Tampereen ammattikorkeakoulu

Skaalautuva ja helppokäyttöinen WordPress-teema Gutenbergillä ja ACF:llä

Paavo Tuomensalo-Porkka

OPINNÄYTETYÖ
Marraskuu 2020

Tietojenkäsittelyn tutkinto-ohjelma
Web-palvelut

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Web-palvelut

TUOMENSALO-PORKKA, PAAVO:
Skaalautuva ja helppokäyttöinen WordPress-teema Gutenbergillä ja ACF:llä

Opinnäytetyö 24 sivua, joista liitteitä 0 sivua
Marraskuu 2020

Opinnäytetyön aiheena oli WordPress-teeman kehitys. Toimeksiantajana toimi Markkinointitoimisto Bermuda Oy, jonka WordPress-pohjateemaa opinnäytetyössä kehitettiin. Opinnäytetyön tarkoituksena oli kehittää ja dokumentoida Bermudan nykyistä WordPress-pohjateemaa helpommin käytettäväksi ja nopeammaksi muun muassa sivujen prototyyppien toteutuksessa. Opinnäytetyön tavoitteena oli tutkia ja selvittää erilaisia työskentelytapoja ja -käytäntöjä Bermudan pohjateeman nopeampaan ja selkeämpään kehittämiseen. Teemankehitystä varten vertailtiin erilaisia avoimen lähdekoodin lisenssillä lisensoituja suosittuja WordPress-teemoja, ja käytettiin niitä pohjana uuden teeman toteutuksessa. Opinnäytetyön toteutus tehtiin ketterästi viiden viikon pituisen kehitysjakson aikana. Jokaisen kehitysjakson välissä tarkastettiin verkkokehitystiimin kanssa teemankehityksen eteneminen ja selvitettiin ilmenneitä ongelmia.

Opinnäytetyön lopputuloksena syntyi uusi pohjateema, joka ei kuitenkaan ole täysin valmis, vaan aloitus jatkuvalle teemankehitystyölle. Suurimpia muutoksia pohjateemaan olivat Gutenberg-editorin alkuperäisten ydinlohkojen käyttöönotto ja niiden tyylien mukauttaminen vanhan teeman tyylien kanssa, Gulp.js-tiedoston uusiminen, Ajax Load More -lisäosan lisääminen pohjateemaan sekä vanhan teeman tiedostorakenteen muuttaminen.

Uusi pohjateema antoi mahdollisuudet nopeammalle työskentelylle sivustojen toteutuksen aloitusvaiheessa. Opinnäytetyön toteutuksen lopputulos oli itsessään toimiva pohjateema, mutta sen kehitys jatkuu vielä tulevaisuudessakin. Siksi toteutuksen aikana kerättiin listaan jatkokehitysideoita, joilla pohjateeman kehitystä voidaan jatkaa. Pohjateeman jatkokehitykseen syntyneet ideat olivat lähinnä uusia toimintoja tai käytettyjen työkalujen uusien, ei vielä julkaistujen, versioiden lisäämistä pohjateemaan.

Asiasanat: wordpress, teemankehitys, gutenberg-editori

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Web Services

Tuomensalo-Porkka, Paavo:
Scalable and user-friendly WordPress theme with Gutenberg and ACF

Bachelor's thesis 24 pages, appendices 0 pages
November 2020

The objective of the thesis was to collect information on better ways to develop WordPress themes faster and easier, so that less time would be spent at the beginning of a website project. The purpose of the thesis was to develop and document the current WordPress starter theme, which the client company Bermuda’s web development team used into a faster and more user-friendly theme.

This study was carried out as a project. The project was split into five weeklong development periods. The data that was used in the theme development was collected by benchmarking different popular WordPress themes. The project was supervised by the web development team and their suggestions and comments were taken into account.

The results suggest that the new theme, which was created, can be used as a new starter theme for the web developer team, and will be tested and be further developed to better suit the needs of the team. The changes that were made to the old theme were functional and made a difference when working with the theme.

The findings indicate that the new theme is an operational WordPress starter theme and will be used in new client projects. However, the theme will not permanently remain as it is currently, because the environment around the theme changes constantly, and so the theme also needs to be constantly developed in order to be considered as a functional theme.

