Aapeli Poranen

OHJELMOINTI WORDPRESS-YMPÄRISTÖSSÄ

Teemat, lisäosat ja multisite

Opinnäytetyö
Tietojenkäsittelyn ko.

6.3.2015

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 6.3.2015
Tekijä(t) Aapeli Poranen	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittelyn koulutusohjelma
Nimeke WordPress-jukaisujärjestelmän soveltuvuus verkkopalvelujen kehitykseen.	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli käsitellä WordPress-julkaisujärjestelmän soveltuvuutta nykyaikaisten responsiviisten ja dynaamisten sivustojen luontiin ohjelmoijan näkökulmasta. Tavoitteena oli luoda muokattava landing page -tyylinen sivu Observis Oy:n uudelle tuotteelle. Sivuston tuli olla monikielinen, responsiivinen ja sekä sisällöllisesti muokattava.

Opinnäytetyön tehtävänä oli käydä läpi sivun ulkoasun, sisällön sekä lisäosien luonti WordPress-julkaisujärjestelmässä, sekä kertoa hieman WordPressin multisite-ominaisuudesta. Tuloksista havaitsin, että WordPress soveltui nykyaikaiseksi verkkosivualustaksi, sekä sen muokattavuus oli melko rajaton. Lisäksi multisite-ominaisuus antoi mahdollisuuden luoda sivustojen verkon käyttäen vain yhtä tietokantaa.

Johtopäätöksenä oli todettava WordPressin oleva varsin varteenotettava vaihtoehto dynaamisten verkkosivustojen alustana. Sille löytyi paljon valmiita lisäosia ja teemoja sivuston nopeaan muokkaamiseen. Myös oman teeman sekä lisäosan rakentaminen WordPressiin onnistui aikaisemman PHP-kokemuksen ansiosta melko näppärästi.

Asiasanat (avainsanat)

Verkkosivut, CMS, WordPress

Sivumäärä	**Kieli**	**URN**
36	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Arto Väätäinen	**Opinnäytetyön toimeksiantaja** Observis Oy

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 6.3.2015
Author(s) Aapeli Poranen	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

The possibilities of the WordPress publishing software for web-development

Abstract

The purpose of this thesis was to test the capability of the WordPress publishing software in the development of modern dynamic websites with responsive layout. The goal was to create a landing page for Observis Oy's new product. The site had to be a multilingual and responsive web page featuring a content management possibility.

The aim of the thesis was to cover the development processes of a new WordPress theme and plugin as well as to introduce the WordPress multisite feature. The results showed that WordPress was very suitable platform to create modern websites, and its configurability seemed limitless. Also the multisite feature added the possibility to create a network of websites by using only one WordPress installation and database.

In conclusion it could be observed that WordPress is a very noteworthy option for creating modern dynamic websites. There were thousands of themes and plugins to get one started quickly, but developing them was not so hard either.

Subject headings, (keywords)

Websites, CMS, WordPress

Pages	Language	URN
36	Finnish	

Remarks, notes on appendices

Tutor	Bachelor's thesis assigned by
Arto Väätäinen	Observis Oy

SISÄLTÖ

1 JOHDANTO

Tämä opinnäytetyö käsittelee WordPress-julkaisujärjestelmää web-kehittäjän näkökulmasta. Kuinka alun perin blogi-työkaluksi tarkoitetulla WordPressillä saadaan tehtyä näyttäviä nykyaikaisia responsiivisia ja dynaamisia sivustoja, sekä kuinka omien lisäosien luonti WordPressiin onnistuu.

Sain tehtäväkseni luoda landing page -tyylisen sivuston Observis Oy:n uudelle tuotteelle. Sivun oli tarkoitus olla dynaaminen, responsiivinen sekä monikielinen. Pohdin erilaisia tapoja toteuttaa tämä, ja mieleeni tulikin ensimmäisenä tehdä koko sivu alusta alkaen itse tietokannoista ja hallintapuolesta ulkoasuun. Tajusin kuitenkin pian, että tähänhän voi käyttää jo olemassa olevia työkaluja ja sainkin kehotuksen tehdä sivu WordPressiä käyttäen.

