

Mari Myllykangas

WORDPRESS-TEEMAN TOTEUTTAMINEN

WORDPRESS-TEEMAN TOTEUTTAMINEN

Mari Myllykangas
Opinnäytetyö
Kevät 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma, Web-sovelluskehityksen sv

Tekijä(t): Mari Myllykangas

Opinnäytetyön nimi: Wordpress-teeman toteuttaminen

Työn ohjaaja(t): Jouni Juntunen

Työn valmistumislukukausi ja -vuosi: Kevät 2014

Sivumäärä: 55

Opinnäytetyön tavoitteena oli kehittää Wordpress-teema käyttämällä Wordpress-julkaisujärjestelmän tarjoamaa sovellusrajapintaa. Kaikki tarvittava materiaali teemaan tehtiin itse. Työllä ei ollut toimeksiantajaa, mutta aihe on ajankohtainen, sillä monet sivustot nykypäivänä käyttävät Wordpressia alustanaan. Näinollen Wordpress osaaminen on varmasti hyödyksi työelämässä.

Suurin osa taustatiedoista löytyi Wordpressin sivuilta löytyvästä Codexista. Teema tehtiin käyttämällä enimmäkseen PHP:tä ja HTML:ää. Työn aikana tuotettu koodi käsiteltiin enimmäkseen Wordpressin hallintapaneelistä löytyvän editorin avulla.

Työn tuloksena tuotettiin toimiva teema, joka työn lopuksi lähetettiin Wordpressin teemakirjastoon teemojen tarkistustiimin tarkistettavaksi. Johtopäätös tehdyntyön perusteella on, että teeman tekeminen itse on paras tapa varmistua siitä, että teema sisältää kaikki halutut ominaisuudet.

Asiasanat: Wordpress, sisällönhallinta, teema, ulkoasu, PHP, HTML

ABSTRACT

Oulu University of Applied Sciences

Degree Programme in Business Information Systems, Web application development

Author(s): Mari Myllykangas

Title of thesis: Development of Wordpress Theme

Supervisor(s): Jouni Juntunen

Term and year when the thesis was submitted: Spring 2014

Number of pages: 55

The objective of this thesis was to develop a Wordpress theme by using Wordpress plugin API. All material needed is self made. The project has no commissioner, but the subject is topical, because many websites nowadays use Wordpress as a basis of their sites. Therefore, Wordpress knowledge is useful and much appreciated at work.

Most of the background information was found at Codex at Wordpress.org. The theme itself is made by using PHP and HTML. The code made during the project was mostly modified via Wordpress Control panel.

The result of the thesis was a functional self-made theme which was sent to the theme archive on the Wordpress.org for Theme Review Team to reviewing. The conclusion is that designing a theme without any material is the best way to create a theme if there is need to ensure that all required features are included.

Keywords: Wordpress, content management, theme, layout, PHP, HTML

SISÄLLYS

1 JOHDANTO.....	6
2 VAATIMUKSET	8
2.1 TOIMINNALLISUUS	8
2.2 ULKOASU	9
2.3 KÄYTTÖLIITTYMÄ.....	10
3 WORDPRESS-TEEMAN RAKENNE.....	13
3.1 TYYLITIEDOSTOT	13
3.2 YLÄTUNNISTE	14
3.3 SIVUPOHJAT	16
3.4 KOMMENTIT	18
3.5 ALATUNNISTE	20
3.6 FUNKTIOT JA ASETUKSET	21
3.7 KIELIVERSIOT.....	24
3.8 KANSIORAKENNE	28
4 WORDPRESS-TEEMAN TOTEUTUS	30
4.1 TYYLITIEDOSTOT	30
4.2 YLÄTUNNISTE	31
4.3 SIVUPOHJAT	35
4.4 KOMMENTIT	38
4.5 SIVUPALKKI	41
4.6 ALATUNNISTE	43
4.7 ASETUKSET	45
5 TEEMAN LISÄÄMINEN TEEMAKIRJASTOON	49
6 POHDINTA.....	53
LÄHTEET	54

1 Johdanto

Yhä useampi web-sivusto rakennetaan nykyisin julkaisujärjestelmää käyttämällä, jotta sivuston päivittäminen olisi helpompaa sellaiselle ylläpitäjälle, joka ei välttämättä ole koskaan aikaisemmin edes katsonut sivuston lähdekoodia, puhumattakaan, että osaisi itse muokata sitä. Netissä on tarjolla lukuisia ilmaisia julkaisujärjestelmiä, joista tällä hetkellä suosituin on Wordpress, joka on vuodesta 2003 asti kehitetty, avoimen lähdekoodin julkaisujärjestelmä (CMS Usage Statistics 2014).

Wordpress tunnettiin aluksi pelkästään blogialustana (Wordpress.org 2014). Vasta myöhemmin sitä alettiin käyttää myös sivustojen tekemiseen, kun uudet versiot mahdollistivat muun muassa erilaisten lisäosien, sekä teemojen eli valmiiden ulkoasujen käytön sivustolla. Teemojen avulla ulkoasun eri osat voidaan jakaa omiin tiedostoihinsa, jolloin on helpompi tehdä muutoksia sivujensa ulkoasuun ja nähdä muutokset heti.

Myöhemmin Wordpress alkoi myös kerätä teemoja sivuilleen tehtyyn teemakirjastoon ja nykyisin sieltä löytyy yli 2000 erilaista teemaa (Wordpress.org 2014). Teemakirjaston teemat ovat ilmaisia, joten kuka tahansa voi etsiä ja ladata sieltä mieleisensä ulkoasun, jonka haluaa asettaa omille sivuilleen. Ilmaisten teemojen lisäksi on myös kaupallisia teemoja, joita esimerkiksi monet freelancerit myyvät sivuillaan. Myytävät teemat eivät välttämättä eroa paljonkaan teemakirjaston teemoista, mutta etuna on, että sivustolta poikkeuksetta löytyy yrityksen tai freelancerin yhteystiedot, jolloin voi neuvotella vaikkapa kokonaan itselleen räätälöidystä yksilöllisestä teemasta.

Teemoja on teemakirjastossa valtava määrä, joten niitä pystyy hakemaan erilaisilla hakukriteereillä. Näitä kriteerejä ovat muun muassa väri, sarakkeiden määrä sekä muut ominaisuudet joita teemasta löytyy, kuten esimerkiksi itse muokattava otsakekuva tai responsiivisuus. Samat hakukriteerit löytyvät yleensä myös kaupallisilta sivustoilta, mutta näiden lisäksi teemoja on mahdollista hakea niiltä myös kategorian mukaan, joka määräytyy sen perusteella millaiseen käyttötarkoitukseen teemaa on alun perin lähdetty

tekemään. Tällaisia kategorioita ovat esimerkiksi portfolio, liiketoiminta ja valokuvaus. Valokuvausta ajatellen tehdyissä teemoissa on esimerkiksi kuvat ja galleriat pyritty saamaan mahdollisimman näyttävästi esille. Vastaavasti liiketoimintaa ajatellen tehdyt teemat on pyritty pitämään yleensä hillittyinä ja mahdollisimman asiallisen ja selkeän näköisinä, jotta kävijä varmasti löytää haluamansa tiedon sivuilta.

2 Vaatimukset

Opinnäytetyön tavoitteena on kehittää teema pieniä sivustoja ja blogeja silmällä pitäen, joten kohderyhmäksi valikoituvat todennäköisesti eri-ikäiset yksityishenkilöt, jotka tarvitsevat ulkoasua esimerkiksi harrastusta käsittelevää blogia tai sivustoa varten. Tavoitteena on toteuttaa teema siten, ettei sen luomisessa käytetä minkäänlaista pohjaa, vaan kaikki teeman tiedostot luodaan itse Wordpressin ohjelmointirajapintaa (engl. API, Application programming interface) käyttämällä. Valmistumisensa jälkeen teema ladataan Wordpressin teemakirjastoon kenen tahansa käytettäväksi.

Aihe valittiin koska monet isotkin sivustot ja palvelut käyttävät pohjanaan juuri Wordpressia tai ovat hiljattain siirtyneet sen käyttöön. Vuonna 2013 internetistä löytyvistä sivustoista 19 % käytti pohjanaan Wordpressia, mikä tarkoittaa 60 miljoonaa sivustoa (O'Dell 2013). Näin ollen perusteellisemmasta Wordpressin ja teemojen tuntemuksesta tulee varmasti olemaan hyötyä työelämässä. Varsinkin Wordpressin kehittyessä, saattaa vanhemmille versioille tarkoitetuissa teemoissa ilmetä ongelmia, esimerkiksi vanhojen Wordpress-komentojen jäädessä pois käytöstä uudemmissa versioissa. Näitä ongelmia on kyettävä korjaamaan, jos ylläpitäjä haluaa edelleen pitää teeman käytössä. Osittain kiinnostus aiheeseen tulee myös työharjoittelupaikasta, jossa Wordpressia käytettiin asiakkaille tehtävien sivustojen alustana.

Koska teema tulee kenen tahansa ladattavaksi teemakirjastoon, sen loppukäyttäjää ei millään voi tietää varmasti etukäteen. Tämän vuoksi teeman toiminnoille ja ulkoasulle asetettavat vaatimukset on määriteltävä teemalle päätetyn käyttötarkoituksen ja sen perusteella oletetun kohderyhmän mukaan.

2.1 Toiminnallisuus

Teeman käyttöön ottava ylläpitäjä ei välttämättä osaa lukea HTML- tai CSS-koodia laisinkaan, joten tämän vuoksi tulee teeman olla helposti muokattavissa, mikäli ylläpitäjä haluaa saada vaikkapa sosiaalisen median linkit näkyviin

sivuille. Teemaan toteutetaan siksi mahdollisuus lisätä sosiaalisen median linkit siten, että osoitteen voi kopioida sille varattuun tekstikenttään asetuksissa.

Lisäksi teemaan lisätään mahdollisuus vaihtaa otsakekuvaa mielensä mukaan ja lisätä Wordpressista löytyviä vimpaimia, niille varattuun kohtaan sivulla. Erityisen tärkeä ominaisuus on kuitenkin yhteensopivuus eri lisäosien kanssa, sillä lisäosien avulla sivustolle voi lisätä ominaisuuksia, joita ei ohjelmoida joka teemaan mukaan esimerkiksi liian suureksi kasvavan työmäärän vuoksi tai siksi, ettei kohderyhmän oleteta tarvitsevan ominaisuutta. Tällainen ominaisuus ovat esimerkiksi kieliversiot, joilla ylläpitäjä saisi bloginsa artikkelit näkymään kahdella tai useammalla kielellä.

2.2 Ulkoasu

Sivustolle tuleva kävijä haluaa ensisijaisesti löytää sivuilta etsimänsä tiedon, joten sen vuoksi ensimmäinen ja kenties tärkein vaatimus teeman ulkoasulle on selkeys. Navigoinnin on oltava sellainen, että kävijä tajuaa sen navigoinniksi, sen sijaan että etusivulla olisi esimerkiksi joukko erikokoisia kuvalinkkejä, mutta ei mitään vihjettä siitä mitä eri kuvien takaa löytyy.

Tietokone on edelleen suosituin väline netin selaamiseen, mutta tablettien ja mobiililaitteiden käyttö on kuitenkin kasvussa (StatCounter Global Stats 2013). Tämän vuoksi kävijät varmasti arvostavat myös sitä, että ulkoasu on responsiivinen ja asettuu hyvin käytössä olevan laitteen näytölle. Ulkoasun on oltava myös muuten kävijäystävällinen. Pelkästään Suomessa arvioidaan, että 1,5 % väestöstä on jollain tavalla näkövammaisia eli noin 80 000 henkilöä (Näkövammaisten Keskusliitto ry 2014). On siis hyvin todennäköistä, että joku kävijöistä ei näe kunnolla, joten fontin on oltava selkeä ja tarpeeksi iso, jotta sen varmasti näkee lukea. Lisäksi ulkoasussa on myös hyvä kiinnittää huomiota värien käyttöön. Esimerkiksi punaisen ja vihreän yhdistäminen on yksi pahimpia virheitä jonka ulkoasua suunnitellessa voi tehdä, sillä tutkimusten mukaan 8 % maailman miesväestöstä on värisokeita ja naisista 0,5 % (Wearecolorblind.com 2012).

Tämän teeman ensisijaiseksi taustaväriksi on valittu valkoinen. Väri aliarvioidaan usein ja ohitetaan tylsän näköisenä, mutta mustan tavoin se sopii yhteen melkein pä minkä tahansa värin kanssa. Siinä missä musta antaa sivustolle hieman raskaamman vaikutelman ja tunnelman, valkoinen mielletään usein kevyeksi ja ilmapiksi. Valkoinen taustaväri on myös silmiä ajatellen helpompi katsella kuin musta tai jokin muu tumma väri, jota katsellessa silmät alkavat ennen pitkää väsyä. (McNeil 2008.)

Toissijaisena oletusvärinä käytetty sinivihreä puolestaan tehostaa entisestään ilmavaa vaikutelmaa, joka valkoisesta syntyy. Lisäksi väri yhdessä valkoisen kanssa myös tuo mielikuvia vedestä ja jäädä, joten teema alkaa näillä väreillä herättää myös talvisia ja luonnonläheisiä mielikuvia. Kolmantena elementtinä ilmavuutta tuo myös ylimpänä olevan otsikkoelementin liukuväritausta, joka saa sivun näyttämään siltä kuin sinivihreä vähitellen haihtuisi ja muuttuisi valkoiseksi. (McNeil 2008.)