Key words: wordpress, theme development, gutenberg editor

SISÄLLYS

LYHENTEET JA TERMIT

B2B	Toisille yrityksille suunnattua markkinointia
Babel.js	Ilmainen avoimen lähdekoodin Javascript-kääntäjä, joka kääntää uudempaa Javascript-koodia yhteensopivammaksi, vanhemmaksi Javascript-koodiksi, jota vanhemmat selaimet ymmärtävät
Bootstrap	Ilmainen avoimen lähdekoodin CSS-kehys, joka sisältää CSS- ja Javascript-tiedostoja responsiivisen ja mobiililähtöisten verkkosivujen luontiin
Gulp.js	Avoimen lähdekoodin Javascript-pohjainen työkalupaketti verkkokehityksen automatisointiin
PHP	PHP: Hypertext Preprocessor, suosituin tekniikka dynaamisten web-palveluiden tuottamisessa
Repositorio	Tietorakenne, joka sisältää metatietoa tiedostokokonaisuudesta, kuten esimerkiksi muutoslokin.
SCSS	Sassy Cascading Style Sheets, esiprosessorin komentosarjakieli, joka tulkitaan CSS-muotoon

1 JOHDANTO

Opinnäytetyö liittyi Markkinointitoimisto Bermudan oman WordPress-pohjateeman päivittämiseen. Opinnäytetyön oli päivittää markkinointitoimisto Bermudan WordPress-pohjateema siten, että yrityksessä pystytään helpommin ja nopeammin luomaan toimivia verkkosivuja. Siten yrityksessä jää enemmän aikaa verkkosivujen viimeistelyyn ja kehittämiseen.

Opinnäytetyön aihepiirinä toimi siis WordPress-teeman kehitys. Työn tarkoituksena oli kehittää ja dokumentoida Bermudan nykyistä WordPress-pohjateemaa helpommin käytettäväksi ja nopeammaksi esimerkiksi sivujen prototyyppien toteutuksessa, sekä luoda WordPressin ja pohjateeman käyttöön yleiset ohjeet Bermudan asiakkaiden käytettäväksi.

Toimeksiantajana opinnäytetyöllä toimi tamperelainen Markkinointitoimisto Bermuda Oy, jonka web-kehittäjä-tiimissä työskentelen itsekin. Bermuda on erityisesti B2B-, teollisuus- ja teknologia-aloihin erikoistunut markkinointiviestinnän ja palvelusuunnittelun strateginen digi- ja mainostoimisto. Bermudalla työskentelee noin 20 vakituista työntekijää sekä muutama freelancer. Itse siirryin opinnäytetyöprosessin aikana freelancerista osa-aikaiseksi web-kehittäjäksi.

Opinnäytetyön tavoitteena oli tutkia ja selvittää erilaisia työskentelytapoja ja -käytäntöjä, joilla Bermudan verkkosivustoprojekteissa käytettävää pohjateemaa voidaan kehittää nopeammin ja selkeämmin. Tavoitteena oli myös projektien toteutusten virtaviivaistaminen.

Työ toteutettiin ketteränä projektina, joka jaettiin viiteen viikon pituiseen kehitysjaksoon. Jokaiselle viikolle määritettiin omat tavoitteet, joiden pohjalta voitiin seurata työn edistymistä, ja mahdollisiin ongelmiin pystyttiin reagoimaan nopeammin. Työ aloitettiin tekemällä esikuva-analyysia erilaisten suosittujen WordPress-teemojen välillä. Esikuva-analyysista saatujen tulosten pohjalta alettiin suunnitella ja toteuttaa oman pohjateeman kehittämistä.

Suurimmat muutokset, joita pohjateemaan toteutettiin, olivat WordPressin omien lohkojen käyttöönotto, sivuston automatisoinnin parantaminen ja teeman yleisen tyylin kohentaminen. Lopputuotoksen toimivuutta ja sopivuutta web-tiimin tarpeisiin testattiin heti tämän opinnäytetyön toteutuksen jälkeen.

Opinnäytetyön tuloksia hyödyntävät Bermudan oma kehitystiimi sekä graafikot ja konseptisuunnittelijat. Tavoitteena on myöhemmin lisensoida lopputuotos eli pohjateema vapaan lähdekoodin lisenssillä ja julkaista se julkiseen käyttöön esimerkiksi GitHubissa, mutta on osa teeman jatkokehitystä.

2 TYÖKALUT JA TEKNIIKAT

2.1 Käytetyt työkalut ja tekniikat

Pohjateeman kehityksessä käytettiin tekstieditorina Microsoftin Visual Studio Codea, ohjelmointikielinä PHP:tä, Javascriptiä, CSS:n esiprosessoria SCSS:ää ja merkintäkielenä HTML:ää. Pohjateemaan on myös sisällytetty Bootstrap 4.0 tiedostot auttamaan sivustojen tyylittelyssä. Paikallinen kehitysympäristö pystytettiin käyttämällä Flywheel-nimisen yhtiön Local-ohjelmaa. Pohjateema toimii alustana WordPressillä toteutettaville sivustoille ja antaa verkkosivuille perustoiminnallisuutensa ja ulkonäön. Bermuda käyttää WordPressiä muiden julkaisujärjestelmien sijasta, koska se tarjoaa monipuolisimmat mahdollisuudet toiminnallisuuden laajentamiseen suuren lisäosa-kirjastonsa ja teemavalikoiman avulla.

