on helpompaa selkeän ja johdonmukaisen koodin kanssa. Standardit sisältävät myös suosituksia latausnopeuksien ja responsiivisuuden optimoimiseksi. Kun tarpeetonta koodia ei suoriteta turhaan, voidaan parantaa suorituskykyä. (WordPress Developer Resources 2019.)

Koodausstandardeja noudattavat teemat voivat saada paremman vastaanoton WordPress-yhteisössä, jonka ansiosta näkyvyys ja käyttäjäarviot voivat parantua. Standardit luovat pohjan toimivalle, turvalliselle ja nopealle teemalle, joten myös käyttäjäkokemus on parempi. (WordPress Developer Resources 2019.)

3.2.3 Mallihierarkia

WordPressin mallihierarkia määrittelee mikä teeman tiedosto ladataan käyttäjän tehdessä tietyn pyynnön. Hierarkia etenee yleisistä malleista tarkempiin malleihin. Jos käyttäjä pyytää esimerkiksi päästä etusivulle, WordPress tarkistaa ensin, onko teemassa mallia nimeltä front-page ja sen puuttuessa mallia home. Kummankin tai minkä tahansa muun mallin puuttuessa WordPress lataa aina mallin index, joka on pakollinen teeman toiminalle. Hierarkiaa noudattamalla hyödynnetään WordPressin tapaa hakea ja lukea mallitiedostoja

ja mahdollistetaan teeman yksityiskohtaisempi mukauttaminen. Mallipohjia voidaan luoda useille eri sivuille, mikä parantaa käyttäjäkokemusta ja käytettävyyttä. Optimoidut mallit voivat myös parantaa suorituskykyä ja hakukoneoptimointia. (Weston 2021.)

3.2.4 Optimointi ja saavutettavuus

Laadukas koodi, tiedostojen hallinta ja WordPressin mallihierarkian tehokas hyödyntäminen vaikuttavat verkkosivujen suorituskykyyn, mutta sitä voidaan parantaa muinkin tavoin. Tyylit ja komentosarjat tulee suorittaa WordPressin tapaan ja lisäksi CSS- ja JavaScript-tiedostot voidaan pienentää (minify). Pienentämiseen on olemassa automatisoituja työkaluja. Suorituskykyä voidaan parantaa myös käyttämällä sisällönjakeluverkkoa eli Content Delivery Networkia (CDN) pienentämään latausaikoja. Selaimen ja palvelimen välimuistin hyödyntäminen ja turhan datan poistaminen tietokannasta ovat myös suositeltuja käytäntöjä. (Elkatan 2023.)

Kuvat on hyvä optimoida pakkaamalla kuvat laatua hävittämättä. Tarkoitukseen on saatavilla esimerkiksi työpöytäsovelluksia, joissa kuvien käsittely verkkoa varten on automatisoitu (WordPress n.d.). Lisäksi lisäämällä laiska lataus (eng. lazy loading) kuviin, voidaan ladata kuvat vasta niiden tullessa näytölle. (Javaid 2024.)

Teeman hakukoneoptimointi koostuu muun muassa muista tutkimuksessa selvinneistä asioista, kuten laadukkaasta koodista, responsiivisuudesta sekä suorituskyvyn optimoinnilla. Lisäksi lisäämällä sisäisiä linkkejä teemaan, sivusto voidaan indeksoida hakukoneissa paremmin (WPBeginner 2024). Sosiaalisen median jakopainikkeet ja Open Graph -tunnisteet, jotka kontrolloivat kuinka verkkosivun osoite näytetään, tehostavat verkkosivujen sisällön tehokkaamman jakamisen eri alustoilla. (WPBeginner 2024).

Sivuston säilyminen käytettävänä ja saavutettavana kaikille käyttäjille on myös olennainen osa teeman laatua ja WordPress onkin määritellyt omat standardinsa saavutettavuudelle. WordPressin ydin on suunniteltu täyttämään

WCAG 2.0-ohjeistuksen AA-tason kriteerejä saavutettavuudeltaan. WordPress ei voi taata, että kaikki teemat noudattavat näitä kriteetejä, mutta jotta teemaa voi mainostaa täysin saavutettavana, sen tulee läpäistä WordPressin vaatimukset. (WordPress Codex n.d.)

Saavutettava teema luodaan käyttämällä semanttista HTML:ää, sillä se auttaa ruudunlukijoita ymmärtämään sivuston rakennetta. ARIA-tunnisteet lisäävät saavutettavuutta HTML-elementteihin, jotka eivät oletuksena ole saavutettavia. Näppäimistönavigoinnin tulee toimia vuorovaikutteisissa elementeissä eli esimerkiksi nappeja pitää pystyä käyttämään ilman hiirtä. Teeman värivalikoiman tulee olla kontrastiltaan riitävä, rakenteen ja erityisesti sivustolla navigoimisen tulee olla intuitiivista ja selkeää. (WordPress 2023.)