Värien lisäksi, teemasta löytyy myös kuvia, kuten Facebookin ja Twitterin logot, jotka ilmestyvät sivupalkkiin näkyviin, mikäli ylläpitäjä syöttää profiiliensa linkit asetuksissa. Suosituimmilla palveluilla on suunnittelijoille ja kehittäjille tarkoitettut infisivunsa, joista löytyvät muun muassa palveluiden logot ja graafinen ohjeistus siitä miten logoja saa käyttää. Teemassa käytetyt logot on kerätty näiltä palveluiden infosivuilta, joten minkäänlaista tekijänoikeusrikkomusta ei näin ollen tapahdu. Teemassa oletuksena käytetyt otsakekuvat puolestaan ovat itse otettuja valokuvia, joten tekijänoikeudet ovat niidenkin osalta kunnossa.

2.3 Käyttöliittymä

Teeman käyttöliittymästä tulee mahdollisimman yksinkertainen, jolloin se on helposti muokattavissa, mikäli ylläpitäjä tahtoo tehdä siitä lapsiteeman eli muokata teemaa paremmin tarpeitaan vastaavaksi tai tehdä sen pohjalta kokonaan uudenlaisen teeman. Tämän vuoksi oletusnäköymänä käytetään

ratkaisua, jossa elementit tulevat allekkain, lukuun ottamatta sisältöaluetta ja sivupalkkia, jotka sijaitsevat sivun keskellä vierekkäin (kuva 1).



KUVA 1. Sivuston oletusnäkömön rautalankamalli.

Nykyisin on laajasti tapana lisätä sivuston nimi suoraan logoon tai muuhun otsakekuvaan, mutta koska teema tulee kenen tahansa ladattavaksi ja käytettäväksi, sivuston nimi tulee tässä ylimmäksi ja otsakekuvan paikka sen alapuolelle. Näin molemmat osuvat kävijän silmiin heti ensimmäisenä kun sivu avataan. Navigointi puolestaan on sijoitettu logon alle, mutta sisällön yläpuolelle siitä syystä, että alaspäin selatessa ei kävijän tarvitse nousta ihan sivun yläreunaan asti kun palaa ylöspäin siirtyäkseen tutkimaan seuraavaa sivua.

Varsinainen sisältöalue taas on jaettu sisältöön ja sivupalkkiin johtuen vimpaimista, joita Wordpressissa on mahdollista lisätä sivuille. Vimpaimille varatun alueen voisi tietysti lisätä myös alatunnisteeseen tai sisältöalueen yläpuolelle, mutta tässä tapauksessa sivupalkin varaaminen vimpaimille on kuitenkin paras ratkaisu. Jos sivulla näkyy vaikkapa useampi artikkeli peräkkäin, vimpainalue voisi mennä hyvin alas sivulla ja näin kävijä joutuisi selaamaan sivua alaspäin kohtuuttoman paljon jos haluaa päästä johonkin tiettyyn sisältöön käsiksi. Sisältöalueen yläpuolella vimpaimet taas suurella todennäköisyydellä aiheuttavat ärsyyntymistä niissä kävijöissä, jotka eivät vimpainalueen sisällöstä

ole kiinnostuneita, varsinkin jos vimpaimia on paljon. Siksi tässä tapauksessa on parempi pitää varsinainen sisältö ja sivupalkki rinnakkain.

Sivupalkin oikeasta sijainnista voi kuitenkin olla montaa mieltä. Osa ylläpitäjistä haluaa sen oikealle puolelle sivua ja osa taas haluaa sen mieluummin vasemmalle puolelle. Siksi tähän teemaan toteutetaan mahdollisuus vaihtaa näkymää siten, että sivupalkki näkyy vasemmassa reunassa oikean reunan sijaan. Joillakin sivuilla saattaa myös olla tarpeellista, että sivupalkki ei näy, joten sen vuoksi teemaan sisältyy myös sivupohja, jolla sisältöalueen saa koko sivun levyiseksi (kuva 2).



KUVA 2. Rautalankamallit toisella puolen sijaitsevasta sivupalkista ja koko sivun levyisestä sisältöalueesta.

3 Wordpress-teeman rakenne

Teeman eri osat on jaettu aina omiin tiedostoihinsa ja näistä tiedostoista osa on joka teemassa pakollisia ja osa taas välttämättömiä vain joissain teemoissa. Teema toimii pääasiassa Wordpressiin itseensä ohjelmoitujen funktioiden avulla, joissa on käytetty hyödyksi runsaasti PHP:tä ja MySQL:ää. Eri funktioiden ja komentojen avulla teeman saa hakemaan jonkin halutun tiedon, jota teeman tekemisessä tarvitaan, esimerkiksi teeman tyylitiedoston URL-osoitteen.

3.1 Tyylitiedostot

Ellei teemassa ole vähintään yhtä tyylitiedostoa, Wordpress näyttää automaattisesti virheilmoituksen siitä, että tyylitiedosto tarvitaan. Tyylitiedoston avulla teemalle määritellään ulkoasu, kuten sivuston rakenne, värit ja grafiikka.

Jotkut teemat ovat niin laajoja, että vaativat tuhansia rivejä määrittelyjä, joten ei ole epätavallista, että yhden tyylitiedoston sijaan niitä voikin olla useampia, sillä tyylejä on näin helpompi hallita. Esimerkiksi hallintapaneelista tapahtuville toiminnoille tarvitsee tyylitiedostossa olla omat tyylinsä, niin etteivät esimerkiksi artikkeleissa olevat kuvat asetu ihan miten sattuu ja ylläpitäjän vaikka kursivoitessa tekstinpätkän, kyseinen teksti myös näkyy kursivoituna. Useat käyttäjät tapaavat siis laittaa nämä niin sanotusti ”Wordpressin omat” tyylit omaan tiedostoonsa ja varsinaisen teeman tyylit taas omaansa. Joissain responsiivisissa teemoissa on myös eri kokoja varten kokonaan omat tiedostonsa, sen sijaan että kaikkien kokojen määrytykset löytyisivät samasta tiedostosta (kuva 3).

```
<link rel="stylesheet" type="text/css" media="only screen and (max-width: 1024px)" href="1024.css" />  
<link rel="stylesheet" type="text/css" media="only screen and (min-width: 1024px)" href="style.css" />
```

KUVA 3. Kuvakaappaus eri kokoja varten suunnitelluista tyyleistä.

Ensimmäisenä kutsutussa tyylitiedostossa sijaitsevat määrytykset siis vaikuttavat vain jos näyttö tai selaimen ikkuna on kapeampi kuin 1024 pikseliä. Alempi tyyli taas vaikuttaa siitä isompiin näyttöihin ja selainikkunoihin.

Wordpressin tyylitiedostoissa tulee lisäksi muistaa kommentoida teeman nimi, versio ja muut tiedot oletustyylien alkuun (Spooner 2011). Ilman oletustyylien alussa olevia kommentteja, teeman nimi ja muut tiedot eivät näy lainkaan Wordpressin hallintapaneelissa. Sen sijaan teeman nimenä näkyy sen kansion nimi, johon teeman tiedostot on tallennettu. Esimerkiksi Wordpressin oman Twenty Ten -teeman kohdalla nimenä näkyisi twentyten, ellei oletustyylien alussa olevia kommentteja olisi (kuva 4).

```
/* Theme Name: Testiteema  
Description: Testikuvaus  
Version: 1.0  
Author: A. Esimerkki  
Author URI: http://www.esimerkki.fi */
```

Testiteema Versio 1.0

Tekijä [A. Esimerkki](#)

Testikuvaus

KUVA 4. Kuvakaappaus teeman tiedoista Wordpressin hallintapaneelissa.

3.2 Ylätunniste

Ylätunnisteeseen eli header.php-tiedostoon sijoitetaan html- ja body-elementtien avaustagit, sekä muut elementit, jotka toistuvat sivuston yläreunassa joka sivulla. Tällaisia elementtejä ovat esimerkiksi sivuston otsikko ja otsakekuva, mikäli teemassa sellainen on. Sen sijaan, että nämä elementit sijoitettaisiin joka sivupohjaan erikseen, ne sijaitsevat omassa tiedostossaan, jotta mahdolliset muutokset tarvitsee käydä tekemässä vain yhteen tiedostoon.

```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

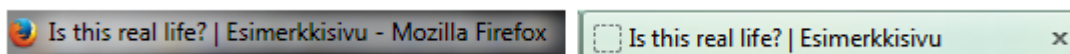
  <title><?php bloginfo('name'); ?> <?php wp_title("|", true); ?></title>
  <link rel="stylesheet" type="text/css" media="all" href="<?php bloginfo('stylesheet_url'); ?>" />

  <?php wp_head(); ?>
</head>
<body>

```

KUVA 5. Kuvakaappaus pätkästä ylätunnisteen koodia

Koska teema tulee kenen tahansa ladattavaksi ja käytettäväksi, sivuston osoitetta ei voi tietää etukäteen. Tämän vuoksi ylätunnisteessa tarvittavia osoitteita ja tietoja kutsutaan erilaisilla komennoilla (kuva 5). Sivuston ylläpitäjä pystyy Wordpressin hallintapaneelissa antamaan sivustolle nimen ja tämä nimi yleensä kutsutaan title-tagien väliin `bloginfo('name')`-komennolla. Sivuston nimen lisäksi voidaan kutsua myös itse sivun nimeä, jolla käyttäjä sillä hetkellä on ja tämä tapahtuu `wp_title()`-komennolla. Lopputuloksena sivuston ja sivun nimi tulostuvat selaimessa välilehteen tai selaimen yläreunaan näkyviin (kuva 6). Kaikki eivät välttämättä halua yksittäisen sivun nimeä näkyville, mutta hakukoneoptimoinnin vuoksi title-tagien välissä on oltava muutakin kuin sivuston nimi (Raittila 2013). Näin kävijä voi tietoa hakiessaan päästä hakukoneen kautta suoraan etsimäänsä tietoon käsiksi, esimerkiksi yhteystietoihin.



KUVA 6. Title elementin tiedot näkyvissä selaimen yläreunassa ja välilehdessä.

Yksi tärkeimmistä komennoista, jotka ylätunnisteeseen on muistettava lisätä on kuitenkin `wp_head()`-komento. Tämä komento lisää Wordpressin ja mahdollisten lisäosien vaatimat tyylit ja tiedot ylätunnisteeseen, joten tämä komento on erityisen tärkeä lisätä, mikäli tahtoo teeman toimivan kunnolla.

Tyylitiedostoja taas voidaan kutsua kahdella tavalla (kuva 7). Ensimmäinen tapa on kutsua oletustyylitiedostoa. Komento `bloginfo('stylesheet_url')` kutsuu

yksinkertaisesti tyylitiedoston osoitteen ja tulostaa sen niin, että teema kykenee käyttämään tiedostossa määriteltyjä tyylejä. Toinen tapa taas on tarkoitettu sellaisiin tilanteisiin joissa teemassa on useampia tyylejä ja ne sijaitsevat vaikkapa niille tehdyssä CSS-kansiossa. Komento `echo get_template_directory_uri();` tulostaa itse teeman osoitteen, ja sen perään voidaan kirjoittaa haluttu kansion ja tyylitiedoston nimi.

```
<link rel="stylesheet" type="text/css" media="all" href="<?php bloginfo( 'stylesheet_url' ); ?>" />
<link rel="stylesheet" type="text/css" media="all" href="<?php echo get_template_directory_uri(); ?>/css/style.css" />
```

KUVA 7. Kuvakaappaus eri kansioissa sijaitsevien tyylien kutsumisesta.

3.3 Sivupohjat

Tyylitiedoston lisäksi teemassa on pakko olla yksi sivupohja, jonka mukaan sivuston sisältö näytetään ja useimmiten tämä sivupohja on `index.php`. Ellei teemassa ole yhtäkään sivupohjaa, Wordpress ilmoittaa sen puuttumisesta. Sivupohjat sisältävät sekä PHP- että HTML-kieltä. Näin ollen sivupohjia luodessa on oltava tarkkana sulkeiden ja muiden merkkien kanssa, jotta sivupohja näkyy halutulla tavalla.

Sivupohjissa ensimmäisenä kutsutaan aina ylätunnistetta, jonka sisältö on omassa tiedostossaan, sen sijaan että ylätunnisteen tiedot kopioitaisiin jokaiseen sivupohjaan erikseen. Samalla periaatteella kutsutaan sivupohjaan myös alatunniste ja mahdollinen sivupalkki, mikäli teemassa sellainen on.

Tärkein osa sivupohjia on kuitenkin `while`-silmukka, jolla kutsutaan kaikki sivuston artikkelit ja sivut listaksi, josta voidaan tulostaa sisältö halutulla tavalla. Sisältö kutsutaan näkyviin erilaisilla komennoilla, jotka sijoitetaan silmukan sisäpuolelle. Esimerkiksi komento `the_title();` kutsuu artikkelin otsikon näkyviin ja vastaavasti `the_content();` kutsuu artikkelin sisällön. Komento `the_time();` puolestaan kutsuu kellonajan jolloin artikkeli on luotu. Luonnollisesti näiden PHP-komentojen sekaan kirjoitetaan myös HTML-merkintöjä, niin että sivulle näkyviin tulostettavasta artikkelista saadaan siistin näköinen (kuva 8).


```

<?php get_header(); ?>

<div id="maincontent">
  <?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>

      <div id="article" <?php post_class(); ?>>
        <h2 class="articletitle"><?php the_title(); ?></h2>
        <p class="time"><?php the_time('j.n.Y'); ?> | <?php the_time(); ?>
        <?php edit_post_link('Edit', '| ', ' '); ?> </p>

        <div id="articlecontent">
          <?php the_content(''); ?>
        </div><!--#articlecontent-->
      </div><!--#article-->

    <?php endwhile; ?>
  <?php endif; ?>
</div><!--#maincontent-->

<?php get_footer(); ?>

```

KUVA 8. Yksinkertaisen sivupohjan rakenne.