2.2 Gutenberg

Nykyään Classic-editorina tunnetun editorin tilalle tuli WordPress 5.0 -versiossa vuoden 2018 joulukuussa Gutenberg-niminen editori, joka mullisti WordPress-sivujen luonnin kokonaan. Gutenberg keskittyi enemmän lohkotyyliseen sivun rakentamiseen ja antoi vapaammat kädet kaikille luoda omia verkkosivujaan helpommin ilman suurta ohjelmointiosaamista.

Editorin vaihtaminen vanhasta uuteen ei kuitenkaan sujunut helposti, vaan Gutenberg-editori oli varsinkin alkuvaiheessa täynnä ohjelmointivirheitä ja epävakaa, jonka takia sen vaihtamisesta WordPressin oletuseditoriksi valitettiin paljon. Uuden editorin vaikeasta alusta on kuitenkin kulunut jo melkein kaksi vuotta, ja säännöllisten päivitysten ansiosta siitä on kehittynyt erittäin varteenotettava sisältömuokkain.

2.3 Lisäosat

Bermudan sivustoprojekteissa käytetään joitain Wordpressin suuren lisäosakirjaston lisäosia. Näitä ovat muun muassa Elliot Condonin rakentama Advanced Custom Fields -lisäosa sekä Darren Cooneyn tekemä WordPress Infinite Scroll – Ajax Load more -lisäosa.

Advanced Custom Fields (myöhemmin ACF) mahdollistaa helpon ja nopean tavan luoda räätälöityjä lisäkenttiä esimerkiksi WordPress-sivujen tai -artikkeleiden hallintapuolelle. Näiden avulla sivujen hallinnasta tehdään helpompaa ja vaivattomampaa loppukäyttäjille. ACF:stä on saatavilla myös maksullinen PRO-versio, joka tarjoaa ostajilleen suuremman valikoiman erilaisia lisäkenttiä sekä mahdollisuuden luoda kokonaan uusia lohkoja paljon yksinkertaisemmalla ja nopeammalla tavalla kuin mitä WordPressin alkuperäinen ratkaisu tarjoaa.

Ajax Load More -lisäosa lisää sivustoille mahdollisuuden listata sivuston sisältöjä kuten artikkeleita yhdelle sivulle loputtomalla selauksella. Nimensä mukaisesti lisäosan toiminta perustuu Ajaxiin eli joukkoon erilaisia web-kehityksessä käytettäviä tekniikoita, jotka mahdollistavat tiedon hakemisen palvelimelta samaan aikaan, kun käyttäjä on sivulla, ilman tarvetta ladata sivu uudelleen.

3 TEEMAKEHITYKSEN TOTEUTUS

3.1 Versionhallinta

Pohjateeman kehitystä suunniteltaessa päädyttiin siirtämään edellisen pohjateeman uusin versio versionhallinnassa uuteen, omaan repositorioon eli varastoon Bermudan käyttämässä versionhallintatyökalussa Bitbucketissa. Versionhallinnan kautta muut verkkokehitystiimin jäsenet pysyivät mukana toteutuksen kulussa ja pystyivät antamaan omia huomioita ja ehdotuksia.

Opinnäytetyön toteutus aloitettiin siis haarauttamalla vanha pohjateema uuden pohjateeman repositorioon. Uuden pohjateeman versionhallintaan siis jäivät edellisen pohjateeman metatiedot, mutta ne eivät kuitenkaan haitanneet kehitystä millään tavalla.

3.2 Teemojen vertailu

Entisen pohjateeman ja tämän opinnäytetyön lopputuotoksena syntyneen teeman erot olivat niin huomattavat, että uusi pohjateema päätettiin erottaa edellisestä teemasta kokonaan, kuitenkin pohjana käyttäen vanhaa teemaa. Uusi pohjateema tulisi siis koostumaan vanhasta Bermudan omasta pohjateemasta ja uusista, eri teemoista mallia ottamalla rakennetuista osista.