3.3 Responsiivisuus

Responsiivinen verkkosivu on välttämätön nykyaikaiselle nettisivustolle, sillä se mahdollistaa optimaalisen käyttökokemuksen eri laitteilla. WordPress-teeman responsiivisuus toteutetaan pääasiassa CSS:n mediakyselyillä (eng. media query). Tämä tarkoittaa sitä, että sivuston ulkoasu ja rakenne mukautuvat automaattisesti käyttäjän laitteen näytön koon ja resoluution mukaan. Esimerkiksi voit määrittää, että tietyt elementit näytetään tai piilotetaan riippuen siitä, kuinka suuri näyttö on. Responsiivisessa suunnittelussa sivuston rakenne pilkotaan erilaisiin osiin, jotka järjestäytyvät uudelleen näytön koon mukaan. Tämä varmistaa, että käyttäjä saa aina parhaan mahdollisen käyttökokemuksen riippumatta siitä käyttäänkö tämä verkkosivustoa tietokoneella tai mobiililaitteella. (Seán 2018.)

Kuvat ovat keskeinen osa monia verkkosivuja ja responsiivisessa suunnittelussa on tärkeää, että ne skaalautuvat oikein eri näyttökokoja varten. Tämä tarkoittaa usein sitä, että käytetään kuvan eri versioita eri näyttökoille, jotta sivusto latautuu nopeasti ja näyttää hyvältä kaikilla laitteilla. (Seán 2018.)

Koska pienillä näytöillä perinteinen vaakasuuntainen valikko ei välttämättä toimi hyvin, responsiivisissa teemoissa käytetään usein vaihtoehtoista valikkoa

mobiililaitteille. Tämä voi olla esimerkiksi hampurilaismenu, joka avautuu pystysuunnassa ja näyttää valikon vaihtoehdot käyttäjän napauttaessa sitä. (Seán 2018.)

3.4 Turvallisuus ja virhetilanteet

WordPress-teemassa suositellaan validoimaan ja puhdistamaan kaikki käyttäjän syötteet ennen niiden käyttöä, jotta SQL-injektiot ja muut haitalliset syötteet voidaan estää. WordPressissä on sisäänrakennettuja funktioita kuten esimerkiksi `esc_html()` ja `sanitize_text_field()`, joita suositellaan käyttäjän syötteisiin. (WordPress Developer Resources 2022.)

Teema ja kaikki kolmannen osapuolen kirjastot ja lisäosat tulee pitää päivitettyinä, jotta kaikki ajantasaiset tietoturvasuojaukset ovat käytössä. Verkkosivujen turvallisuutta suositellaan myös testaamaan säännöllisesti, jotta mahdolliset tietoturva-aukot pystytään havaitsemaan ajoissa. Lisäksi virheiden varalta suositellaan lokeja ja seurantaa, jotta niihin pystytään reagoimaan nopeasti. (Shah 2024.)

Nonce (number used once) on kertakäyttöinen, lomakkeissa ja URL-osoitteissa käytettävä tunniste, jolla estetään CSRF-hyökkäyksiä WordPressissä. Se liitetään tiettyyn pyyntöön ja varmistaa sen lähteen. Lomakkeissa nonce tarkistaa, että pyyntö on aito kun lomake lähetetään. URL-osoitteissa noncella voidaan tarkistaa, että linkkiä klikkaa käyttäjä, jolla on oikeat käyttäjäoikeudet. (Cap 2023.)

WordPress sisältää myös useita valmiita funktioita joita suositellaan käytettäviksi, sillä ne ovat valmiiksi suojattuja ja suunniteltu käytettäväksi WordPress-teeman kehityksessä. Niitä käyttämällä voidaan myös olla varmoja siitä, että koodi on yhteensopivaa WordPressin eri versioiden ja lisäosien kanssa. WordPressin funktiot päivittyvät automaattisesti uuden julkaisun mukana. (Perna 2017.)

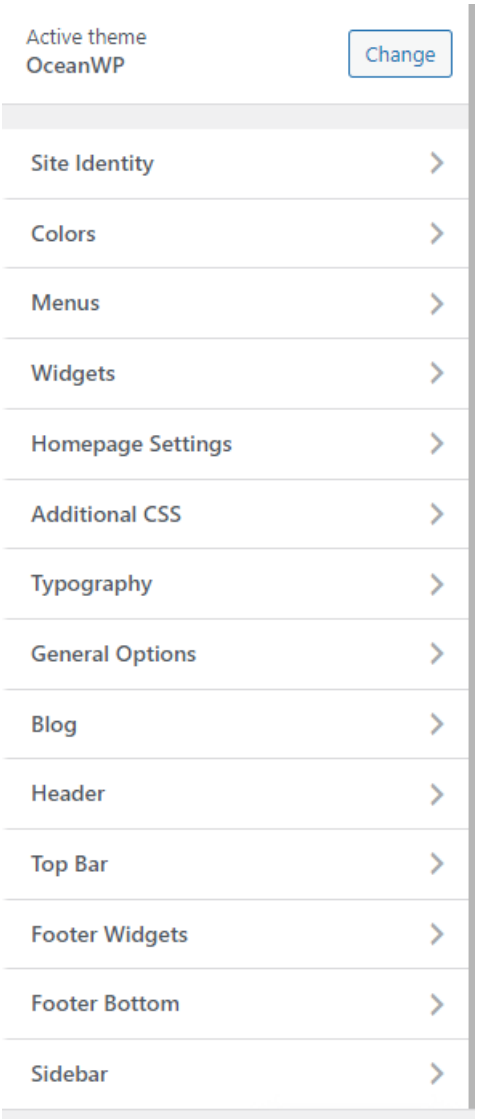
Virhetilanteisiin voi varautua luomalla teemaan mukautettu 404.php -tiedosto, sillä se voi tarjota käyttäjälle informaatiota ja navigaatiovaihtoehtoja tämän kohdatessa virhesivun. Lisäksi WordPress Codex suosittelee kertomaan, ettei virhe ollut käyttäjästä johtuva. Tämä pitää sivuston käyttäjäystävällisempänä, joka voi vähentää sivustolta poistumisia. Virhesivun tuleekin olla selkeä ja sisältää käyttäjää hyödyttävää informaatiota. Siihen voi sisällyttää hakuominaisuuden, linkkejä ja esimerkiksi mahdollisuuden raportoida ongelmasta. (WordPress Codex n.d.)