Vaikka Wordpress vaatii teemaan vain yhden sivupohjan, jotta se kykenisi näyttämään sivuston sisällön, teemoissa on yleensä tästä huolimatta vähintään kolme sivupohjaa. Ensimmäinen näistä on index.php eli pääsivun sivupohja, jonka mukaan etusivu näytetään. Tämä sivupohja listaa sivuston kaikki artikkelit näkyviin ja jakaa ne sivuihin sen mukaan, kuinka monta artikkelia ylläpitäjä haluaa yhdellä sivulla näyttää.

Toinen sivupohja on single.php eli yksittäisen artikkelin sivupohja. Yleensä tämän sivupohjan sisältö on muuten samanlainen kuin pääsivun pohjassa, mutta artikkelien jakaminen sivuihin on korvattu sillä, että esillä olevasta artikkelista pääsee klikkaamaan edelliseen tai seuraavaan artikkeliin. Tässä sivupohjassa myös kutsutaan mahdolliset kommentit näkyviin.

Kolmas teemojen yleisimmistä sivupohjista taas on page.php eli oletussivupohja. Tämän pohjan perusteella näytetään kaikki muu sisältö sivustolla, ellei enempää sivupohjia ole tehty. Tässä sivupohjassa ei yleensä ole minkäänlaista sivutusta, kuten artikkelien pohjassa ja pääsivun pohjassa. Myös tässä sivupohjassa voidaan antaa mahdollisuus kommentointiin kuten

artikkeleissa, mutta tämä kuitenkin vaihtelee eri teemojen välillä. Kaikissa teemoissa yksittäisen sivun kommentointimahdollisuutta ei ole katsottu tarpeelliseksi.

3.4 Kommentit

Kommenttipohja eli comments.php tiedosto sisältää kommenttien kutsumisen tietylle artikkelille, kommenttien sivutuksen ja tietenkin kommentointilomakkeen. Tiedoston alussa tarkistetaan aina tietoturvan vuoksi, ettei kävijä yritä ladata itse comments.php-tiedostoa näytölle (Maes 2008). Näin kommentit tulevat näkyviin pelkästään yksittäistä artikkelia tai sivua tarkastellessa, riippuen siitä missä sivupohjissa kommenttipohjaa kutsutaan.

Toisinaan Wordpressin käyttäjät ovat voineet suojata jotain sisältöä salasanalla, joten tämän vuoksi kommenttipohjassa myös tarkistetaan, onko artikkeli tai sivu salanasuojattu. Jos on, kävijä ei pääse näkemään kommentteja ennen kuin on syöttänyt oikean salasanan. Tämän jälkeen lähdetään kutsumaan artikkeliin kirjoitettuja kommentteja ja tulostetaan ne näkyviin. Kommenttien kutsumisen voi tehdä monella eri tavalla ja näistä yksi on kutsua kommentit yksi kerrallaan foreach-silmukan avulla (kuva 9) tai sitten Wordpressista löytyvällä wp_list_comments(); komennolla.

```
<div id="comments">
  <h3><?php comments_number('No comments', '1 comment', '% comments'); ?></h3>

  <?php if(have_comments()): ?>
  <ol class="commentlist">
    <?php //comments loop start
    foreach ($comments as $comment) : ?>
```

KUVA 9. Kommenttien kutsuminen foreach silmukan avulla.

foreach-silmukan sisällä kutsutaan yksittäisen kommentin tietoja erilaisilla komennoina, samaan tapaan kuin artikkelien tietoja kutsutaan sivupohjissa. Kommenttien tiedot myös suljetaan li-tagien sisään, jotta ne tulostuisivat mahdollisimman siistin näköiseksi listaksi. Tagien sisällä ensimmäisenä

kutsutaan yleensä kommentin kirjoittajaan liittyvät tiedot, kuten kirjoittajan ilmoittama nimimerkki ja kommentin lähetyspäivä. Kommentin kirjoittajan nimen kutsuminen tapahtuu `comment_author_link()`-komennolla (kuva 10). Tämä komento tulostaa sekä nimen, että mahdollisen blogin tai muun sivuston osoitteen, mikäli kirjoittaja on kommenttia lähettäessään ilmoittanut sivuston osoitteen kommentointilomakkeelle. Lisäksi näkyviin voidaan kutsua myös pieni käyttäjäkuva, mikäli kommentoija on asettanut itselleen sellaisen esimerkiksi `gravatar.com` sivustolla.

```
<li <?php echo $oddcment; ?>id="comment-<?php comment_ID() ?>">
    <div class="comment-info">
        <?php //get the avatar of the commentor
        echo get_avatar( $comment, 70 ); ?><br/>
        <strong><?php comment_author_link() ?></strong><br/>
        <?php comment_date('') ?> | <?php comment_time(); ?>
    </div><!--.comment-info-->
```

KUVA 10. Kommentin kirjoittajan tietojen kutsuminen.

Seuraavaksi kutsutaan itse kommenttia. Silmukan sisällä myös tarkistetaan, onko kommentti hyväksytty, mikäli ylläpitäjä haluaa tarkistaa kommentit ennen niiden näyttämistä (kuva 11). Usein kommenttien yhteydessä myös näytetään pelkästään ylläpitäjälle näkyvä linkki, jota painamalla ylläpitäjä pääsee muokkaamaan kommenttia, jos katsoo sen jostain syystä tarpeelliseksi.

```
<div class="comment-data">
    <?php comment_text(); ?>
    <?php //if comment is under moderation, this line will be shown
    if ($comment->comment_approved == '0') : ?>
        <em>Your comment is awaiting moderation.</em>
    <?php endif; ?>
    <?php //the edit link for the admin
    edit_comment_link(); ?>
</div><!--.comment-data-->
```

KUVA 11. Kommentin tietojen kutsuminen.

Kun kommentin tiedot on kutsuttu, tarkistetaan vielä lopuksi, onko kommentointi mahdollista vai onko ylläpitäjä estänyt kommentoimisen hallintapaneelista (kuva 12). Jos kommentointi on estetty, näytetään asiasta yleensä ilmoitus, jotta käyttäjä tietää kommentoinnin olevan estetty. Muussa tapauksessa kuitenkin kutsutaan kommentointilomaketta. Tämän voi tehdä joko Wordpressin `comment_form()`-kutsulla tai sitten lomakkeen voi tehdä itse. Kuten kommentteja kutsuessa, myös kommentointilomakkeessa kykenee muokkaamaan vain tyylejä, jos haluaa käyttää Wordpressin valmista komentoa eikä kaikissa tageissa ole lainkaan valmiita tyylejä joita muokata. Itse tehdyssä kommentointilomakkeessa taas voi itse päättää miten haluaa asetella tekstikentät ja mitä tietoja ylipäätään haluaa kysyä sivustoa kommentoivalta kävijältä.

```
<?php if (comments_open()) { ?>
    <div id="respond">
        <?php comment_form(); ?>
    </div><!--#respond-->

<?php } else { ?>
    <h3>Comments are closed</h3>
<?php } ?>
```

KUVA 12. Kommentointimahdollisuuden tarkistus ja kommentointilomakkeen kutsuminen sivulle.

3.5 Alatunniste

Alatunnisteessa suljetaan `body`- ja `html`-tagit. Lisäksi sinne sijoitetaan `wp_footer()`-komento juuri ennen sulkevaa `body` tagia, sillä osa Wordpressin ja lisäosien javascript-kutsuista voidaan sijoittaa ylätunnisteen sijaan alatunnisteeseen. Näiden lisäksi alatunnisteessa ei yleensä ole juuri muuta sisältöä kuin teksti "Powered by Wordpress" ja mahdollisesti teeman tekijän nimi ja linkki tekijän sivuille.

Useissa ilmaisissa teemoissa ei ole mahdollisuutta muokata alatunnisteen sisältöä ilman että osaa lukea teeman lähdekoodia ja tietää mihin kohtaan on

kirjoitettava saadakseen alatunnisteeseen haluamansa tekstin. Joissain teemoissa on kuitenkin toteutettu ylläpitäjää varten mahdollisuus muokata alatunnisteen tekstiä hallintapaneelin kautta siten, että teemaan ohjelmoitu oletusteksti korvautuu käyttäjän haluamalla tekstillä tai se lisätään oletustekstin alapuolelle.

3.6 Funktiot ja asetukset

Funktioille tarkoitettussa functions.php-tiedostossa kutsutaan joitain Wordpressissa valmiina olevia ominaisuuksia, jotka löytyvät melkeinpä jokaisesta ajatuksella tehdystä teemasta. Tällaisia ominaisuuksia ovat esimerkiksi artikkelikuvat, vimpaimet tai itse muokattava otsakekuva. Wordpressin omia ominaisuuksia voi lisätä helposti `add_theme_support()`-komennolla (kuva 13).

```
add_theme_support('post-thumbnails');  
add_theme_support('custom-header', $args);
```

KUVA 13. Wordpressin sisältämien valmiiden ominaisuuksien kutsuminen `add_theme_support()`; funktiolla.

Komennon avulla voidaan kuvan mukaisesti kutsua Wordpressin sisältämiä funktioita, joiden avulla teema tässä tapauksessa alkaa tukea artikkelikuvia ja itse muokattavaa otsakekuvaa. Artikkelikuvien lisääminen teemaan vaatii pelkästään niille tehdyn funktion kutsumista, mutta muokattavaa otsakekuvaa varten tarvitaan array-lista, jossa määritellään muun muassa teemassa käytettävä oletuskuva (kuva 14).

```

$args = array(
    // Default image
    'default-image'      => get_template_directory_uri() . '/images/banner.png',

    // Set height and width, with a maximum value for the width.
    'height'            => 360,
    'width'             => 980,

    // Support flexible height and width.
    'flex-height'       => true,
    'flex-width'        => false
);

```

KUVA 14. Otsakekuvan asetuksien määrittely array-listan avulla.

Koska sivuston URL-osoitetta ei tiedetä etukäteen, on järkevintä kutsua teeman osoite komennolla ja sitten lisätä sen perään kansion ja halutun otsakekuvan nimi, jotta teema varmasti tulostaa näkyviin oikean kuvan. Lisäksi voidaan määritellä otsakekuvan oletuskoko ja esimerkiksi halutaanko kuvan pysyvän täsmälleen tietyn kokoisena vai voiko ylläpitäjä halutessaan rajata pienemmän kuvan kuin mitä oletuskooksi on määritelty.

Kaikkia teemassa tarvittavia ominaisuuksia ei kuitenkaan löydy Wordpressista valmiina ja näitä ominaisuuksia varten pitää tehdä omat funktionsa, jotka myös tulevat functions.php-tiedostoon. Toisinaan asetuksia ja muita erilaisia ominaisuuksia tulee kuitenkin niin paljon, että on syytä jakaa eri ominaisuuksien funktiot omiin tiedostoihinsa, jotta koodia olisi helpompi hallita. Näitä tiedostoja sitten kutsutaan functions.php-tiedostossa, jotta ominaisuudet tulisivat voimaan (kuva 15).

```

//Theme Options
require_once ( get_stylesheet_directory() . '/inc/theme-options.php' );

```

KUVA 15. Teeman asetuksien kutsuminen erillisestä tiedostosta functions.php tiedostoon.

Jotta Wordpressin hallintapaneeliin saataisiin tehtyä yksi sivu teemakohtaisia asetuksia varten, tarvitaan neljä funktiota. Yhdessä rekisteröidään asetukset

(kuva 16), toisessa luodaan sivu hallinatapaneeliin, kolmannessa määritellään sivun sisältö ja neljännessä tarkistetaan asetussivulla syötetyt tiedot, niin ettei käyttäjä voi syöttää vääränlaista tietoa johonkin tiettyyn kenttään.

```
function mm_register_settings() {  
    // Register settings and call sanitation functions  
    register_setting('mm_theme_options', 'mm_options', 'mm_validate_options');  
}  
add_action('admin_init', 'mm_register_settings');
```

KUVA 16. Teemakohtaisten asetusten rekisteröinti.

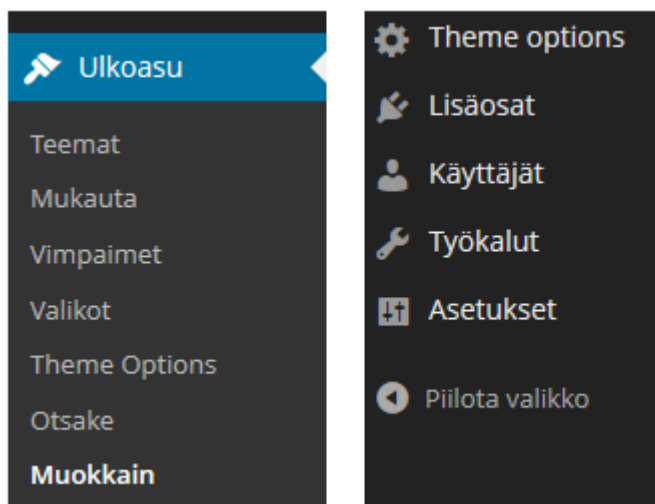
Asetussivun voi kuitenkin luoda kahdella tavalla (kuva 17). Ensimmäinen tapa on käyttää `add_theme_page()`-komentoa, jolla asetussivun saa lisättyä välilehdeksi hallintapaneelin Ulkoasu-kohdan alle. Toinen vaihtoehto on käyttää `add_menu_page()`-komentoa, jolloin asetukset ilmestyvät kokonaan omaksi kohdaksi hallintapaneelin valikkoon (kuva 18).

```
function mm_theme_options() {  
    // Add theme options page to the admin menu  
    add_theme_page('Theme Options', 'Theme Options', 'edit_theme_options', 'theme_options', 'mm_theme_options_page');  
  
    //create new top-level menu  
    add_menu_page('Theme options', 'Theme options', 'administrator', 'theme_options', 'mm_theme_options_page', '', 61);  
}  
add_action('admin_menu', 'mm_theme_options');
```

KUVA 17. Esimerkit eri tavoista luoda sivu asetuksia varten.