Pohjateeman kehittämistä varten päätettiin vertailla suosittuja WordPress-teemoja ja niissä käytettyjä tekniikoita ja käytäntöjä. Bermudan pohjateeman kehityksessä vertailtiin muun muassa Aucor-, Air-, UnderScores- ja Twenty Twenty-teemoja. Näistä sopivimmiksi Bermudan tarpeisiin antoivat pohjaa varsinkin Aucor, UnderScores sekä Twenty Twenty.

Aucor Starter -teema on suomalaisen Turussa toimivan WordPress-toimiston tekemä ja ylläpitämä avoimen lähdekoodin lisenssillä lisensoitu aloitusteema. Teeman hyviä puolia olivat sen Gulp.js-tiedoston selvyys, sekä Gutenberg-editorin lohkojen selkeä tyylittely ja tiedostorakenne.

UnderScores, eli _s -teemaa käytettiin edellisen, vanhemman pohjateeman rakentamisessa ja sillä saatiin hyvä ja tukeva tiedostorakenne aluilleen nykyisen teeman luomiseen. Teeman on tehnyt ja sitä ylläpitää Automattic-niminen yhtiö (Hughes 2017).

Twenty Twenty -teema, eli WordPress-tiimin oma 2020 vuodelle rakennettu teema antoi vertailukohdan oman teeman kehitykseen siitä, mitä WordPressin kehittäjät ovat alun perin halunneet WordPressiin luoduilta teemoilta. Twenty Twenty -teema on erilainen verrattuna muihin WordPress-tiimin tekemiin teemoihin, koska se keskittyy Gutenberg-editoriin ja sitä ei olla tehty kokonaan alusta alkaen, vaan se on tehty toisen teeman, Chaplinin, pohjalta (Daniele 2020). Muihin esikuva-analyysissa käytettyihin teemoihin verrattuna Twenty Twenty on erilainen, koska se on käytännössä valmis teema, eikä aloitusteema, kuten muut tässä esitellyt teemat.

Jokaisesta teemasta löytyi hyviä ja huonoja puolia, joten päätettiin kerätä jokaisesta teemasta parhaiten yhteensopivat ja toimivat osat ja kasata niistä toimiva ratkaisu. Aucorista otettiin tiedostojen pakkaamiseen soveltuva Gulp.js-tiedosto ja sen rakenne, UnderScoresista otettiin käyttöön tiedostorakenne pohjaksi ja sitä muutettiin omaan käyttöön sopivaksi. Twenty Twenty -teemasta ei suoranaisesti otettu osia käyttöön, vaan sitä käytettiin lähtökohtaisesti vertailukohtana hyvästä WordPress-teemasta.

3.3 Tiedostorakenteen uudelleenjärjestäminen

Tiedostorakenteeseen tehtyjen muutosten tarkoituksena oli tukea Aucorin teeman pohjalta rakennetun Gulp.js-tiedoston käyttämistä ja muokata teeman kehityksestä ymmärrettävämpää. Muutaman kansion ja tiedoston nimet vaihdettiin uusiin, kuvaavampiin nimiin, kuten esimerkiksi page-frontpage.php-tiedoston nimi vaihdettiin front-page.php nimeen, koska WordPress luokittelee jälkimmäisen nimen korkeammalle omassa sivumalli hierarkiassaan (Template Hierarchy 2020). Tällöin mahdollisia järjestysongelmia ei pitäisi normaaleissa olosuhteissa syntyä.

3.4 Gulp.js-konfigurointi ja käyttöönotto

Entinen Gulp.js-tiedosto oli toimiva, mutta sitä haluttiin kuitenkin parantaa joissakin tehtävissä, kuten Javascript tiedostojen käsittelyssä. Tiimin päätöksestä päätettiin uudistaa koko Gulp.js-tiedosto ja pohjaksi tälle otettiin Aucorin Gulp.js-tiedosto.

3.4.1 Manifest.js

Src-kansioon lisättiin aluksi Manifest.js-tiedosto, jossa määritellään Javascript- ja SCSS-tiedostojen alku- ja lopputiedostot. Yhteen lopputiedostoon voidaan yhdistää useampi alkutiedosto, kuten alla olevassa kuvasta voi nähdä (kuva 1).

Kuva 1. manifest.js-tiedoston sisältöä

Kuvassa ylimpänä näkyy ”main.js”, mikä on kyseisen lopputiedoston nimi, ja sen jälkeen on määritelty listassa lopputiedostoon tulevat tiedostot, eli tässä tapauksessa kaikki tiedostot, jotka löytyvät scripts-kansion sisältä löytyvästä lib-kansiosta.