3.5 Muut laatuun vaikuttavat tekniset ratkaisut

3.5.1 Mukauttaminen ja teeman asetukset

WordPress-teeman laatuun vaikuttaa myös mukautettavuus, jota voidaan lisätä teeman asetuksilla, jotka tarjoavat mahdollisuuden säätää tiettyjä ominaisuuksia verkkosivujensa ulkoasussa ilman koodaamista (Elkatan 2023). Verkkosivujen omistajalle voidaan tarvittaessa antaa mahdollisuus esimerkiksi vaihtaa sivuston logoa tai ylätunnisteen kuvaa, sivupohjien asettelua tai lisätä omaa CSS:ää.

WordPressin suositus on käyttää Customizeria asetusten hallintaan. Sen avulla on mahdollista tarkastella muutoksia reaaliajassa. WordPress tarjoaa monia sisäänrakennettuja asetuksia, mutta lisäksi Customizeriin voi lisätä omia asetuksia. (WordPress Codex n.d.). Esimerkiksi teeman OceanWP Customizeriin on lisätty hurjasti mahdollisuuksia muokata erilaisia ominaisuuksia verkkosivujen ulkoasussa. (Kuva 5.)



Kuva 5. Esimerkki OceanWP:n Customizer-työkalusta.

Lisäksi sisällöntuottaja voi muokata verkkosivuja niin kutsutuilla vimpaimilla (eng. widgets). Ne ovat elementtejä sivuston ulkoasussa, joille rekisteröidään omat alueet tiedostossa functions.php. Vimpainalueille voi lisätä haluamiaan vimpaimia, kuten linkkejä tai mainoksia WordPressin hallintapaneelin kautta. Esimerkiksi sivupalkki on tyypillinen alue vimpaimille. (Newcomer 2024.)

3.5.2 Lokalisointi

Laadukas teema on lokalisoitu ja kaikki käännettävät merkkijonot on sisällytetty WordPressin sisäänrakennettuihin funktioihin kuten `__()` ja `_e()`. Funktioiden perusteella WordPress ymmärtää, että kyseiset merkkijonot ovat dynaamisesti käännettävissä. (Learn WordPress n.d.)

Lisäksi teemaan voi luoda POT-tiedoston (portable object template). Se ei ole pakollinen sivuston käännettävyyden kannalta, mutta se on vakiintunut tapa teemankehityksessä. Tiedosto sisältää kaikki teeman käännettävät merkkijonot ja toimii mallina kääntäjille, kun luodaan lokalisoituja kielitiedostoja. Kielitiedostojen luomiseen on työkaluja, jotka luovat automaattisia käännöksiä halutulle kielelle, mutta vastineet merkkijonoille voi luoda myös itse. (Learn WordPress n.d.)

4 Parhaiden käytäntöjen soveltaminen

Tässä luvussa toteutetaan parhaita ratkaisuja soveltaen niitä omaan teemaan. Teema luodaan harjoitustyönä Turun ammattikorkeakoulun projektitoimisto theFIRMAlle ja otetaan käyttöön verkkosivuilla, jotka sisältävät sivuja ja julkaisuja. Lisäksi teemalle on luotu perustyylit ja asettelut.

Teeman tulee noudattaa tässä opinnäytetyössä läpikäytyjä parhaita käytäntöjä niiltä osin, jotka ovat olennaisia harjoitustyössä. Valituilla sovellutuksilla pyritään luomaan vahva pohja teeman laadukkaaseen jatkotyöstämiseen.

Harjoitustyön teema toteutetaan hybridinä eli klassisena teemana, joka sisältää lohkoteemalle tyypillisiä ominaisuuksia. Suuri muokattavuus WordPress-editorissa ei ole tarpeen verkkosivuilla, mutta opetuksen kannalta lohkoteeman ominaisuuksien ja esimerkkien sisällyttäminen on aiheellista. Lohkoteeman toteutukset eivät kuitenkaan ole vielä ajankohtaisia harjoitustyössä, joten opinnäytetyössä näytettävät esimerkit noudattavat klassisen teeman mallia.

4.1 Teemakansio ja koodi

Teema toteutetaan parhaiden käytäntöjen mukaan noudattaen WordPressin mallihierarkiaa ja koodausstandardeja. Kansiorakenne tarjoaa mahdollisuuden työn loogiselle laajentamiselle, joten esimerkiksi kuvat, käännöstiedostot ja JavaScript-tiedostot sijoitetaan omiin alikansioihinsa.