Tämän jälkeen luodaan funktiot asetussivun sisällölle ja asetusten tarkistukselle, jotka tässä esimerkissä ovat nimeltään `mm_theme_options` ja `mm_validate_options`. Asetussivun sisällön määrittelevä funktio voidaan luoda kuten mikä tahansa sivu, PHP:n ja HTML:n avulla. Jos käyttäjälle esimerkiksi halutaan antaa mahdollisuus valita, kummassa reunassa mahdollinen sivupalkki sijaitsee, voidaan luoda sivulle tarvittavat radiobutton-elementit sisältävä lomake, josta ylläpitäjä voi valita haluamansa vaihtoehdon. Kun ylläpitäjä sitten on painanut päivitä painiketta, sivu tekee tarkistuksen siitä käskettiinkö asetukset päivittää ja olivatko tiedot tarkistusfunktion mukaan oikein. Riippuen tarkistuksen tuloksesta, käyttäjälle näytetään viesti siitä, että asetusten

tallentaminen onnistui tai vaihtoehtoisesti virheilmoitus siitä, että joitain tietoja ei virheellisten syötteiden vuoksi voitu tallentaa.



KUVA 18. Kuvakaappaus hallintapaneelin näkymästä. Vasemmalla `add_theme_page()`- ja oikealla `add_menu_page()`-komennolla tehty näkymä.

3.7 Kieliversiot

Englantia käytetään internetissä laajasti monella sivustolla, mutta muut kieliryhmätkin on silti hyvä ottaa huomioon tekemällä teemasta kieliversiot eri kielille. Wordpressissa itsessään on olemassa jo kieliversiot, jotka kääntävät suuren osan teemoissa käytetyistä elementeistä valmiiksi, esimerkiksi kommenttien lähetyksen painikkeen teksti näkyy suomeksi jos sivuston ylläpitäjällä on käytössään suomenkielinen versio Wordpressista. Vastaavasti jos käytössä on englanninkielinen Wordpress, painikkeen teksti näkyy englanniksi.

Joihinkin osiin teemasta joutuu kuitenkin sisällyttämään omia tekstejä, jotka vaativat teemakohtaisen kieliversion näkyäkseen myös muilla kielillä. Hyvä esimerkki tästä ovat esimerkiksi teemakohtaiset asetukset, jotka rakennetaan hallintapaneeliin itse. Asetussivua luodessa, sille tarvitsee määritellä nimi, joka voi englanniksi olla esimerkiksi "Theme Options". Ilman teeman omaa kieliversiota, tämä nimi näkyy Wordpressin hallintapaneelissa vain englanniksi, riippumatta siitä minkä kielinen Wordpress ylläpitäjällä on käytössään.

Kieliversiotiedoston tekeminen ilman valmista pohjaa on yllättävän yksinkertaista. Functions.php-tiedostossa tarvitsee vain load_theme_textdomain()-komennon avulla määritellä tunniste, jonka avulla käännettävät kohdat teemasta merkitään. Lisäksi komennon avulla määritellään myös kansio, josta teeman kieliversiot löytyvät (kuva 19). Tunnisteessa on suositeltavaa välttää välilyöntejä, sillä välilyönnit tunnisteessa saattavat toisinaan aiheuttaa virhetilanteita käännöksen toimivuuden suhteen (Catswhocode.com 2009, jälkikommentti).

```
load_theme_textdomain('translate', TEMPLATEPATH.'/'lang' );
```

KUVA 19. Käännettävien kohtien merkitsemiseen vaadittu tunniste ja kansio johon kieliversioiden tiedostot laitetaan.

Tämän jälkeen määritellään oletuskieli, joka voidaan kutsua get_locale()-komennon avulla (kuva 20). Ilman lisämääryksiä, komento asettaa oletuskieleksi englannin (Wordpress Codex 2013). Tämän jälkeen tarkistetaan vielä ovatko kielitiedostot luettavissa niille tarkoitetusta kansioista ja jos ovat, oikeaa kieliversiota kutsutaan kansioista sen mukaan millä kielellä ylläpitäjä käyttää Wordpressia. Jos oikeaa kieltä ei löydy kansioista, käyttäjälle näkyy teeman koodissa käytetty oletuskieli.

```
$locale = get_locale();  
$locale_file = TEMPLATEPATH."/"lang/$locale.php";  
if (is_readable($locale_file) )  
    require_once($locale_file);
```

KUVA 20. Get_locale() –komento ja kieliversioiden tarkistus.

Kun tarvittavat koodirivit on lisätty functions.php-tiedostoon, teeman tiedostot pitää käydä läpi ja merkitä kaikki käännöstä vaativat kohdat. Kohdat merkitään _e- ja __ (kaksi alaviivaa) -funktioiden avulla (Catswhocode.com 2009). Lisäksi tekstien yhteyteen merkitään myös aiemmin functions.php-tiedostoon määritelty tunniste, jonka avulla käännettävät kohdat pystytään tunnistamaan muun koodin joukosta (kuva 21).

```
<h2><?php _e("Nothing Found", "translate"); ?></h2>
<?php edit_post_link(__('Edit', "translate"), '|', ' '); ?>
```

KUVA 21. Funktiot joiden avulla käännettävät tekstit merkitään.

Ensimmäisenä näkyvää `_e`-funktiota käytetään niin sanotusti irrallisten HTML:n seassa olevien tekstinpätkien merkitsemiseen. Kahdella alaviivalla taas merkitään ne tekstinpätkät, jotka ovat PHP-koodin seassa. Kun kaikki käännettävät kohdat on tällä tavoin merkitty, voidaan teeman tiedostot vain pakata zip-paketiksi, joka voidaan syöttää iconlocalize.com -sivustolta löytyvään työkaluun, joka käy tiedostot läpi ja generoi po-tiedoston, joka sisältää kaikki käännöstä vaativat kohdat (kuva 22).

Generated PO files

PO file, generate from: 'Teema.zip'. 36 lines, 89 words

```
msgid ""
msgstr ""
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"

#. Text in function
#: Teema/page-fullwidth.php:35
#: Teema/index.php:54
#: Teema/comments.php:16
#: Teema/page.php:37
msgid "%s comments"
msgstr ""

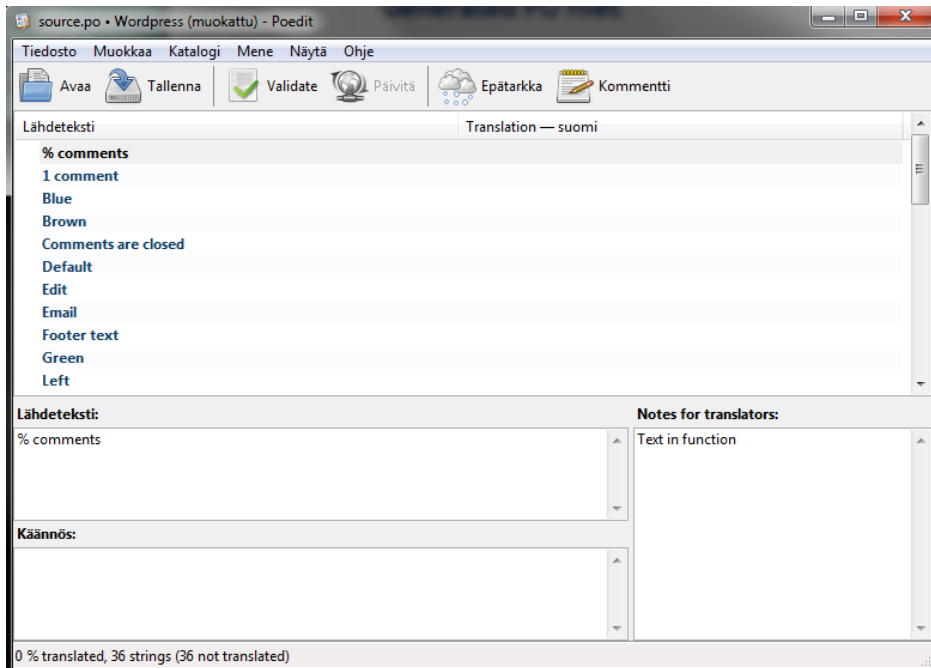
#. Text in function
#: Teema/page-fullwidth.php:35
#: Teema/index.php:54
```

Download: translate.po.gz

KUVA 22. Valmiiksi generoitu po-tiedosto. Tiedosto ladataan koneelle alareunan linkistä.

Kun tiedosto on generoitu, sen voi tallentaa koneelleen ja avata PoEdit-ohjelmassa. Ohjelman avulla käännettävät kohdat voi kääntää jopa muutamassa minuutissa, riippuen siitä kuinka paljon käännettävää teemassa on (kuva 23). Käännöksen jälkeen tiedosto tallennetaan nimellä ja nimetään se halutun kieliversioon mukaan, esimerkiksi `fi.po` eli suomenkielinen versio. Joidenkin kielten kohdalla tiedoston nimi voi vaatia kaksikin koodilyhennettä, esimerkiksi `en_US.po` tarkoittaisi englantia siten kuin sitä USA:ssa puhutaan.

Halutulle kieliversiolle tarvittavat kieli- ja maakoodit, löytää esimerkiksi gnu.org - sivustolta löytyvästä Gettext-manuaalista (Gnu.org 2013).







KUVA 23. Kuvakaappaus PoEdit-ohjelmassa avatusta kielitiedostosta.

Tallennuksen yhteydessä PoEdit luo tiedostosta myös mo-version. Sekä po- että mo-tiedosto lisätään teeman tiedostoista löytyvään lang -kansioon, joka aiemmin määriteltiin functions.php-tiedostossa kieliversioiden kansioksi. Jos kaikki kohdat tiedostosta on käännetty, teeman suomenkielisen version pitäisi toimia ilman ongelmia. Muilla kielillä taas näkyy koodissa käytetty oletuskieli.















3.8 Kansiorakenne

Wordpressin teemat sijaitsevat wp-content -kansiossa olevan themes-kansion alla. Asennettujen teemojen sisältämät tiedostot näkyvät omista kansioistaan ja usein kansion nimi on sama kuin itse teemalla, mutta pienellä kirjoitettuna (kuva 24).

 twentyfourteen	Tiedostoka...	12.12.2013 22:0...	flcdmpe (0...	1681 33
 twentyten	Tiedostoka...	6.12.2013 22:14...	flcdmpe (0...	1681 33
 twentythirteen	Tiedostoka...	11.11.2013 1:03...	flcdmpe (0...	1681 33
 twentytwelve	Tiedostoka...	11.11.2013 1:04...	flcdmpe (0...	1681 33

KUVA 24. Wp-content kansiossa olevan themes kansion sisältö.

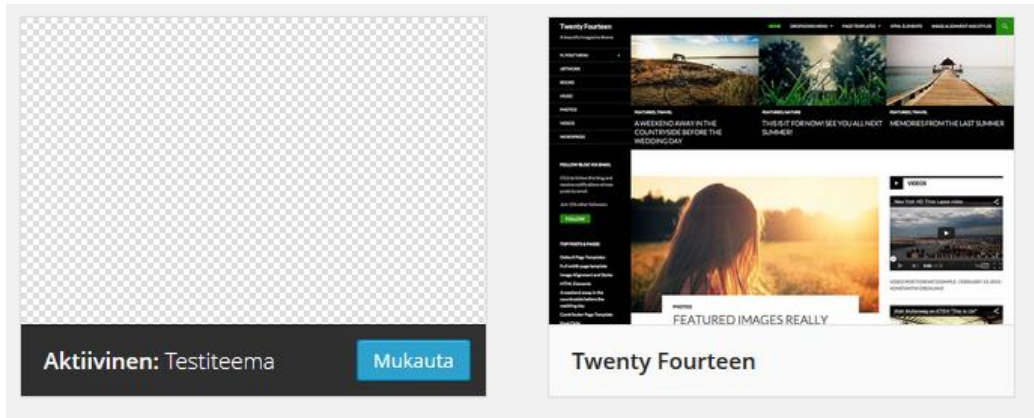
Itse teeman kansiorakenne on hyvin yksinkertainen, sillä kaikki tärkeimmät tiedostot, kuten functions.php ja sivupohjat jäävät kansion juureen, mistä Wordpress hakee niiden sisältämät tiedot. Kaikki muu taas voidaan lajitella teeman sisällä omiin kansioihinsa, jotta rakenne pysyy selkeänä ja helposti hallittavana (kuva 25).

 css	Tiedostoka...	14.2.2014 13:33...	flcdmpe (0...	1681 1681
 images	Tiedostoka...	23.1.2014 21:14...	flcdmpe (0...	1681 33
 inc	Tiedostoka...	30.1.2014 14:37...	flcdmpe (0...	1681 1681
 js	Tiedostoka...	6.12.2013 15:11...	flcdmpe (0...	1681 33
 404.php	620	PHP-tiedo...	19.2.2014 12:59...	adfrw (0755) 1681 1681
 comments.php	1 328	PHP-tiedo...	19.2.2014 19:08...	adfrw (0755) 1681 1681
 footer.php	185	PHP-tiedo...	17.1.2014 21:58...	adfrw (0755) 1681 33
 functions.php	2 887	PHP-tiedo...	20.2.2014 17:54...	adfrw (0755) 1681 33
 header.php	1 057	PHP-tiedo...	19.2.2014 18:45...	adfrw (0755) 1681 33
 index.php	2 873	PHP-tiedo...	18.2.2014 17:00...	adfrw (0755) 1681 33
 page.php	1 804	PHP-tiedo...	16.2.2014 16:03...	adfrw (0755) 1681 1681
 sidebar.php	998	PHP-tiedo...	4.2.2014 13:09:09	adfrw (0755) 1681 33
 single.php	2 389	PHP-tiedo...	19.2.2014 13:35...	adfrw (0755) 1681 1681
 style.css	5 890	CSS Docu...	19.2.2014 19:28...	adfrw (0755) 1681 33

KUVA 25. Yksinkertaisen teeman kansiorakenne.