Tällä menetelmällä kehittäjä löytää helposti tiedostot, jotka Gulp muuttaa ja siirtää dist-kansioon, yhdestä tiedostosta, ja pystyy tarvittaessa tekemään muutoksia tähän tiedostoon, verrattuna siihen, että hän joutuisi tekemään saman muutoksen kaikkiin paikkoihin, joissa muutosta vaativaa tiedostoa on käytetty.

3.4.2 SCSS

Gulp.js-tiedostoon määriteltiin kaksi erilaista tehtävää, jotka käsittelevät SCSS-tiedostoja. Ensimmäinen tehtävä prosessoi tiedostot, eli luo tiedostoista lähdekartat, muuntaa SCSS-tiedostot CSS-tiedostoiksi, yhdistää ja pienentää tiedostot (kuva 2).

```
const cssTasks = (filename) => {
  return lazypipe()
    // catch syntax errors (don't break pipe)
    .pipe(function() {
      return gulpif(!enabled.failStyleTask, plumber());
    })
    // init sourcemaps
    .pipe(function() {
      return gulpif(enabled.maps, sourcemaps.init());
    })
    // sass
    .pipe(function() {
      return gulpif('*.scss', sass({
        outputStyle: 'nested', // libsass doesn't support expanded yet
        precision: 8,
        includePaths: ['.'],
        errLogToConsole: !enabled.failStyleTask
      }));
    })
    // combine files
    .pipe(concat, filename)
    // autoprefix
    .pipe(autoprefixer, {})
    // minify
    .pipe(cleancss, {})
    // build sourcemaps
    .pipe(function() {
      return gulpif(enabled.maps, sourcemaps.write('.', {
        sourceRoot: path.styles.source
      }));
    })();
};
```

Kuva 2. Ensimmäinen tyylitiedostoja koskeva tehtävä.

Lähdekarttojen avulla selaimen kautta tapahtuva virheenkorjaus onnistuu paremmin, koska kartat kertovat selainten kehitystyökaluille, mistä tiedostosta CSS-säännöt ovat alunperin lähtöisin. Tiedostojen yhdistäminen ja pienentäminen vähentää selaimen ja palvelimen välisten pyyntöjen määrää nopeuttaen sivustojen latautumista.

Toinen SCSS-tiedostoja käsittelevä tehtävä käyttää ensimmäistä tehtävää prosessoidakseen tyylitiedostot ja ilmoittaa niistä löytyneistä virheistä. Jos virheitä ei ole löytynyt niin se siirtää lopulliset tyylitiedostot dist-kansioon ja ilmoittaa onnistuneesta toteutuksesta kehittäjälle.

3.4.3 Javascript

Gulp.js-tiedostosta löytyy samat tehtävät myös Javascript-tiedostoille, eli prosessointitehtävä ja kokoamistehtävä. Ensimmäinen tehtävä toimii samalla tavalla kuin SCSS-tiedostojen kanssa, mutta se ei muuta tiedostoja muuhun ohjelmointikieleen, vaan se käyttää Babelia muuttaakseen Javascript-koodin sopivaksi myös vanhemmille selaimille.

Toinen Javascript-tiedostoihin liittyvä tehtävä käyttää ensimmäistä tehtävää tiedostojen prosessointiin, mutta lisää tiedostoihin scripts-kansiosta löytyvän minified-kansion kaikki tiedostot, koska muuten Gulp rikkoo jo valmiiksi pienennetyt Javascript-tiedostot. Tällä tavalla jo valmiiksi pienennetyt tiedostot saadaan helposti siirrettyä muiden tiedostojen kanssa dist-kansioon. Virheiden ilmaantuessa tämä tehtävä ilmoittaa niistä terminaalin kautta kehittäjälle (kuva 3).

```
gulp.task('scripts', () => {
  const merged = merge();
  // process all assets
  for (i = 0; i < jsAssets.length; i++) {
    let asset = jsAssets[i];
    const jsTasksInstance = jsTasks(asset.name);
    merged.add(
      gulp.src(asset.globs, {base: 'scripts'})
        .pipe(jsTasksInstance)
    );
  }
  return merged
    .on('error', function(err) {
      beeper().
      notify({
          message: "\n\n🔥🔥🔥🔥Error in JS🔥🔥🔥🔥\n",
          onLast: true,
      });
      console.log(err);
    })
    .pipe(gulp.src("src/js/minified/*.js"))
    .pipe(gulp.dest(path.scripts.dist))

    .pipe(
      notify({
          message: "\n\n🔥🔥🔥  Compiled JS successfully 🔥🔥🔥\n",
          onLast: true,
      })
    );

});
```

Kuva 3. Toinen Javascript-tiedostoja käsittelevä kuva

Vendor-tiedostoille on luotu erillinen tehtävä, joka kopioi vendor-kansiosta kaikki tiedostot suoraan dist-kansioon. Vendor-tiedostoja oli opinnäytetyön toteutuksen aikaan vain Bootstrapiltä.