Kommentteja lisätään koodiin kohtiin, joissa ne voivat tarjota tärkeää informaatiota teeman tai sivun toiminnasta. Lisäksi käytetään ohjelmointikielien uusimpia versioita kuten HTML5 ja CSS3.

4.2 Responsiivisuuden toteuttaminen

HTML5:n myötä voidaan hyödyntää niin kutsuttua meta-tagia, jonka avulla selain osaa skaalata verkkosivut nykystandardien mukaan. Ensin `meta`

`name=”viewport”` määrittää, että verkkosivujen siällön tulee skaalautua näyttöikkunan mukaan, sitten sivun sisällön skaalautuminen laitteen näytön leveyden mukaan määritellään koodilla `width=device-width` ja `initial-scale` määrittelee millä mittasuhteilla sivu ladataan.

Ohjelma 1. HTML5 ja meta-tagin hyödyntäminen responsiivisuudessa.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Koska ylätunnisteen malli luetaan ensimmäisenä, asetetaan kyseinen koodinpätkä ylätunnisteen malliin.

4.2.1 Sisällön järjestäminen

Jotta sivun sisältö järjestäytyisi uudelleen laitteen ruutukoon mukaan, käytetään mediakyselyjä jotka kirjoitetaan tyylitiedostoon style.css. Esimerkkiteeman tapauksessa sivupalkki ja sivun otsake asettuvat huonosti ensimmäisen kerran kun ruutu on alle 650 pikseliä. Koska CSS luetaan ylhäältä alas, mediakyselyt tulevat viimeiseksi, yleisten määrittelyjen jälkeen. `@media only screen` tarkoittaa, että kyselyn sisältämät tyylit pätevät vain näytöllä, ei tulostettaessa. Sitten määritellään niin kutsuttu hajoamispiste, eli leveys, jossa verkkosivujen sisältö alkaa asettumaan huonosti ruutua pienennettäessä. Tämän jälkeen annetaan uudet tyylimäärittelyt aaltosulkeitten sisään.

Ohjelma 2. Mediakyselyt CSS-tyylitiedostossa.

```
/* Media query for screens under 650px in width */


@media only screen and (max-width: 650px) {
    #site-header {
        background: url('header-img-sm.jpg') no-repeat right;
        height: 200px;
```

```
        margin: 0 auto; /* Short for margin-right: auto; margin-left:
auto; */

    }

}
```

4.2.2 Navigaatiopalkki

Muun sisällön lisäksi navigaatiopalkki on monesti osa, joka kärsii ruutukoon pienentymisestä. Isolla näytöllä navigaatiopalkin sisältö asettuu yleensä riville kun taas mobiililaitteen ruudulla niin sanottu hampurilaismalli, jossa sisältö asettuu allekkain ja näkyy vain käyttäjän klikatessa menun auki, on yleisesti käytetty malli. Menua edustaa kolme viivaa, jotka näkyvät menun ensimmäisenä objektina vain kun ruutu on alle määritetyn koon. (Seán 2018).

Esimerkissäni navigaatiopalkin objektit järjestyvät kahdelle riville ruutukoon ollessa hieman alle 400 pikseliä, joten kirjoitetaan erillinen media query vaihtoehtoiselle menurakenteelle ja hienosäädetään myös otsakkeen mittasuhteita. Menun sisältö piilotetaan oletuksena ja järjestetään allekkain CSS:n avulla, mutta tarvitaan JavaScriptia, tarkemmin sanottuna kirjastoa jQuery, kun sisältö halutaan esille klikkaamalla.