Jos tyylitiedostoja on useita, jätetään oletustyyli tiedosto yleensä kansion juureen ja kaikki muut tyylitiedostot taas laitetaan omaan kansioonsa teeman sisällä. Samalla tavoin omiin kansioihinsa menevät myös kuvat ja mahdolliset javascript-tiedostot. Ainoa kuva, joka kansion juureen on syytä jättää, on

screenshot.png eli kuvakaappaus teemasta. Ilman kuvakaappausta Wordpressin hallintapaneelissa ei näy teemasta esimerkkikuvaa (kuva 26).



KUVA 26. Kuvakaappaus hallintapaneelin näkymästä, jossa Testiteema ei sisällä screenshot nimistä kuvaa ja Twenty Fourteen teema sisältää.

4 Wordpress-teeman toteutus

Teeman tekemisessä lähdettiin liikkeelle ensimmäisenä Wordpressin asennuksesta, joka tehtiin arkku.net-sivustolta saatuun ilmaiseen sivutilaan. Kun asennus oli tehty, lähdettiin teemaa rakentamaan siten, että luotiin tarvittavat PHP- ja tyylitiedostot Notepad-ohjelmalla ja siirrettiin ne FileZilla-ohjelman avulla käytössä olevaan sivutilaan. Tämän jälkeen teeman rakentamisessa ja muokkaamisessa käytettiin pääasiassa Wordpressin hallintapaneelista löytyvää muokkainta, jolloin tehdyt muutokset saatiin näkyviin vain yhdellä klikkauksella.

Sekä muuttujien nimissä että muissa teeman koodiin sisältyvissä teksteissä käytettiin kielenä englantia. Työn lopuksi teeman sisältämät tekstit käännettiin vielä suomeksi, joten teemaan sisältyy kaksi kieliversiota. Muille kielille teemaa ei käännetty, koska käännöksen suhteen ei haluttu luottaa netistä löytyviin käännskoneisiin joiden toiminnasta ei voi olla täysin varma.

4.1 Tyylitiedostot

Tyylitiedostoja teeman lopulliseen versioon tehtiin yhdeksän kappaletta. Oletustyyli löytyvät teemassa style.css-tiedostosta, joka sisältää kaikki teeman kokomääritykset, div-elementtien asettelut, kuvien koot, responsiivisuuden määritykset ja muut tiedot. Loput kahdeksan tyylitiedostoa taas ovat värimäärityksiä varten.

Koska teemaan tehtiin yhtenä ominaisuutena värivaihtoehdot, piti elementtien värimääritykset tallentaa erilleen muista tyylimäärityksistä. Värivaihtoehdojen tyylitiedostot ovat muuten identtisiä keskenään, mutta vain värikoodit ovat joka tiedostossa erilaiset. Taustavärinä käytetään jokaisessa vaihtoehdossa valkoista, mutta oletusvaihtohtona oleva sinivihreä voidaan kuitenkin teeman asetuksissa korvata vihreällä, sinisellä, ruskealla, punaisella, oranssilla, pinkillä tai violetilla. Nämä värivaihtoehdot päätettiin sisällyttää teemaan mukaan, jotta mahdollisimman moni löytäisi joukosta mieleisensä värin.

Responsiivisuuden vuoksi teema sovitettiin neljään eri kokoon joista kaksi on tarkoitettu isommille näytöille. Isoimmassa koossa on maksimileveydeksi annettu 960 pikseliä ja ulkoasun leveys ei kasva sen leveämmäksi, mikäli näytön tai selainikkunan leveys on enemmän kuin 1024 pikseliä.

Tähän ratkaisuun päädyttiin testilaitteiden puuttumisen vuoksi, sillä työtä tehdessä ei ollut käytössä isoja näyttöjä, niin että olisi varmuudella voinut testata miltä ulkoasu isommilla näytöillä näyttää. Alle 1024 pikselin levyisillä näytöillä sivuston leveydeksi taas on asetettu 95 %, jolloin saatiin käytettyä mahdollisimman suuri osa näytöstä hyödyksi ilman, että sivuston ympärille jää hirveästi turhaa tilaa tai ettei ylimääräistä tilaa jää lainkaan.

Kaksi pienempää kokoa taas on tarkoitettu älypuhelimille, tableteille ja iPadeille. Kun näytön tai selainikkunan koko menee alle 768 pikselin, sivuston valikko korvautuu alavetovalikolla, josta voi valita sivun, jolle haluaa mennä. Näin sivustoa on helppo selata myös älypuhelimilla, sillä kosketusnäytöillä erilaiset hover-efektit eivät välttämättä toimi halutulla tavalla.

4.2 Ylätunniste

Ylätunnisteessa ensimmäisenä on tietenkin avaustagi html-elementille. Tämän jälkeen seuraa head-elementti ja avaustagi body-elementille. Head-elementin sisällä on annettuna tärkeimmät metatiedot, joista osa tarvitaan, jotta teema toimisi esimerkiksi älypuhelimissa halutulla tavalla (kuva 27). Monissa älypuhelimissa resoluutio on isompi kuin itse näytön leveys ja pituus, joten esimerkiksi ilman viewport-määritystä, sivuston ulkoasu skaalautuu puhelimen näytölle hyvin pieneksi ja käyttäjän pitää itse suurentaa sivua, jotta näkisi lukea tekstin. Kun taas viewport-määritykset löytyvät metatiedoista, ulkoasu skaalautuu näytölle sopivan kokoiseksi. Metatietoihin on myös hakukoneoptimoinnin vuoksi kutsuttu kuvaus, jonka ylläpitäjä pystyy sivustostaan antamaan Wordpressin hallintapaneelissa, sillä Raittilan (2013)

mukaan, kuvaus voi näkyä esimerkiksi Googlen hakutuloksissa lyhyenä kuvauksena sivun sisällöstä.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="description" content="<?php bloginfo('description'); ?>"/>
<meta name="author" content="Southpaw"/>
<meta name="HandheldFriendly" content="True">
<meta name="MobileOptimized" content="320">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=yes">
```

KUVA 27. Kuvakaappaus teeman metatiedoista.

Title-tagien väliin taas on kutsuttu sivuston nimi ja yksittäisen sivun nimi, sillä hakukoneoptimoinnin vuoksi tätä menettelyä suositeltiin Raittilan (2013) kirjoittamassa oppaassa koskien hakukoneoptimointia. Näin kävijä voi hakukoneen kautta löytää suoraan haluamalleen sivulle, sen sijaan että hänen pitäisi itse käydä koko sivusto läpi löytääkseen kaipaamansa tiedon. Title-elementin jälkeen on vielä kutsuttu tyylitiedostoja, joista oletustyylytiedosto on kutsuttu Wordpressin omalla komennolla. Väreihin vaikuttava tyylitiedosto taas on kutsuttu paikalle kutsumalla ensin teeman URL-osoite ja kutsumalla sen perään teeman asetuksissa määräytyvää merkkijonoa (kuva 28). Jos ylläpitäjä esimerkiksi on asettanut väriksi vihreän, kutsutaan teeman osoitteen perään merkkijono `"/css/green.css"`. Näin teema saa haettua oikean tyylitiedoston oikeasta kansioista.

```
<title><?php bloginfo('name'); ?> <?php wp_title("|", true); ?></title>
<link rel="stylesheet" type="text/css" media="all" href="<?php bloginfo('stylesheet_url'); ?>" />
<link rel="stylesheet" type="text/css" media="all" href="
<?php echo get_template_directory_uri(); ?><?php echo get_option("theme_color_css"); ?>" />
```

KUVA 28. Kuvakaappaus teeman title-elementistä ja tyylitiedostojen kutsumisesta.

Lopuksi on vielä `wp_head()`-komento, joka kutsuu esimerkiksi lisäosien tyylit ja javascript-tiedostot ylätunnisteeseen, ennen kuin head-tagit suljetaan. Tämän jälkeen on ylätunnisteeseen laitettu kaikki sellaiset elementit, joiden on tarkoitus toistua joka sivulla. Tällaisia ovat esimerkiksi otsakekuva ja navigointi. Ennen

niiden kutsumista sivulle on luotu kuitenkin allwrap- ja pagewrap-elementit, jotka pitävät sisällään kaiken muun sisällön mitä sivulla on. Näiden kahden elementin avulla responsiivisuus on saatu toteutettua mahdollisimman yksinkertaisesti.

Koska sivuston nimeä ei tiedetä etukäteen, on ulkoasu tarkoituksella toteutettu niin, että sivuston otsikko ja mahdollinen kuvaus ovat erillään otsakekuvasta. Sivuston otsikosta on myös tehty linkki, jota klikkaamalla pääsee halutessaan takaisin etusivulle (kuva 29).

```
<div id="pagetitle">
  <a href="<?php echo home_url(); ?>"><h1 class="title"><?php bloginfo('name'); ?></h1></a>
  <p class="description"><?php bloginfo('description'); ?></p>
</div><!--#pagetitle-->
```



KUVA 29. Kuvakaappaus sivuston otsikon toteutuksesta ja siitä miltä otsikko näyttää käytännössä.

Otsikon alapuolella on alkuperäisten suunnitelmien mukaisesti otsakekuva, jota ylläpitäjä pystyy halutessaan vaihtamaan Wordpressin hallintapaneelissa (kuva 30). Otsakekuva pysyy aina samankokoisena, vaikka ylläpitäjä rajaisi haluamastaan kuvasta pienemmän alueen, kuin mitä otsakekuvan koko on. Kuten sivuston otsikostakin, myös otsakekuvaa klikkaamalla pääsee etusivulle niin halutessaan.

Teema

Lorem ipsum



KUVA 30. Kuvakaappaus teemassa käytetystä otsakekuvasta.

Otsakekuvan alapuolella on lopulta navigointi. Jotta teema saatiin responsiiviseksi, navigointi piti kutsua sivulle kahteen kertaan siten, että div-elementin sisällä oleva navigointi on oletuksena näkyvissä. Pienempiin näyttöihin siirryttäessä div-elementin sisällä oleva navigointi kuitenkin piilotetaan näkyvistä tyylitiedoston määrittelyissä ja näkyviin tulee nav-tagien sisällä oleva navigointi, niin että se saadaan tulostettua näkyviin alavetovalikkona (kuva 31).

```
<div id="mainnavi">
  <?php wp_nav_menu($defaults); ?>
<div class="spacer"></div>
</div><!--#mainnavi-->
<nav><?php wp_nav_menu($defaults); ?></nav>
```

Esimerkkisivu Esimerkkisivu 2 Esimerkkisivu 3

...

KUVA 31. Kuvakaappaus navigoinnin koodista ja siitä miltä valikot näyttävät käytännössä.

4.3 Sivupohjat

Sivupohjia teemaan tuli kuusi kappaletta. Jokaisessa sivupohjassa on tietyt peruselementit, jotka toistuvat joka sivupohjassa ja tällaisia ovat esimerkiksi ylätunnisteen, alatunnisteen ja sivupalkin kutsuminen. Lisäksi jokainen sivupohja sisältää silmukan, jolla sisältö kutsutaan näkyviin. Ainoastaan 404-sivupohja, ei sisällä silmukkaa vaan sen sijaan tiedosto sisältää virheilmoituksen, joka näytetään, mikäli kävijä kirjoittaa esimerkiksi URL-osoitteen väärin tai yrittää päästä sivulle, joka on ylläpitäjän toimesta poistettu (kuva 32).

Mitään ei löytynyt

Pahoittelut, etsit jotakin mikä ei ole täällä.

[Palaa etusivulle](#)

KUVA 32. Kuvakaappaus 404.php -tiedoston näyttämästä virheilmoituksesta.

Oletussivupohjana toimiva index.php listaa näkyviin kaikki artikkelit. Tiedostossa kutsutaan ensin while-silmukan sisällä näkyviin artikkelikuva, mikäli sellainen on näytettävälle artikkelille tai sivulle asetettu. Kuvan kutsumisen yhteydessä kutsutaan myös functions.php-tiedostossa olevaa funktiota, jonka avulla tarkistetaan onko kuva vaakasuorassa vai pystysuorassa (kuva 33). Tarkistuksen perusteella artikkelikuvalla asetetaan koko, jossa se näytetään. Näin kuvien laatu pysyy hyvänä, kun niitä ei kaikkia aseteta täsmälleen samankokoiseksi.