3.4.4 Teeman kuvatiedostot

Src-kansiossa sijaitsevaan images-kansioon tallennetaan teemassa käytettävät kuvat, joita ei hallinnoida WordPressin hallintasivuilta. Näitä tiedostoja ei ole merkitty manifest.js-tiedostoon, vaan niiden lähdepolku on merkitty suoraan Gulp.js-tiedostossa olevaan images-tehtävään. Tämä tehtävä siirtää kaikki images-kansion tiedostot dist-kansioon.

Tähän tehtävään on mahdollista lisätä toiminnallisuus, joka pakkaa kuvatiedostot pienempään tiedostokokoon, mutta toiminnallisuutta ei otettu käyttöön vielä tässä vaiheessa, koska pakkaustoiminto hidasti tiedostojen käsittelyä huomattavasti. Jatkokehityksessä toiminto kuitenkin otetaan todennäköisesti käyttöön, jos se saadaan toimimaan nopeammin ja se saataisiin tunnistamaan jo valmiiksi pakatut kuvatiedostot.

3.5 Vanhan pohjateeman osien arviointi

Vanhan Snow-pohjateeman tiedostot käytiin yksitellen lävitse, ja niiden dokumentointeja parannettiin ja lisättiin puuttuvia dokumentointeja. Functions.php-tiedosto, johon kasataan kaikki WordPress-teeman toiminnallisuudet, järjestystä muutettiin ja osioita jaoteltiin tarkemmin kuvaavampien otsikoiden alle kuten alla olevassa kuvassa näkyy (kuva 4).

```
18
19  // Setup Gutenberg editor
20  require get_template_directory() . '/includes/editor-capabilities.php';
21  require get_template_directory() . '/includes/content-editor.php';
22
23  // Add favicon to dashboard.
24  require get_template_directory() . '/includes/dashboard-favicon.php';
25
26  // disable all unnecessary features.
27  require get_template_directory() . '/includes/disable-features.php';
28
29  // Add ajax functions
30  require get_template_directory() . '/includes/ajax-load-more.php';
31  require get_template_directory() . '/includes/ajax-search.php';
32
33  // Add population for gravity forms block.
34  require get_template_directory() . '/includes/populate-gravity-forms-block.php';
35
```

Kuva 4. functions.php-tiedoston järjestystä ja jaottelua

Vanhan pohjateeman räätälöidyistä ACF:llä tehdyistä lohkoista poistettiin ne lohkot, jotka pystyttiin korvaamaan Gutenbergin omilla lohkoilla. Nämä lohkot olivat ankkurilohko, ruudukkolohko ja jaettu sisältölohko. Ankkurilohko, joka loi ankkurilinkin, jota pystyttiin käyttämään URL-osoitteessa osoittamaan ankkurilinkin kohtaan, ruudukkolohko, jolla pystyttiin toteuttamaan vaativampia asetteluja lohkoille. Jaettu sisältölohko jakoi lohkon kahteen osaan, joihin pystyi lisäämään sisältöä. Kaksi viimeisintä lohkoa pystyttiin korvaamaan Gutenbergin alkuperäisellä kolumnilohkolla.

3.6 Ajax toiminnallisuuksien kehitys

Entiseen pohjateemaan oli Ajax:lla toteutettu sivuston hakuun ja arkistosivuilla toimiva ratkaisu, joka mahdollisti esimerkiksi hakusivulla haun tekemisen ilman sivun uudelleenlatausta. Tässä ratkaisussa oli kuitenkin omat ongelmansa, kuten se, että päivittääkseen hakusivun tyyliä piti tiedot muuttaa kahdessa eri tiedostossa. Kyseinen ratkaisu vaihdettiin uudessa teemassa toteutettavaksi Ajax Load More -lisäosalla.