WordPress näyttää aluksi asiaan perehtymättömälle melkoiselta sekamelskalta php-tiedostoja. Mitä ihmeen vimpaimia täällä on? Mikä ihmeen functions.php täynnä outoja koodin pätkiä ja miksi index.php sisältää pelkästään php-funktioita? Kuinka saan sivuni näyttämään sellaiselta kuin haluan? WordPressin rakenteen ja toiminnallisuuden ymmärtäminen vie oman aikansa.

Pikkuhiljaa WordPressin hienous tulee esille ja alamme ymmärtää sen taipuvuutta erilaisiin toteutuksiin; kuinka ulkoasu on erillään itse sisällöstä, kuinka paljon WordPressille on saatavana erilaisia lisäosia tuomaan toiminnallisuuksia sivustolle ja miten paljon säästetään aikaa kun dynaaminen verkkosivusto toteutetaan WordPressiä käyttäen.

Kerron aluksi hieman Wordpressistä itsestään ja ohjelmoinnin standardeista WordPress-ympäristössä. Siitä siirryn esittelemään WordPressin toiminnalisuuden muokkaamisen siihen tarkoitetuilla koukuilla, eli hookeilla. Tämän jälkeen kerron oman teeman luonnista WordPressiin. Tästä etenen käsittelemään vimpaimien tarkoitusta WordPressissä ja rakennan oman pienen lisäosan. Viimesenä käsittelen WordPressin multisite-ominaisuutta, jolla voidaan muuttaa yksi WordPress-asennus sivustojen verkoksi.

En siis käsittele WordPressiä järjestelmänä vaan kerron ohjelmoijan näkökulmasta kuinka WordPress muuttuu blogista nykyaikaiseksi verkkosivuksi, miten sivustolle lisätään toiminnallisuuksia. Toivoisinkin että tästä opinnäytetyöstä olisi apua samassa tilanteessa oleville web-ohjelmoijille.

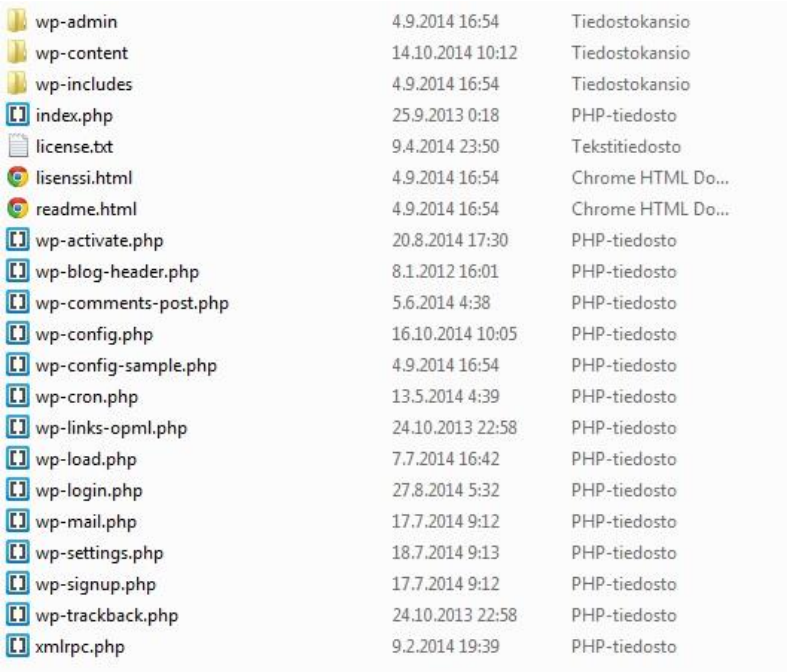
2 WORDPRESS-JULKAISUJÄRJESTELMÄ

WordPress on julkaisualusta blogeille ja verkkosivuille. Sen tarina alkoi 2003 pienestä sisäpiirin koodin pätkästä, mutta se on vuosien saatossa ja suosion kasvaessa saavuttanut nykyisen asemansa maailman suurimpana itse hallinnoitavana verkkojulkaisutyökaluna. Aluksi WordPress oli vain blogaamiseen tarkoitettu työkalu, mutta nykyisin teemojen, lisäosien ja vimpaimien myötä sillä voidaan rakentaa laajojakin verkkosivustoja. (WordPress.org 2014a)