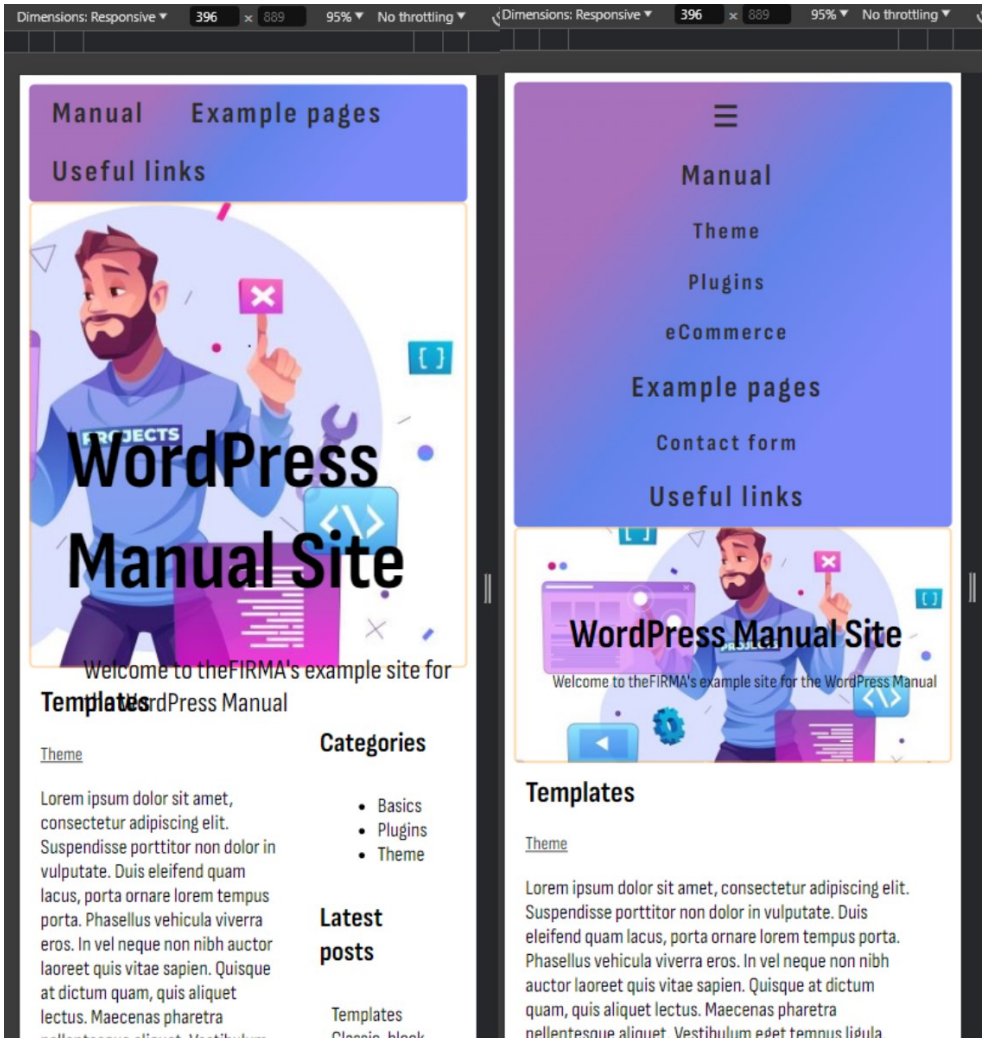
Ohjelma 3. Ohjelmakoodi mobiilimenun avaamiseen klikkaamalla.

```
//jQuery to open menu on click


jQuery(document).ready(function($) {

    // Toggle menu items on click

    $('#top-nav > div > ul > li:first-child').on('click', function() {

        $('#top-nav > div > ul > li:not(:first-
child)').slideToggle(400);

    });
```

```
    // Adjust menu display on window resize
    $(window).resize(function() {
        if ($(window).width() > 650) {
            $('#top-nav > div > ul > li:not(:first-child)').show();
        } else {
            $('#top-nav > div > ul > li:not(:first-child)').hide();
        }
    }).resize(); // Trigger the resize event on load to set the
initial state
});
```

Koodia varten luodaan oma JavaScript-tiedosto, joka otetaan käyttöön functions.php-tiedostossa.

Kyseisten osien koodiin lisäämisen ja tyylien hienosäädön jälkeen verkkosivuilla saavutetaan responsiivisuus ja sisältö pysyy luettavana myös ruudun ollessa pieni. Esimerkissä nähdään vasemmalla miltä sisältö näytti ennen ja oikealla miltä se näyttää responsiivisuuden luomisen jälkeen kun ruutukoko on hieman alle 400 pikseliä. (Kuva 6.)



Kuva 6. Verkkosivut ennen ja jälkeen responsiivisuuden lisäämistä.

Menu, joka on esimerkissä auki klikattuna, järjestäytyy siististi allekkain, otsake sekä otsakekuva pienenevät ruutukokoon sopivaksi ja sivupalkki on siirtynyt sivun varsinaisen sisällön alle.

4.3 Turvallisuus ja virhetilanteet

Kaiken dynaamisen datan asianmukainen turvaaminen suojaa teemaa XSS-hyökkäyksiltä. Kaikkiin mallitiedostoihin lisätään turvallisuutta lisäävät funktiot dynaamisen datan ja käyttäjän syötteiden käsittelyyn. Esimerkiksi index-tiedoston Loopissa on nähtävissä WordPressin oma funktio wp_kses_post(), joka varmistaa HTML-tunnisteiden turvallisuuden ja poistaa haitallista sisältöä