Ajax Load More -lisäosan lyhytkoodi lisättiin pohjateeman arkistosivun mallipohjaan sekä hakusivun mallipohjaan, joka mahdollistaa hakutulosten helpon käsittelyn ja muokkaamisen tarvittaessa. Lisäosan kanssa syntyviä ongelmatilanteita varten sivuille laitettiin WordPressin alkuperäistä WP_Query-luokkaa käyttävä varajärjestelmä, joka antaa samanlaisen lopputuloksen kuin lisäosa, mutta ilman Ajax:in antamia hyötyjä. Alla olevassa kuvassa näkyy pohjateeman hakusivulla käytetty Ajax Load More -lisäosan lyhytkoodi (kuva 5).

```
<?php
if (shortcode_exists('ajax_load_more')) {
    echo do_shortcode('[ajax_load_more  container_type="div"
                                        post_type="any"
                                        posts_per_page="6"
                                        search=\'' . $_GET["s"] . '\'
                                        scroll="false"
                                        button_label=\'' . $loadmore . '\'
                                        transition_container_classes=""
                                        button_loading_label=\'' . $loading . '\']'
                                    );
} else {
```

Kuva 5. Ajax Load More -lisäosan lyhytkoodi hakusivulla.

3.7 Pohjateeman mallipohjien siistiminen

WordPress-teemoissa käytetään sivujen ulkomuodon määrittelyssä mallipohjia, jotka sisältävät kyseisten sivujen toiminnallisuuksia ja HTML-rakenteen. Näissä esiintyviä osia, jotka toistuvat eri sivuilla, voidaan tallentaa omiin tiedostoihin, joista voidaan hakea WordPressin funktiolla get_template_part(). Näin sivuja voidaan rakentaa modulaarisemmin ja turhalta koodintoistolta säästytään.

Vanhasta pohjateemasta löytyi seuraavat perusmallipohjat:

- index.php
- 404.php
- archive.php
- comments.php
- page-frontpage.php
- search.php
- searchform.php
- single.php
- page.php.

Näistä poistettiin kokonaan uudessa pohjateemassa comments.php-tiedosto, koska sille ei ollut minkäänlaista käyttöä. Kuten aiemmin raportissa todettiin, page-frontpage.php-tiedosto vaihdettiin front-page.php-tiedostoksi, jotta sen arvo mallipohjien hierarkiassa olisi korkeampi. Search.php-, searchform.php- ja archive.php-tiedostoihin tehtiin muutoksia, jotta Ajax Load More -lisäosa saatiin sisällytettyä teemaan.

Kaikkiin perusmallipohja-tiedostoihin ei tehty muutoksia, koska niiden sisällöt ovat lyhyitä ja yksiselitteisiä. Tämän takia niiden toimintaa ei tarvitse lähteä muuttamaan suuresti, koska se tekisi pohjateeman toimivuudesta monimutkaisempaa. Suurimmalta osin muutokset olivat erilaisten luokkien lisäämistä ylimpiin HTML-elementteihin tyylittelyn yksinkertaistamista varten tai Bootstrap-luokkien lisäystä.

4 POHDINTAA JA JATKOTOIMENPITEET

4.1 Pohdintaa

Vaikka opinnäytetyö toteutettiin nopeasti noin kuukaudessa, opittiin siinä paljon WordPress-teeman kehityksestä. Vuonna 2020 WordPressistä on julkaistu kaksi merkittävää julkaisua, WordPress 5.4 ”Adderley” 31. marraskuuta (Wahalahti 2020a) ja WordPress 5.5 ”Eckstine” 12. elokuuta (Wahalahti 2020b).

Molemmat julkaisut toivat useita muutoksia WordPressin toimintaan ja näin tiheä päivitysrytmi vaatii sen, että teemankehittäjät päivittävät omia teemojaan tarpeeksi usein pysyäkseen mukana. Tämän takia tämän opinnäytetyön lopputuotosta eli uutta pohjateemaa ei voi kutsua valmiiksi, vaan tämä on enemmänkin versio 1.0, jota tullaan aluksi testaamaan pienissä asiakasprojekteissa ja kehittämään esille tulevien ehdotusten ja ongelmien pohjalta.

Alkuperäisestä suunnitelmasta poiketen sovittiin työn toimeksiantajan kanssa, että WordPressin käyttöön tarkoitetut ohjeet tullaan siirtämään myöhemmälle ajankohdalle, jotta pohjateema saataisiin käyttövalmiiksi ennen ensimmäistä asiakasprojektia. Tämä päätös tehtiin, koska ohjeisiin haluttiin panostaa enemmän aikaa, kuin mitä oli tarjolla, että niistä saataisiin mahdollisimman hyvänlaatuiset.

Advanced Custom Fields -lisäosan käyttäminen ja sen konfiguraatioon ei tämän opinnäytetyön aikana koskettu, koska sen nähtiin toimivan hyvin omissa tehtävissään. Kyseinen osuus siis siirrettiin vanhasta pohjateemasta uuteen hyvin pienillä muutoksilla.