Tässä luvussa kerrotaan, kuinka WordPress muuttuu yksinkertaisesta blogista nykyaikaiseksi verkkosivustoksi, kuinka sen ulkoasua muutetaan ja kuinka siihen lisätään toimintoja. Kaikki muokkaukset voidaan tehdä alusta alkaen itse, tai niihin voidaan käyttää valmiita komponentteja ja kehyksiä.

2.1 Johdatus Wordpress-ohjelmointiin

WordPress on rakennettu PHP-ohjelmointikieltä käyttäen. WordPress-kokonaisuus käsittää ytimen, hallintapaneelin, sisällön (kuva 1) sekä tietokannan. Kun halutaan rakentaa sivusto WordPressillä, keskitytään wp-content-kansioon (sisältö). Täällä sijaitsevat tässä opinnäytetyössä käsiteltävät themes ja plugins -kansiot (kuva 2).



KUVA 1. WordPressin juurikansio

WordPress-järjestelmä on rakennettu siten, että sen omaa koodia ei ole tarkoitus muuttaa, vaan siihen tartutaan ulkoa päin kiinni ja lisätään siihen ominaisuuksia. Tämä estää mm. sen, että ohjelmoijan omat muutokset eivät WordPressin päivityksessä ylikirjoitu. Sivusto säilyy siis muuttumattomana vaikka WordPress päivitetäänkin.

KUVA 2. wp-content-kansion sisältö

WordPress-ohjelmoinnille on olemassa standardit, joita tulisi noudattaa luettavuuden ja yhtenäisyyden kannalta. Näitä ovat muun muassa tiedostojen sekä funktioiden nimeämiset, sisennykset sekä php-tagien käyttö. Kaikki standardit löytyvät osoitteesta http://codex.wordpress.org/WordPress_Coding_Standards.

Aloittelevan WordPress-ohjelmoijan olisi hyvä ymmärtää käsitteet php, html, css sekä javascript. Tulemme huomaamaan kuitenkin että ohjelmointi WordPressin kanssa ei

juuri eroa tavallisesta php-ohjelmoinnista. Järjestelmän kanssa pitää vain osata keskustella, ja tämä tapahtuukin näppärästi WordPressin koukuilla.

2.2 Koukut

Kun WordPressin olemassa olevia toimintoja ja ominaisuuksia halutaan muuttaa, tai halutaan luoda itse uusia, tulevat vastaan niin kutsutut koukut (hooks). Koukkujen perusideana on antaa ohjelmoijalle mahdollisuus muokata WordPressin käyttäytymistä ilman, että sen alkuperäistiedostoja muokataan. Koukku on siis tapahtuma, johon tartutaan, kun halutaan muokata jotain WordPressin toiminnoista ja täten käsitellä sitä ulkoa päin omalla funktiolla. (Cohen 2011.)

Koukkuja on kahteen eri tarkoitukseen; toiminnallisuuteen liittyvät koukut (action hooks), joilla voidaan muokata toiminallisuutta lisäämällä koukkuun omaa koodia, sekä sisältöä manipuloivat koukut (filter hooks), jolla voidaan käsitellä esimerkiksi julkaistua tekstisisältöä. Esittelen seuraavaksi molemmat pienillä esimerkeillä tuettuina. Olemassa olevat koukut ovat listattuna Wordpressin Codexissa.