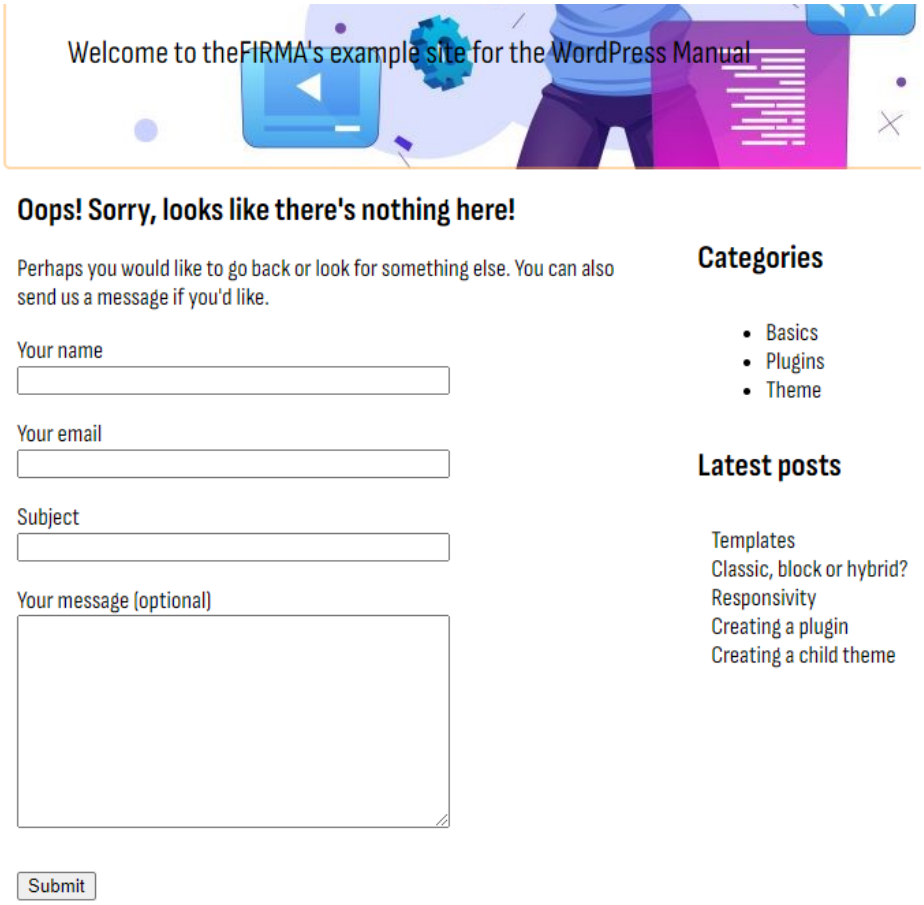
käyttäjän syötteistä ennen sivulle tulostamista. Funktio esc_html_e() suojaa tekstin turvallista näyttämistä HTML-sivulla ja tekee tekstistä käännettävän. (WordPress Developer Resources 2024.)

Ohjelma 4. Turvallisuus ja WordPressin sisäänrakennetut funktiot koodissa.

```
<?php

        <?php if (have_posts()): ?>

            <?php while (have_posts()): the_post(); ?>

                <article <?php post_class(); ?>>

                    <h2><a href="<?php the_permalink(); ?>"><?php
the_title(); ?></a></h2>

                    <p class="category"><?php echo
wp_kses_post(get_the_category_list(', ')); ?></p>

                    <?php the_excerpt(); ?>

                </article>

            <?php endwhile; ?>

        <?php else: ?>

            <p><?php esc_html_e('No posts found', 'WP manual theme');
?></p>

        <?php endif; ?>
```

Lisäksi malleissa hyödynnetään WordPressin omia funktioita kuten get_header() ja have_posts() parhaiden käytäntöjen mukaan.

Teemaan luodaan myös malli 404.php siltä varalta, että käyttäjä päätyy sivulle, jota ei ole olemassa. Tällöin käyttäjälle esitetään ystävällinen viesti ja tarjotaan mahdollisuutta palata takaisin, etsiä sisältöä tai lähettää viesti upotetun lomakkeen kautta. (Kuva 7.)



Kuva 7. 404-sivu.

Lomake on luotu käyttämällä lisäosaa Contact Form 7, jolla on omat turvallisuus-asetuksensa, eikä käyttäjän syötteisiin siis tarvita erillistä koodia. Mallin viestit tulostetaan näytölle käyttämällä funktiota esc_html_e() ja koodissa hyödynnetään jälleen WordPressin omia, suojattuja funktioita.

4.4 Muut toteutukset

Opinnäytetyössä käytetyn teeman työstö jatkuu vielä pitkään opinnäytetyön esimerkkien jälkeen. Teeman laajentuessa ja ominaisuuksien lisääntyessä pyritään noudattamaan parhaita suosituksia ja perustelemaan ne myös muille teeman käyttäjille.

Koodissa noudatetaan saavutettavuuden vaatimuksia, mutta sivuston saavutettavuustestaus toteutetaan myöhemmin. Tästä syystä saavutettavuus jää käsiteltävien aiheiden ulkopuolelle sovellettavissa toteutuksissa.

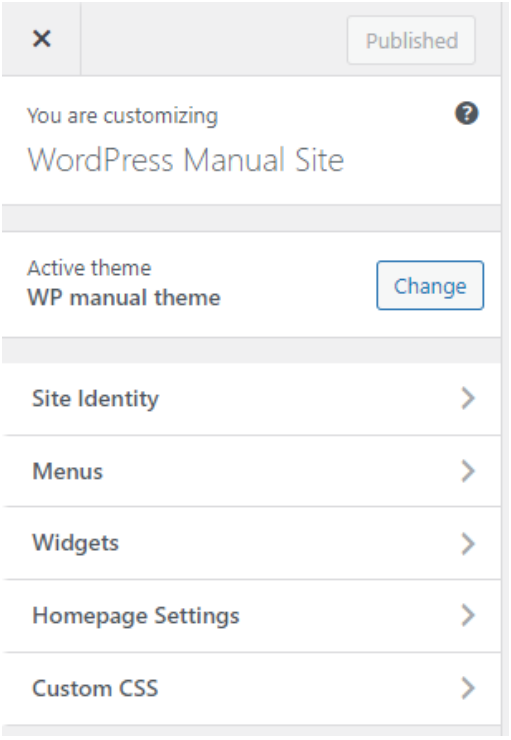
4.4.1 Mukauttaminen

Esimerkkiteemassa ei ole tarkoituksenmukaista antaa sisällöntuottajalle loputtomasti mukauttamisvaihtoehtoja, sillä tarkoitus on demonstroida sivuston taustalla toimivaa teeman koodia. Customizer on kuitenkin olennainen osa laadukasta teemaa, joten se on hyvä ottaa käyttöön. Customizer ei ole oletuksena käytössä, joten se pitää ensin rekisteröidä teeman tiedostossa functions.php seuraavalla koodinpätkällä:

Ohjelma 5. Ohjelmakoodi WordPress Customizerin aktivoimiseen.

```
function manual_customize_register( $wp_customize ) {

    //All sections, settings and controls here

}

add_action( 'customize_register', 'manual_customize_register' );


?>
```

Nyt Customizer on aktiivinen verkkosivuilla ja omia asetuksia voidaan lisätä omiin välilehtiinsä. (Kuva 8.)