```
function is_portrait($url){
    $imagesize=(getimagesize($url));
    if ($imagesize[0] >= $imagesize[1]){return true;}
    else{return false;}
}
```

KUVA 33. Kuvakaappaus funktiosta joka tarkistaa onko kuva pysty- vai vaakasuorassa.

Vasta artikkelikuvan kutsumisen jälkeen lähdetään kutsumaan varsinaista sisältöä, alkaen artikkelin tai sivun otsikosta. Otsikon alapuolelle on myös kutsuttu päivämäärä ja aika jolloin artikkeli on lähetetty ja vasta sen alapuolelle on kutsuttu varsinainen teksti the_content()-komennolla. Artikkelin alapuolelle on kutsuttu lisää tarkempia tietoja, kuten kategoria, avainsanat ja kommenttien määrä. Vasta näiden tietojen jälkeen on lopetettu silmukka endwhile-komennolla. Silmukan päätyttyä, tiedoston loppuun on tehty vielä div-elementti artikkelien sivutusta varten, sillä ylläpitäjä varmasti haluaa jakaa artikkelit sivuihin, sen sijaan, että sivua pitäisi selata loputtomasti alaspäin (kuva 34).

Lorem ipsum 4

11.3.2014 | 15:41 | [Muokkaa](#)

Etiam sollicitudin quis felis non iaculis. Mauris bibendum, diam eu consectetur condimentum, tellus augue adipiscing eros, nec vehicula elit orci ac eros. Nunc rutrum odio ante, vitae iaculis nulla iaculis quis. Phasellus eget molestie turpis, ut tristique velit. Vivamus tincidunt luctus elit in tempus. Nam et porta libero. Suspendisse elementum diam quis erat semper venenatis. Duis hendrerit nisl vitae ornare molestie. Nam non diam ut quam sagittis mollis. Nullam auctor bibendum lorem id imperdiet. Donec vitae ipsum quam. Integer eleifend ac ipsum sed hendrerit. Nam eget sapien in ipsum ornare auctor et non tortor. Ut volutpat justo lacus, ut mollis tortor dapibus nec. Morbi non ornare lacus.

 Hölpöti, Toka, Yleinen
 avain, sana, testiä
 Ei kommentteja

[< Vanhemmat artikkelit](#)

KUVA 34. Kuvakaappaus siitä miltä oletussivupohja näyttää käytännössä.

Yksittäisen artikkelin sivupohja puolestaan on hyvin samanlainen kuin oletussivupohja. Ainoat erot oletussivupohjaan näkyvät sivutuksessa ja artikkelin tiedoissa. Koska yksittäisen artikkelin sivupohjassa kutsutaan kommentit näkyville niin, että kävijä voi lukea niitä, artikkelin tiedoissa näytetään vain kategoria ja avainsanat. Kommenttien määrän näyttämiseksi ei ole tarvetta, sillä kommenttien määrä näkyy kommenttipohjassa otsikkona. Oletussivupohjassa oleva sivutus on puolestaan korvattu sillä, että kävijä pääsee halutessaan katsomaan edellistä tai seuraavaa artikkelia (kuva 35). Koska artikkelit voivat toisinaan olla pitkiä, on sivutus laitettu sekä yläreunaan, että alareunaan. Näin kävijä voi jotain tiettyä sisältöä etsiessään siirtyä seuraavaan artikkeliin jo yläreunasta sen sijaan, että selaisi sivua alaspäin.

Lorem ipsum 4

11.3.2014 | 15:41 | Muokkaa

Etiam sollicitudin quis felis non iaculis. Mauris bibendum, diam eu consectetur condimentum, tellus augue adipiscing eros, nec vehicula elit orci ac eros. Nunc rutrum odio ante, vitae iaculis nulla iaculis quis. Phasellus eget molestie turpis, ut tristique velit. Vivamus tincidunt luctus elit in tempus. Nam et porta libero. Suspendisse elementum diam quis erat semper venenatis. Duis hendrerit nisl vitae ornare molestie. Nam non diam ut quam sagittis mollis. Nullam auctor bibendum lorem id imperdiet. Donec vitae ipsum quam. Integer eleifend ac ipsum sed hendrerit. Nam eget sapien in ipsum ornare auctor et non tortor. Ut volutpat justo lacus, ut mollis tortor dapibus nec. Morbi non ornare lacus.

Hölpöti, Toka, Yleinen
avain, sana, testiä

KUVA 35. Kuvakaappaus yksittäisen artikkelin pohjasta käytännössä.

Lisäksi teemassa on mukana yksittäisen sivun pohja. Koska yksittäinen sivu tulee todennäköisesti pitämään sisällään esimerkiksi yhteystiedot tai vaikkapa yhteydenottolomakkeen, ei yksittäisellä sivulla ole tarvetta minkäänlaiselle sivutukselle tai yksityiskohtaisempien tietojen kuten avainsanojen näyttämislle. Siksi yksittäisen sivun pohja on kuin riisuttu versio yksittäisestä artikkelista (kuva 36). Sama pätee myös Full Width -sivupohjaan, joka myös vaikuttaa yksittäisiin sivuihin. Full Width -sivupohjassa ei kuitenkaan kutsuta sivupalkkia, vaan sivu on leveydeltään koko ikkunan mittainen.

Esimerkkisivu

Muokkaa

Tämä on esimerkkisivu. Se eroaa artikkelista siten, että se pysyy paikallaan ja näkyy sivustosi navigointivalikossa (useimmissa teemoissa). Useimmat aloittavat Tietoja-sivulla, joka esittelee sivuston kirjoittajan mahdollisille lukijoille, esimerkiksi näin:

Moro! Päivisin jaan postia, iltaisin olen menestyvä stand-up-koomikko. Tämä on minun blogini! Asun Tampereella kissan kanssa ja tykkään Sihi-juomasta.

...tai jotain tällaista:

XYZ Härpäkeyritys Oy perustettiin vuonna 1971 ja on siitä lähtien ollut maan merkittävin härpäkkeiden toimittaja. Kotipaikkamme on Ankkalinn ja työllistämme yli 3000 ihmistä.

Olet uusi WordPress-käyttäjä, joten kypäsepa [hallintänäköymääsi](#) poistamaan tai muokkaamaan tätä sivua, ja luomaan uusia sivuja. Pidä hauskaa!

KUVA 36. Kuvakaappaus yksittäisen sivun pohjasta käytännössä.

Viimeinen sivupohjista on arkistosivun pohja, jonka mukaan artikkelit näytetään, jos kävijä haluaa saada näkyviin vaikkapa tietyn avainsanan omaavat artikkelit. Sen sijaan, että artikkelin tai sivun koko sisältö tulostettaisiin näkyviin kuten muissa sivupohjissa, siitä kutsutaan näkyviin vain artikkelikuvan pienennetty versio tai mikäli artikkelikuvaa ei ole asetettu, kutsutaan näkyviin harmaalla tehty valokuvaa muistuttava kuva, joka ilmaisee kävijälle, että artikkelikuvaa ei ole tälle artikkelille asetettu. Kuvan viereen oikealle puolestaan on kutsuttu artikkelin otsikko, aika jolloin se on lähetetty, sekä artikkelin 25 ensimmäistä sanaa, niin että kävijä saa käsityksen siitä mistä artikkelissa on kyse (kuva 37).

MAALISKUU 2014



Lorem ipsum 4

11.3.2014 | 15:41

Etiam sollicitudin quis felis non iaculis. Mauris bibendum, diam eu consectetur condimentum, tellus augue adipiscing eros, nec vehicula elit orci ac eros. Nunc rutrum odio...



Lorem ipsum 2

11.3.2014 | 14:37

Ut tempus ut urna non dictum. Praesent est nunc, lobortis nec ipsum id, viverra luctus turpis. Sed imperdiet odio fermentum dignissim volutpat. Vestibulum ante ipsum...

KUVA 37. Kuvakaappaus arkistosivun pohjasta käytännössä.

4.4 Kommentit

Kommenttipohja on tehty Googlen kautta löydettyjen ohjeiden mukaan, joten ohjeiden mukaisesti kommenttipohjan alussa tarkistetaan, ettei kävijä yritä saada näkyviin itse kommenttipohjaa tai salasanasuojattuja kommentteja. Vasta tämän jälkeen alkaa varsinainen kommenttien pohja, jossa ensin tarkistetaan onko kommentointi suljettu vai avoinna. Jos kommentointi ei ole mahdollista, kävijä näkee tiedon siinä kohtaa artikkelia, missä kommenttien pitäisi näkyä (kuva 38).

Kommentointi on suljettu

KUVA 38. Kuvakaappaus ilmoituksesta jonka kävijä näkee jos artikkelia ei voi kommentoida.

Jos taas kommentointi on mahdollista, niin tarkistetaan onko artikkeliin lähetetty kommentteja. Jos on, teema tulostaa kommenttien sivutuksen, jolla kommentteja pystyy selaamaan ja tietenkin itse kommentit. Sivutus löytyy sekä kommenttien ylä- että alapuolelta, sillä joillain sivuilla kommentteja voi olla samalla sivulla paljonkin. Osa kävijöistä on tottunut siirtymään sivulta toiselle yläreunasta ja osa taas alareunasta.

Tässä ratkaisussa, kommenttien kutsuminen on hoidettu Wordpressin omalla `wp_list_comments()`-komennolla. Koska Wordpressin omat asettelut kommentteille eivät kuitenkaan olleet aseteltavissa halutulla tavalla, on tähän teemaan tehty `functions.php`-tiedostoon funktio, jolla asettelut on saatu halutunlaiseksi (kuva 39).

```
<ol class="commentlist">
    <?php wp_list_comments( array( 'callback' => 'custom_comment' ) );?>
</ol><!--.commentlist-->
```

KUVA 39. Kuvakaappaus custom_comment-funktion kutsumisesta kommenttipohjassa.

Kommenttien näyttämiseen tarvittavat tiedot, kuten avatar-kuva, vastauslinkki, ylläpitäjälle näkyvä muokauslinkki sekä lähetysaika, on tehty Wordpressiin sisältyvillä komennoilla ja sen jälkeen muokattu CSS-määritysten avulla. Funktioon myös sisältyy tarkistus, jossa katsotaan, odottaako kommentti moderointia, sillä jotkut ylläpitäjät haluavat roskakommenttien suodattamiseksi hyväksyä lähetetyt viestit käsin. Jos kommentti siis odottaa hyväksymistä, kävijä näkee siitä ilmoituksen (kuva 40).

```
<div class="comment-data">
<?php //if comment is under moderation, this line will be shown
if ($comment->comment_approved == '0') { ?>
    <em><?php _e("Your comment is awaiting moderation.", "translate"); ?></em>
<?php } else{ comment_text(); } ?>
</div><!--.comment-data-->
```



Kommenttisi odottaa moderointia.

Testaaja
26.3.2014 | 19:21
[Vastaa](#)

KUVA 40. Kuvakaappaus hyväksymistarkistuksesta ja ilmoituksesta jonka käyttäjä näkee, jos kommentti odottaa hyväksyntää.

Koska yksittäiseen kommenttiin on mahdollista myös vastata, joko ylläpitäjän tai muiden kävijöiden toimesta, teema tehtiin myös tukemaan sisennettyjä kommentteja. Jos siis johonkin viestiin on vastattu siihen tarkoitettu linkistä, vastaukset näkyvät kyseisen viestin alapuolella hieman sisennettyinä, niin että tietyille kommentoijalle tarkoitettua vastaukset näkyvät hieman erillään tavallisista kommentteista. Kommenttien näyttämisen lisäksi kommenttipohja sisältää vielä kommentointilomakkeen, joka kävijän on täytettävä jos edes haluaa kommentoida. Jos kävijällä on sivustolle tunnukset, hänelle näkyy vain Wordpressin oma tekstikenttä, johon kommentin voi kirjoittaa (kuva 41).

Vastaa

Kirjaututtu sisään käyttäjänä [Southpaw](#). [Kirjautu ulos?](#)

Kommentti

Voit käyttää näitä HTML-tageja ja attribuutteja: ` <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike> `

KUVA 41. Kuvakaappaus kommentointilomakkeesta, joka näkyy sisään kirjautuneelle kävijälle.

Sivustolla käyville vieraille käyttäjille näkyy kuitenkin laajempi lomake, jossa kysytään nimi, sähköposti, kotisivu ja itse kommentti. Myös tässä käytetään Wordpressin omaa lomaketta, mutta sitä ei kuitenkaan ole kutsuttu Wordpressin omalla komennolla. Lomakkeeseen haluttiin rivivaihdot ja pakollisia kohtia merkitsevät tähdet punaisiksi. Tämän vuoksi Wordpressissa käytössä olevat oletuslomakkeen kentät oli kopioitava sellaisenaan ja lisättävä halutut class-määrytykset tageihin käsin (kuva 42).

Vastaa

Sähköpostiosoitettasi ei julkaista. Pakolliset kentät on merkitty *

Nimi*

Sähköposti*

Kotisivu

Kommentti

Voit käyttää näitä HTML-tageja ja attribuutteja: <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike>

Lähetä kommentti

KUVA 42. Kuvakaappaus muokatusta kommentointilomakkeesta.

4.5 Sivupalkki

Sivupalkki toimii tässä teemassa vimpainalueena ja sen vuoksi se piti ensin rekisteröidä käyttöön functions.php-tiedostossa, siihen erikseen tarkoitettulla komennolla. Komennon avulla määriteltiin muun muassa vimpainalueen nimi, joka ylläpitäjällä näkyy hallintapaneelissa ja lisäksi komennolla voi luoda myös div-elementin jonka sisään jokainen vimpain asetetaan (kuva 43). Näin myös yksittäiselle vimpaimelle pystyy myöhemmin asettamaan tyylimäärytyksiä.

```

//Registers widget area
function arphabet_widgets_init() {
    register_sidebar( array(
        'name' => 'Sidebar 1',
        'id' => 'default_sidebar',
        'before_widget' => '<div id="widget">',
        'after_widget' => '</div>',
        'before_title' => '<h3 id="widget_title">',
        'after_title' => '</h3>',
    ) );
}
add_action( 'widgets_init', 'arphabet_widgets_init' );

```

KUVA 43. Kuvakaappaus funktiosta, jonka avulla vimpainalue rekisteröidään niin, että se näkyy hallintapaneelissa.

Vimpainalueen rekisteröinnin jälkeen sivupalkkiin piti lisätä ainoastaan tarkistus sille, onko se rekisteröity ja kaiken muun Wordpress hoitaa automaattisesti (kuva 44). Sivupalkki ilmestyi näkyviin hallintapaneeliin ja siihen pystyi raahaamaan Wordpressiin sisäänrakennettuja vimpaimia näkyviin, kuten esimerkiksi haku-kentän, jonka avulla sivustolta voi hakea sisältöä erilaisilla hakusanoilla.

```

<?php //widget area ?>
<?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar('Sidebar 1') ) : ?>

<?php endif; ?>

```

KUVA 44. Kuvakaappaus vimpainalueen merkitsemisestä sidebar-php-tiedostoon.

Teemaan haluttiin tehdä myös oma vimpain, joka näyttää sivupalkissa muun muassa linkin RSS-syötteeseen ja erilaiset sosiaalisen median linkit, jotka ylläpitäjä mahdollisesti haluaa linkittää profiiliinsa (kuva 44). Sosiaalisen median linkit haetaan teeman asetuksista siten, että teema tarkistaa onko ylläpitäjä antanut esimerkiksi Facebook-profiilinsa osoitetta ja jos on, teema tulostaa näkyville Facebookin logon, jota klikkaamalla pääsee ylläpitäjän profiiliin.