4.2 Toteutuksen aikana syntyneet jatkokehitysideat

Pohjateeman jatkokehitystä tullaan toteuttamaan pitkälle tulevaisuuteen. Tässä luvussa käsittelen tämän opinnäytetyön työstön aikana syntyneitä jatkokehitysideoita. Suurimmat ideat tulevat esille pohjateeman testaamisen aikana, mutta olen kerännyt niitä jo valmiiksi itselleni listaan tulevan kehityksen suunnittelemiseksi.

Ensimmäisenä askeleena pohjateeman jatkokehityksessä tulee olemaan teeman toimivuuden laajempi testaaminen pienikokoisissa asiakasprojekteissa ja niissä syntyneiden sivustojen testaaminen Googlen tarjoamalla PageSpeed Insights -työkalulla, jolla saataisiin parempaa kuvaa siitä, mitä teemassa voitaisiin parantaa esimerkiksi sivujen latausnopeuden suhteen. Testauksella saadaan myös paljastettua työssäni syntyneitä ohjelmointivirheitä, joita toteutuksen aikana ei tullut esille.

Opinnäytetyön suunnitteluvaiheessa mietittiin Bootstrap 5.0 alpha-version käyttöönottamisesta. Seuraavassa Bootstrap-versiossa muun muassa luovutaan jQuery JavaScript-kirjastosta, joka pudottaa koko Bootstrap-version tiedostokokoa ja nopeuttaa sivujen latausta (Otto 2020). Päädyimme kuitenkin jättämään alpha-version pois tässä vaiheessa, koska pohjateemaan halutaan vakaa versio, jonka toimintakyvystä voidaan olla varmoja. Tämän takia Bootstrap 5.0-versio lisätään pohjateemaan vasta sitten, kun se on varsinaisesti julkaistu.

4.3 Loppupäätelmät

Opinnäytetyön tavoitteena oli tutkia ja selvittää erilaisia työskentelytapoja ja -käytäntöjä, millä Bermudan verkkosivustoprojekteissa käytettävää pohjateemaa voidaan kehittää nopeammin ja selkeämmin. Tähän tavoitteeseen päästiin jossakin määrin, ei kuitenkaan täydellisesti. Pohjateeman kanssa työskentely muuttui enemmän Gutenberg-pohjaiseksi, koska toimivia sivuja pystytään nykyään luomaan ilman, että joudutaan heti muokata pohjateeman koodia. Sivujen jne. luonti voi lähteä suoraan WordPressin hallintopuolelta käyttäen Gutenbergin omia lohkoja hyväksi.

Työn tarkoituksena oli kehittää ja dokumentoida Bermudan nykyistä WordPress-pohjateemaa helpommin käytettäväksi ja nopeammaksi esimerkiksi sivujen prototyyppien toteutuksessa, sekä luoda WordPressin ja pohjateeman käyttöön yleiset ohjeet Bermudan asiakkaiden käytettäväksi. Ensimmäisestä osiosta onnistuttiin, mutta ohjeiden luonti jouduttiin yhteisymmärryksessä siirtämään jatkokehityksen puolelle. Kuitenkin uudesta pohjateemasta saatiin toteutettua toimiva kokonaisuus, jonka käyttö edesauttaa ja nopeuttaa projektien alkuvaiheen työskentelyä.

LÄHTEET

Hughes, John. Julkaistu 22.08.2017. The Beginner's Guide to Creating a Theme With Underscores. Luettu 22.10.2020. https://torquemag.io/2017/08/beginners-guide-to-creating-a-theme-underscores/

Template Hierarchy, Theme Handbook. n.d. Luettu 22.10.2020. https://developer.wordpress.org/themes/basics/template-hierarchy/

Daniele, Carlo. Kinsta. 2020. Twenty Twenty: An Introduction to the New Default WordPress Theme. Julkaistu 05.10.2020. luettu 27.10.2020. https://kinsta.com/blog/twenty-twenty-theme/

Otto, Mark. 2020. Bootstrap 5 alpha!. julkaistu 17.06.2020. luettu 28.10.2020. https://blog.getbootstrap.com/2020/06/16/bootstrap-5-alpha/

Wahalahti, Timi. WordPress.org. 2020a. WordPress 5.4 "Adderley". Julkaistu 01.04.2020. Luettu 27.10.2020. https://fi.wordpress.org/2020/04/01/adderley/

Wahalahti, Timi. WordPress.org. 2020b. WordPress 5.5 "Eckstine". Julkaistu 11.08.2020. Luettu 27.10.2020. https://fi.wordpress.org/2020/08/11/eckstine/