Kuva 8. Customizer teemassa.

4.4.2 Käännettävyys

Lisäksi teemaan luodaan .POT-tiedosto käännöksiä varten. Tiedoston luomiseen käytettiin ilmaista kolmannen osapuolen työkalua qSandbox. Teemakansio pakataan ZIP-tiedostoksi, ladataan työkaluun ja qSandbox luo käännöstiedoston automaattisesti. Tiedosto sisältää kaikki teemassa olevat merkkijonot sekä niiden sijainnit tiedostoissa. Kaikissa käännettävissä merkkijonoissa käytetään käännösfunktioita, joten työkalu osaa tunnistaa ne koodissa. Merkkijonot voidaan sitten kääntää valituille kielille käyttämällä kääntäjäohjelmaa kuten poEdit, joka luo käännökset .POT-tiedoston pohjalta. Esimerkkimalliin luotiin kääntäjäohjelmalla tiedosto, joka sisältää suomenkieliset vastineet käännöstä vaativille merkkijonoille. (Kuva 9).

```
#: 404.php:16
msgid "Oops! Sorry, looks like there's nothing here!"
msgstr "Hups! Pahoittelut, täällä ei näytä olevan mitään!"

#: 404.php:17
msgid "Perhaps you would like to go back or look for something else?"
msgstr "Ehkäpä haluat palata tai etsiä jotakin muuta?"

#: 404.php:20
msgid "If you'd like, you can also send us a message with the following form."
msgstr ""
"Halutessasi voit myös lähettää meille viestin seuraavalla lomakkeella."
```

Kuva 9. Käännöstiedosto .POT.

5 Johtopäätökset

Työn tavoite oli saavuttaa ymmärrys teknisistä ratkaisuista, joita suositellaan parhaina käytäntöinä WordPress-teeman laadun parantamiseksi. Tarkoitus oli selvittää perustelut suosituksille ja kuinka ratkaisuja sovelletaan käytännössä. Työn tuloksista näkee, että tekniseen laatuun vaikuttaa erittäin laaja kokonaisuus, jonka osa-alueet vaikuttavat toisiinsa. Standardeja ja suosituksia noudattamalla saadaan kuitenkin aikaan toimiva teema ja taataan sen yhteensopivuus WordPressin ydinkoodin ja lisäosien kanssa.

Parhaisiin käytäntöihin tutustuttiin työssä yleisellä tasolla. Laatuun kokonaisuutena vaikuttavista asioista saavutettiin ymmärrys, mutta yksittäiset aiheet, kuten esimerkiksi turvallisuus, vaativat kattavampaa perehtymistä. Koska suositusten perustelut selvitettiin työssä, myös parhaiden teknisten ratkaisujen soveltaminen omassa teemassa oli perusteltua. Teknisesti laadukkaan teeman pohjatyö oli melko työläs, mutta suositusten noudattamisesta on hyötyä.

Teknisten ratkaisujen soveltaminen omassa teemassa käsitti vain olennaisimmat aiheet harjoitustyön kannalta. Teematiedostojen ollessa järjestyksessä ja koodin noudattaessa loogista rakennetta, teeman työstö on johdonmukaista. Virheet koodissa löytyivät helposti selkeän tiedosto- ja koodirakenteen ansiosta ja standardien noudattamisen johdosta vertailukohtia ja apua ongelmatilanteisiin oli helppo löytää WordPress-yhteisöstä. Merkkijonojen valmistelu dynaamista kääntämistä varten vaatii oikeellisten funktioiden lisäämistä, mutta lisäämisen jälkeen käännöstiedoston luominen on vaivatonta ja voidaan toistaa lukuisia kertoja eri kielille.

Tutkimuksen tuloksia voidaan hyödyntää WordPress-teemankehityksen oppimisessa ja teknisen laadun ymmärtämisessä. Tutkimuksessa selviää keskeisimmät teeman laatuun vaikuttavat tekniset ratkaisut ja työtavat, jotka tulee huomioida menestyksekkään teeman luomisessa käyttötarkoituksesta riippumatta. Tutkimuksessa tuodaan esille myös perustelut suositeltujen metodien noudattamiseen.

Työtä voisi jatkaa tutkimalla syvemmin eri teknisen laadun osa-alueita, kuten turvallisuutta, optimointia tai saavutettavuutta. Lisäksi voisi tutkia aiheita kuten Node.js ja React ja niiden hyödyntäminen teemankehityksessä sekä niiden vaikutuksia tekniseen laatuun. Node.js sallii JavaScript-koodin suorittamisen palvelimella ja esimerkiksi täysin räätälöityjen lohkojen luomisen. React taas on JavaScript-kirjasto, jonka avulla luodaan moderneja käyttöliittymiä verkkosivuille. Näiden teknologioiden hyödyntäminen voi monipuolistaa teemankehitystä ja tarjota lisää mukautusvaihtoehtoja.