```

<?php
    $facebook = get_option("facebook_url");
?>
<div id="widget">
    <a href="<?php bloginfo('rss2_url'); ?>" target="_blank">
    
    </a>

<?php if (empty($facebook)){
//Do nothing since facebook value is empty
} else { ?><a href="<?php echo $facebook; ?>" target="_blank">
    
    </a>
<?php } ?>

```

KUVA 44. Kuvakaappaus sivupalkin koodista, jossa tulostetaan näkyville RSS-syötteen linkki ja Facebook-profiilin linkki.

Ellei ylläpitäjä ole antanut minkään profiilin osoitetta, sosiaalisen median viimpaimessa näkyy pelkästään RSS-syötteen linkki, josta kävijä voi halutessaan tilata sivuston artikkelit, niin että saa uusista artikkeleista ilmoituksen kun niitä ilmestyy. Vastaavasti jos ylläpitäjä on antanut hallintapaneelissa kaikkien mahdollisten profiiliensa osoitteet, linkit tulostuvat riviin ja tilan loppuessa aloittavat uudelta riviltä (kuva 45).



KUVA 45. Kuvakaappaus sosiaalisen median linkeistä, jotka teemaan on mahdollista saada näkyviin.

4.6 Alatunniste

Alatunnisteessa on kaikista teeman tiedostoista vähiten sisältöä, sillä sinne on laitettu vain valmis teksti, joka sisältää linkin Wordpressin sivuille ja sen lisäksi on kerrottu, kuka teeman on tehnyt. Näiden lisäksi on wp_footer()-komennolla kutsuttu alatunnisteeseen lisäosien ja Wordpressin mahdolliset javascript-kutsut, sillä osa javascript-koodista tulee ylätunnisteeseen ja osa alatunnisteeseen.

Osa ylläpitäjistä saattaa kuitenkin haluta lisätä oman tekstin alatunnisteeseen, mutta ei osaa kirjoittaa sitä koodimuotoon. Tämän vuoksi teemaan on toteutettu ominaisuus, jonka avulla ylläpitäjä voi hallintapaneelissa kirjoittaa haluamansa tekstin, joka sitten ilmestyy alatunnisteeseen näkyviin. Ylläpitäjän asettama teksti kutsutaan alatunnisteeseen `get_option()`-komennon avulla (kuva 46).

```
<?php echo get_option("custom_footer_text"); ?>
```

KUVA 46. Kuvakaappaus komennosta, jolla ylläpitäjän asettamaa alatunnisteen tekstiä kutsutaan.

Viimeisenä ennen `body`- ja `html`-tagien sulkemista, alatunnisteessa on vielä JavaScript-tarkistus, joka ottaa selvää, onko sivustolla käyvällä kävijällä käytössään kosketusnäyttö vai ei. Jos kävijällä on kosketusnäyttö ja iso resoluutio, erilaiset hover-efektit eivät toimi ja tämä aiheuttaa ongelmia sivuston valikon kanssa, mikäli ylläpitäjä on asettanut navigoinnin linkeistä aukeamaan alavalikoita. Alavalikot vain käyvät nopeasti näkyvissä eikä kävijä ehdi klikata linkkejä päästäkseen haluamalleen sivulle. Tämän vuoksi alatunnisteessa lisätään navigoinnin linkkeihin jQuerya käyttäen `aria-haspopup` -määrittys, jonka avulla valikot aukeavat näkyviin tarpeeksi pitkäksi aikaa (kuva 47). Tämä ominaisuus saatiin kuitenkin toimimaan vain joillakin selaimilla.

```
<script>
//check if user is using touch screen
var isTouch = (('ontouchstart' in window) || (navigator.msMaxTouchPoints > 0));

if (isTouch) {
    //If touchscreen device isn't showing dropdown menu for some reason, this should fix
    //hover effect which isn't always working with touch screens.
    jQuery(document).ready(function () {
        jQuery(".page_item").has("ul").children("a").attr("aria-haspopup", "true");
    });
} else { }
</script>
```

KUVA 47. Kuvakaappaus `aria-haspopup` -määrittymisen lisäämisestä navigoinnin linkkeihin.

4.7 Asetukset

Asetusten toteuttaminen aloitettiin asetussivun luomisesta. Koska asetuksia varten ei tarvittu kuin yksi sivu, päätettiin se lisätä omaksi kohdaksi hallintapaneelista löytyvän Ulkoasu-kohdan alle. Sivun luominen tehtiin siten, että ensin tarkistetaan, että asetussivulle pyrkivä kävijä on ylläpitäjä, eikä joku ulkopuolinen. Tämän jälkeen rekisteröitiin asetuksia varten tarvittavat funktiot siihen tarkoitetulla omalla funktiolla (kuva 48).

```
if (is_admin()) : // Load only if we are viewing an admin page

function mm_register_settings() {
    // Register settings and call sanitation functions
    register_setting('mm_theme_options', 'mm_options', 'mm_validate_options');
}
add_action('admin_init', 'mm_register_settings');
```

KUVA 48. Kuvakaappaus asetuksia varten vaadittavien funktioiden rekisteröinnistä.

Vasta rekisteröinnin jälkeen luotiin theme_options-funktio, joka lisäsi asetussivun näkyviin hallintapaneeliin (kuva 49). Itse sivun sisältö taas laitettiin kokonaan omaan funktioonsa, jota sitten kutsuttiin theme_options-funktiossa.

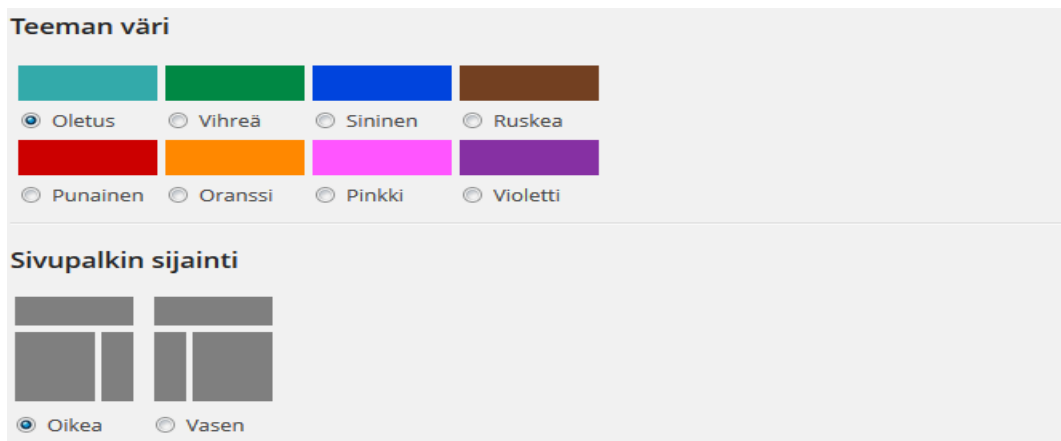
```
function mm_theme_options() {
    // Add theme options page to the admin menu
    add_theme_page(__('Theme Options', "translate"), __('Theme Options', "translate"),
        'edit_theme_options', 'theme_options', 'mm_theme_options_page');
}
add_action('admin_menu', 'mm_theme_options');
```

KUVA 49. Kuvakaappaus sivun lisäämisestä näkyville Wordpressin hallintapaneeliin.

Itse sivun sisällön luominen ei juuri eroa esimerkiksi sivupohjan luomisesta. Sisältö vain kirjoitetaan siihen tarkoitetun funktion sisälle. mm_theme_options_page-funktion sisälle lähdettiin järjestelmällisesti luomaan sisältö alkaen otsikosta. Otsikon alapuolelle luotiin form-elementti eli lomake,

jonka ylä- ja alareunaan tehtiin Wordpressin omalla komennolla Tallenna asetukset -painike. Näin käyttäjän ei tarvitse selata sivua, joka kerta alareunaan asti, halutessaan tallentaa esimerkiksi lomakkeen ensimmäiseen kohtaan muutetut tiedot.

Vasta tämän jälkeen päästiin itse asetuksiin, jotka luotiin pää-asiassa radiobutton-elementeillä ja tekstikentillä. Ylläpitäjä pystyy valitsemaan haluamansa värivaihtoehdon ja sivupalkin sijainnin vain parilla klikkauksella, sen sijaan, että hänen pitäisi itse ryhtyä muokkaamaan teeman koodia saadakseen teemasta mieleisensä (kuva 50).



KUVA 50. Kuvakaappaus värivaihtoehdoista ja sivupalkin sijainnista käytännössä.

Radiobutton-elementeistä näkyy aktiiviseksi valittuna se vaihtoehto, joka sillä hetkellä on käytössä. Tämä on toteutettu yksinkertaisella tarkistuksella, joka elementin näyttämisen yhteydessä tarkistaa värivaihtoehto-asetuksen arvon. Jos arvo on sama kuin kyseisellä radiobutton-elementillä, elementti asetetaan käytössä olevaksi (kuva 51).

```
<input type="radio" name="colorchoice" value="/css/default.css"
<?php if($theme_color == "/css/default.css"){echo "checked";} else{ }?>
<?php _e('Default', "translate") ?></td>
```

KUVA 51. Kuvakaappaus radiobutton-elementin koodista.

Myös sosiaalisen median linkit ja oman alatunnisteen tekstin, pystyy asettamaan asetuksissa. Ylläpitäjä voi syöttää profiiliensa linkit niille tarkoitettuihin tekstikenttiin. Tallennuksen yhteydessä teemassa tarkastetaan, että Facebook-kenttään syötetystä osoitteesta todella löytyy merkkijono facebook.com (kuva 52). Tarkistus myös karsii linkeistä välilyönnit pois ja hyväksyy tyhjät syötteet.

```
//Selects the previous value, to restore it in case an invalid entry has been given
$prev = get_option("facebook_url");
//Get given value and strip spaces out
$facebook = esc_attr($_POST["url_facebook"]);
$facebook = str_replace(' ', '', $facebook);
//Verifies the given value
if (strpos($facebook, 'https://www.facebook.com/') !== false || empty($facebook)){
    update_option("facebook_url", $facebook); }
else{
    update_option("facebook_url", $prev);
    $errors++;}
```

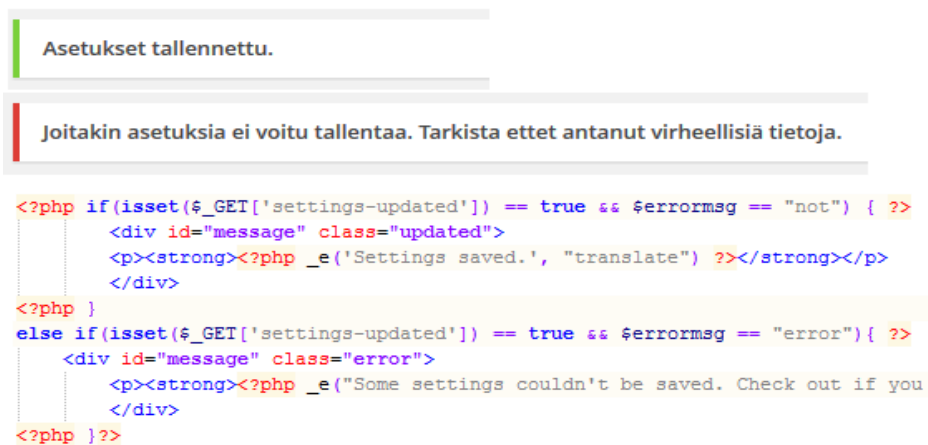
KUVA 52. Kuvakaappaus facebook-profiilin osoitteen tarkistuksesta.

Asetussivusta ja tarkistusfunktioista ei kuitenkaan ole mitään iloa, ellei tarvittavia asetuksia ole luotu. Asetuksia varten tarvittavat asetukset rekisteröitiin functions.php-tiedostossa siihen tarkoitettulla add_option-komennolla (kuva 53). Käytännössä lomakkeen joka kohdalle piti tehdä oma asetuksensa tällä komennolla, jotta asetukset saatiin toimimaan halutulla tavalla.

```
//Adds options for settings in theme-options.php
add_option('theme_color_css', '/css/default.css');
add_option('sidebar_location', 'right');
add_option('facebook_url');
add_option('twitter_url');
add_option('tumblr_url');
add_option('instagram_url');
add_option('pinterest_url');
add_option('google_url');
add_option('youtube_url');
add_option('custom_footer_text');
add_option('option_errormsg');
```

KUVA 53. Kuvakaappaus asetuksien luomisesta functions.php -tiedostossa.

Oletuksena jokainen asetus pitää sisällään alkutilanteessa vain tyhjän merkkijonon. Ylläpitäjän antamien syötteiden tarkistuksessa asetuksia kuitenkin päivitetään update_option-komennolla ja muualla teemassa niitä kutsutaan get_option-komennolla. Asetuksien tarkistuksen yhteydessä myös lasketaan virheellisten syötteiden määrä, jolle jouduttiin lopulta tekemään oma asetuksensa, jotta tieto saatiin siirrettyä funktiosta toiseen ongelmitta. Viimeisenä asetussivun sisällön joukkoon, lisättiin otsikon alapuolelle tarkistus, jossa virheellisten syötteiden määrä tarkistetaan ja tuloksesta riippuen näytetään virheilmoitus tai ilmoitus siitä että asetukset on päivitetty onnistuneesti (kuva 54).