Lähteet

Cap M. 2023. Understand and use WordPress nonces properly. Blogi. Viitattu 11.4.2024. https://developer.wordpress.org/news/2023/08/01/understand-and-use-wordpress-nonces-properly/

de Valk J. 2011. Anatomy Of A WordPress Theme. Blogi. Viitattu 11.4.2024. https://yoast.com/wordpress-theme-anatomy/

Elkatan M. 2023. 28 best practices for WordPress theme development. Blogi. Viitattu 29.4.2024. https://dev.to/menemelkatan/28-best-practices-for-wordpress-theme-development-5d2m

Ilyas M. 2024. Block Themes vs Classic Themes in WordPress – What You Need to Know. Blogi. Viitattu 29.4.2024. https://codup.co/blog/block-themes-vs-classic-themes-in-wordpress/

Javaid S. 2024. Why You Need WordPress Lazy Load to Speed up a Web Page? Verkkoaineisto. Viitattu 29.4.2024. https://www.cloudways.com/blog/wordpress-lazy-load/

Learn WordPress. 2023. Accessibility. Verkkoaineisto. Viitattu 29.4.2024. https://wordpress.org/about/accessibility/

Learn WordPress. N.d. Common APIs – Internationalization. Verkkoaineisto. Viitattu 29.4.2024. https://learn.wordpress.org/tutorial/common-apis-internationalization/

Mahfuz M 2024. The Power of Custom WordPress Theme Development: Benefits you Can’t Ignore. Blogi. Viitattu 11.4.2024. https://www.linkedin.com/pulse/power-custom-wordpress-theme-development-benefits-you-mahfuz-m-hozkc/

Navas A. 2023. Switching from Classic to Block-Based Themes in WordPress. Blogi. Viitattu 11.4.2024. https://webdevstudios.com/2023/12/12/switching-from-classic-to-block-based-themes-in-wordpress/

Nazaryan T. 2024. How to Add JavaScript to WordPress: A Simple Guide for Beginners. Verkkoaineisto. Viitattu 29.4.2024. https://10web.io/blog/how-to-add-javascript-to-wordpress/

Newcomer C. 2024. How to Add and Use WordPress Widgets: A complete guide. Blogi. Viitattu 29.4.2024. https://blog.hubspot.com/website/add-widget-to-wordpress-page

Perna Maria Antonietta. 2017. Coding Safe Themes with Built-In WordPress Functions. Verkkoaineisto. Viitattu 11.4.2024. https://www.sitepoint.com/safe-wordpress-themes-validation-escaping-functions/

Seán. 2018. Media Queries in WordPress. Blogi. Viitattu 11.4.2024. https://wordpressforgood.com/wordpress-media-queries/

Shah J. 2024. Complete WordPress Security Guide: An Approach to Safeguard Your Website. Viitattu 11.4.2024. https://wpwebinfotech.com/blog/wordpress-security-guide/#top-tips-for-word-press-security

Vesa J. 2022. WordPress Teematyypin vaikutus teemojen käytettävyyteen. Opinnäytetyö. Viitattu: 29.4.2024. https://www.theseus.fi/handle/10024/787498

Weston I. 2021. The WordPress template hierarchy explained. Verkkoaineisto. Viitattu 29.4.2024. https://www.easywp.com/blog/wordpress-template-hierarchy-explained/

WordPress. N.d. Optimize Your Images. Verkkoaineisto. Viitattu 29.4.2024. https://wordpress.com/support/media/image-optimization/

WordPress. N.d. Theme Accessibility. Verkkoaineisto. Viitattu 29.4.2024. https://make.wordpress.org/accessibility/handbook/which-themes-can-you-use/

WordPress Codex. N.d. Why WordPress Themes. Verkkoaineisto. Viitattu 11.4.2024. https://codex.wordpress.org/Theme_Development

WordPress Developer Resources. 2023. Core Concepts. Verkkoaineisto. Viitattu 29.4.2024. https://developer.wordpress.org/themes/core-concepts/

WordPress Developer Resources. 2024. Security. Verkkoaineisto. Viitattu 29.4.2024. https://developer.wordpress.org/apis/security/

WordPress Developer Resources. 2014. What Is a Theme? Verkkoaineisto. Viitattu: 9.4.2024. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/

WordPress Developer Resources. 2019. Wordpress Coding Standards. Verkkoaineisto. Viitattu 29.4.2024. https://developer.wordpress.org/coding-standards/wordpress-coding-standards/

WPBeginner. 2024. Internal Linking for SEO: The Ultimate Guide of Best Practices. Viitattu 29.4.2024. https://www.wpbeginner.com/wp-tutorials/internal-linking-for-seo-ultimate-guide-best-practices/

WPBeginner. 2024. How to Add Facebook Open Grap Meta Data in WordPress Themes. Viitattu 29.4.2024. https://www.wpbeginner.com/wp-themes/how-to-add-facebook-open-graph-meta-data-in-wordpress-themes/