```
<?php if(isset($_GET['settings-updated']) == true && $errmsg == "not") { ?>
    <div id="message" class="updated">
    <p><strong><?php _e('Settings saved.', "translate") ?></strong></p>
    </div>
<?php }
else if(isset($_GET['settings-updated']) == true && $errmsg == "error"){ ?>
    <div id="message" class="error">
    <p><strong><?php _e("Some settings couldn't be saved. Check out if you
    </div>
<?php }?>
```

KUVA 54. Kuvakaappaus ylläpitäjän näkemistä ilmoituksista ja tarkistuksesta jonka perusteella ne näytetään.

5 Teeman lisääminen teemakirjastoon

Teeman lisääminen teemakirjastoon tapahtuu Wordpressin sivuilta. Ennen teeman lisäämistä on kuitenkin hyvä luoda testiympäristö, jossa käytetään Wordpressin tarjoamaa testidataa teeman testaamiseen (.Wordpress Codex 2014). Testidata on tallennettuna xml-tiedostoksi, joka tuodaan Wordpressiin, niin, että asennuksessa näkyy laajasti erilaista sisältöä, jolla katsotaan, että ulkoasu toimii mahdollisimman hyvin. Lisäksi on luotava Wordpressin sivuille tunnukset, sillä teemoja ei pysty lähettämään ilman rekisteröitymistä.

Wordpressilla on myös oma Review Team eli tarkistusryhmänsä teemoille ja lisäosille. Ryhmään voi liittyä kuka tahansa ja ryhmän liittymisohjeissa mainitaan monia erilaisia lisäosia, joiden avulla voi testata oman teemansa toimivuutta (Wordpress.org 2014). Luomalla itselleen testiympäristön ja testaamalla teemaa myös lisäosien avulla, voi karsia suuren osan mahdollisista virheistä ja ongelmista ja ennen kuin teeman lähettää teemakirjastoon. Kun teeman tarkistus on tehty, sen voi lähettää teemakirjastoon Wordpressin sivuilla (kuva 55).

Add Your Theme To The Directory

Information about Theme development is available on the Codex at [Theme Development](#). For questions about Theme development please use the [Themes and Templates forum](#).

Make sure to review the guidelines at [Theme Review](#) before uploading a Theme.

We will be reviewing your Theme using the sample data available in the WordPress export file available at [Theme Unit Test](#). Before uploading your Theme file please test it with this sample export data.

Select your zipped theme file

 Ei valittua tiedostoa.

(Maximum allowed filesize: 8M)

KUVA 55. Kuvakaappaus teeman lisäysnäköymästä.

Jotta teeman saisi lähetettyä, se on ensin pakattava zip-paketiksi, joka sitten lähetetään sivustolle. Sivusto tekee teeman sisältämälle koodille jo lähetyksen aikana tarkistuksen ja riippuen tuloksesta, sivusto näyttää ilmoituksen siitä läpäisikö teema tarkastuksen vai ei. Jos teema ei läpäissyt tarkastusta, sivusto näyttää myös virheilmoitukset, joiden vuoksi näin tapahtui (kuva 56).

Add Your Theme To The Directory

Results of Automated Theme Scanning: **Fail**

- **REQUIRED:** `.sticky` css class is needed in your theme css.
- **REQUIRED:** `.bypostauthor` css class is needed in your theme css.
- **REQUIRED:** No content width has been defined. Example:

```
if ( ! isset( $content_width ) ) $content_width = 900;
```

- **REQUIRED:** Could not find the `comment-reply` script enqueued. See: [Migrating Plugins and Themes to 2.7/Enhanced Comment Display](#)

```
<?php if ( is_singular() ) wp_enqueue_script( "comment-reply" ); ?>
```

- **REQUIRED:** Could not find `wp_link_pages`. See: [wp_link_pages](#)

```
<?php wp_link_pages( $args ); ?>
```

- **RECOMMENDED:** could not find the file `readme.txt` in the theme. Please see [Theme Documentation](#) for more information.
- **RECOMMENDED:** No reference to `add_theme_support("custom-background", $args)` was found in the theme. If the theme uses background images or solid colors for the background, then it is recommended that the theme implement this functionality.
- **RECOMMENDED:** No reference to `add_editor_style()` was found in the theme. It is recommended that the theme implement editor styling, so as to make the editor content match the resulting post output in the theme, for a better user experience.

KUVA 56. Kuvakaappaus teemasta joka ei läpäissyt sivuston tekemää tarkastusta.

Jos teema ei läpäise tarkastusta, kaikki punaisella merkityt kohdat on korjattava teemasta, jotta se täyttäisi teemakirjaston teemoille asetetut kriteerit. Kun alkutarkastus kuitenkin lopulta menee läpi, teemasta luodaan tiketti, joka sisältää teeman lähdekoodin ja teeman yleistiedot (kuva 57). Tämän jälkeen joku tarkastustiimin jäsenistä voi varata tiketin itselleen ja tarkistaa teeman Wordpressin antamien kriteerien mukaan, sillä automaattinen tarkistus ei pysty tarkistamaan kaikkea.

Priority: **new theme** Keywords: **theme-nature-light**

Cc: marimyl@...

Description

Nature Light - 1.0 ↳ Reply

Light theme for anyone in need of simple layout. Different colour options and chance to decide sidebar location.

Theme URL -
Author URL -

SVN - <https://themes.svn.wordpress.org/nature-light/1.0>
ZIP - <https://wordpress.org/themes/download/nature-light.1.0.zip?nostats=1>

History:

Ticket	Summary	Status	Resolution	Owner
#17782	THEME: Nature Light - 1.0	new		

(this ticket)

KUVA 57. Kuvakaappaus teeman tiketistä.

Kun teema on tarkistettu, sen tarkistaja antaa tikettiin palautteen, jonka perusteella teema on joko hylätty tai hyväksytty. Tarkistuksessa kiinnitetään tarkasti huomiota pieniinkin asioihin, joten vielä tässä vaiheessakin voi ilmetä virheitä, jotka on korjattava ennen kuin teeman voi lähettää uudelleen arvioitavaksi (kuva 58).

Credit Links

OK

Screenshot

- *screenshot.png* file is sized inappropriately (it should maintain 4:3 aspect ratio, the best resolution for it is 880x660)

Theme Check

- Delete *thumbs.db* Windows thumbnail
- **Recommended** - instead of using `TEMPLATEPATH` use `get_template_directory()` (*functions.php*, lines 122 and 125)

Code Quality

- *header.php* - You've hardcoded charset type here. Use proper function instead.
- *footer.php* - `wp_footer()` tag should be placed immediately before closing HTML *body* tag
- *footer.php* - No scripts should be hardcoded here (these must be enqueued and hooked in appropriately)
- *functions.php* - All custom theme functions must be prefixed with 'theme-slug
- *functions.php* - Theme options should be added to database via single options array, not separately

KUVA 58. Kuvakaappaus tarkistajan tekemistä huomioista.

Seuraavaksi teemasta korjataan nämä tarkistajan huomioimat virheet ja lähetetään se uudelleenarvioitavaksi. Kun teema lopulta hyväksytään, se lisätään teemakirjastoon ja sen voi tämän jälkeen löytää ja ladata teemakirjastosta sekä Wordpressin hallintapaneelin, että sivujen kautta. Teema ilmestyy näkyviin myös omaan profiiliin ja Themes-linkin takaa löytyvän My themes-kohdan taakse. (kuva 59).

Your Themes

Nature Light



2014-04-04 - 1.0

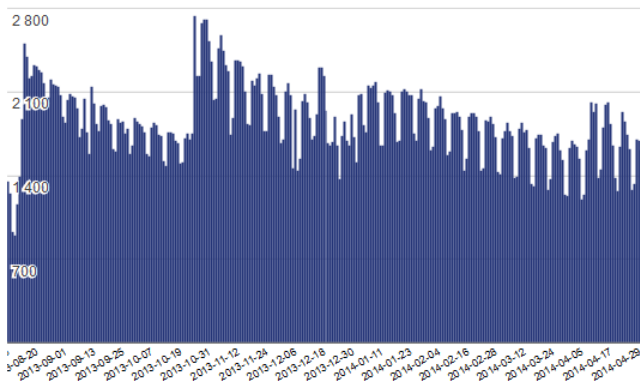
2014-04-26 - 1.1

2014-04-27 - 1.2

KUVA 59. Kuvakaappaus My themes-sivusta, johon lähetetyt teemat listautuvat näkyviin.

Teeman tiedoissa näkyvät välilehtinä erilaiset tiedot jotka teemasta kerätään, esimerkiksi päivittäiset lataukset, teeman käyttäjien tekemät arvostelut ja erilaiset tukipyynnöt, jos jollakulla on tullut ongelmia teeman käyttämisen suhteen (kuva 60). Lisäksi teeman tiedoista voi tarkastella myös teeman aikaisempia versioita.

Downloads Per Day



History

Today	769
Yesterday	1,615
Last Week	11,773
All Time	1,925,943

KUVA 60. Kuvakaappaus Twenty Ten teeman -latausmääristä.

6 Pohdinta

Opinnäytetyön tavoitteena oli luoda pienille sivustoille ja blogeille suunnattu teema. Teeman ulkoasun haluttiin olevan yksinkertainen ja teeman haluttiin sisältävän muun muassa värivaihtoehdot ja sivupalkin sijainnin määrittäminen. Valmiiseen teemaan onnistuttiin toteuttamaan kaikki halutut ominaisuudet. Kaikkiaan teeman luominen oli kuitenkin kohtalaisen helppoa, sillä Codexista löytyi laajasti tietoa teemaan tarvittavista tiedostoista ja eri funktioista ja niiden toiminnasta.

Vaikein osuus teeman tekemisessä olivat teeman asetukset. Asetuksien tekemisestä löytyi useita erilaisia tutoriaaleja ja lähes joka tutoriaalista löytyi aina erilainen ratkaisu, joka ei sitten käytännössä toiminutkaan tekeillä olevassa teemassa. Tässä opinnäytetyössä esitelty ratkaisu saatiin aikaan lopulta monesta eri ohjeesta yhdistelemällä, mutta ongelmaksi muodostuivat Wordpressin kriteerit teemakirjastoon lisättävien teemojen laadusta, sillä usean yksittäisen asetuksen sijaan, asetukset tulisi tallentaa array-listaksi. Tämän ja muutaman muun pienemmän ongelman vuoksi teemaa ei lopulta hyväksytty teemakirjastoon vaan teeman tarkastanut käyttäjä kehotti korjaamaan virheet ja yrittämään sitten uudestaan. Jälkeenpäin ajatellen olisi ollut hyvä ottaa selvää teeman lisäämisestä teemakirjastoon jo ennen työn aloittamista, sillä näin kriteerit olisivat olleet tiedossa heti ja teeman olisi voinut rakentaa alusta asti niiden mukaan.

Opinnäytetyöstä oli kuitenkin monelta osin hyötyä sillä taitoni CSS-määritysten ja PHP-kielen suhteen paranivat entisestään ja lisäksi sain myös paremmin käsityksen siitä mistä eri osista teemat rakentuvat ja miten esimerkiksi Wordpressin sisältämiä ominaisuuksia voi muokata omiin tarkoituksiinsa sopiviksi. Jatkossa teemaa on tarkoitus kehittää siten, että `update_option`-komennolla tehdyt asetukset korvataan Wordpressin vaatimusten mukaisesti array-listalla. Lisäksi teemasta on tarkoitus tehdä myös tumma versio niille, jotka pitävät enemmän mustasta taustaväristä.

LÄHTEET

Cantero, L. 2013. How to optimize WordPress popup-on-hover menus for touch devices. Viitattu 21.3.2014

<http://blogs.msdn.com/b/notime/archive/2013/02/01/how-to-optimize-wordpress-popup-on-hover-menus-for-touch-devices.aspx>

Catswhocode.com 2009. How to make a translatable Wordpress theme. Viitattu 16.3.2014

<http://www.catswhocode.com/blog/how-to-make-a-translatable-wordpress-theme>

CMS Usage Statistics 2014. Viitattu 10.3.2014

<http://trends.builtwith.com/cms>

GNU.org 2013. Introduction to gettext. Viitattu 16.3.2013

<http://www.gnu.org/software/gettext/>

Maes, G. 2008: Unraveling the Secrets of WordPress – Comments.php file. Viitattu 14.3.2014

<http://code.tutsplus.com/articles/unraveling-the-secrets-of-wordpress-commentsphp-file--net-28>

McNeil, P. 2008: The Web Designer's Idea Book – the ultimate guide to themes, trends and styles in website design. HOW Books.

Näkövammaisten Keskusliitto ry 2014. Viitattu 27.3.2014

http://www.nkl.fi/fi/etusivu/nakeminen/julkaisu/nvrek_vuosikirja/1_2_arviot_nv_lu_kumaarasta

O'Dell, J. 2013. 19 Percent of the web runs on Wordpress. Viitattu 18.2.2013

<http://venturebeat.com/2013/07/27/19-percent-of-the-web-runs-on-wordpress/>

Raittila, A. 2013: Hakukoneoptimointi lyhyesti. Viitattu 14.3.2014

<http://nettibisnes.info/hakukoneoptimointi/>

Spooner, C. 2011. How to Create Your Own Custom Wordpress theme. Viitattu 14.2.2014

<http://line25.com/tutorials/how-to-create-your-own-custom-wordpress-theme>

StatCounter Global Stats 2013. Viitattu 14.3.2014

<http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-201301-201312>

Wearecolorblind.com 2012. Viitattu 27.3.2014

<http://wearecolorblind.com/article/a-quick-introduction-to-color-blindness/>

Wordpress Codex 2014. Function reference/get_locale. Viitattu 16.3.2014

http://codex.wordpress.org/Function_Reference/get_locale

Wordpress Codex 2014. Theme Unit Test. Viitattu 4.4.2014

http://codex.wordpress.org/Theme_Unit_Test

Wordpress.org 2014. History. Viitattu 14.3.2014

<http://wordpress.org/about/history/>

Wordpress.org 2014. How To Join WPTRT. Viitattu 4.4.2014

<http://make.wordpress.org/themes/about/how-to-join-wptrt/>

Wordpress.org 2014. Themes Directory. Viitattu: 10.3.2014

<http://wordpress.org/themes/>