2.2.1 Action hook

Toiminnallisuuteen liittyvät koukut, kuten aikaisemmin sanoin, antavat mahdollisuuden muokata WordPressin toiminnallisuutta lisäämällä tapahtumaan funktion. Näitä koukkuja käsitellään add_action() sekä do_action()-funktioilla. Esimerkiksi kuvassa 3 luomme sivuillemme uuden julkaisutyypin functions.php-tiedostossa.

```
add_action( 'init', 'create_post_type' );
function create_post_type() {
  register_post_type( 'movie-review',
    array(
      'labels' => array(
        'name' => __( 'Elokuva-arvostelut' ),
        'singular_name' => __( 'Elokuva-arvostelu' )
      ),
      'public' => true,
      'has_archive' => true,
      'supports' => array( 'title', 'editor', 'comments',
                           'excerpt', 'custom-fields', 'thumbnail' )
    )
  );
}
```

Kuva 3. Tartutaan init-koukuun ja lisätään siihen funktio

Kuvassa 3 on lisätty init-koukkuun, eli tapahtumaan, joka ajetaan, kun WordPress on käynnistynyt, create_post_type-funktio. Init on siis valmiiksi määritelty koukku WordPressissä, jota ohjelmoijan ei tarvitse itse ajaa.

Ohjelmoija voi kuitenkin luoda myös omia koukkuja ja kytkeä niihin toiminnallisuuksia. Kuvassa 4 luodaan funktio ja lisätään se itse luotuun koukkuun. Funktio voi olla monimutkainen tai todella yksinkertainen, kuten kuvassa 4 näkyy.

```
function say_hello() {

    echo "Hello everyone! I grabbed my_first_hook -hook.";

}

add_action('my_first_hook','say_hello');
```

KUVA 4. Yksinkertainen funktio, joka sijaitsee functions.php-tiedostossa

Koukku voidaan ”laukaista” halutussa vaiheessa. Tässä esimerkissä koukku laukaistaan index.php-tiedostossa, kun julkaisu haetaan kannasta ja tulostetaan sivulle (kuva 5).

```
<div class="service-box">

    <div class="service-icon popup"
         href="<?php echo get_permalink(); ?>">
        <?php echo get_the_post_thumbnail($post_id,
                                          array(200,200)); ?>
                                          <!-- Image size -->
    </div>
    <div class="service-desc">

        <h5 href="<?php echo get_permalink(); ?>">
            <?php echo get_the_title(); ?></h5>
        <p><?php echo get_the_excerpt(); ?> </p>

        <?php do_action('my_first_hook'); ?>

    </div>
</div>
```

KUVA 5. Koukku voidaan lisätä haluttuun paikkaan sivustolla

Add_action()-funktio siis lisää koodin koukkuun ja do_action() laukaisee sen. Samalla tavalla toimivat myös suodatinkoukut, joista kerrotaan seuraavaksi.

2.2.2 Filter hook

Filter hookeilla, eli suodatinkoukuilla, voidaan manipuloida WordPressin tulostamaa dataa. Koukkuun siis lisätään oma suodatin (filter), joka käsittelee datan ja tekee siihen määritellyt muutokset. Esimerkiksi WordPressin koukku ”the_content” antaa mahdollisuuden käsitellä julkaisun sisältöä. Kuvassa 6 tartutaan the_content-koukkuun ja lisätään julkaisun sisältöön tekstiä.

```
add_filter('the_content', 'add_text_to_content');
function add_text_to_content($content) {

    $content .= "Hi! I came from add_text_to_content filter.";
    return $content;

}
```

Kuva 6. Lisätään jokaisen julkaisun sisältöön omaa tekstiä

Kuvassa 6 siis liityttiin WordPressin the_content-koukkuun, joka ajetaan julkaisun tietokantahaun ja tulostuksen välissä (WordPress Codex 2015d). Filtteri ei siis mene kantaan asti, vaan manipuloi kannasta haettua sisältöä. Tämän jälkeen loimme funktion, joka hakee julkaisun sisällön content-muuttujaan ja lisää siihen pätkän tekstiä. Jos pudottaisimme pisteen pois yhtäsuuruusmerkin edestä, lukisi julkaisussa vain kyseinen tekstin pätkä. Filttereitä voikin käyttää näppärästi vaikkapa kirosanojen suodattamiseen julkaisusta.

2.3 Teemat

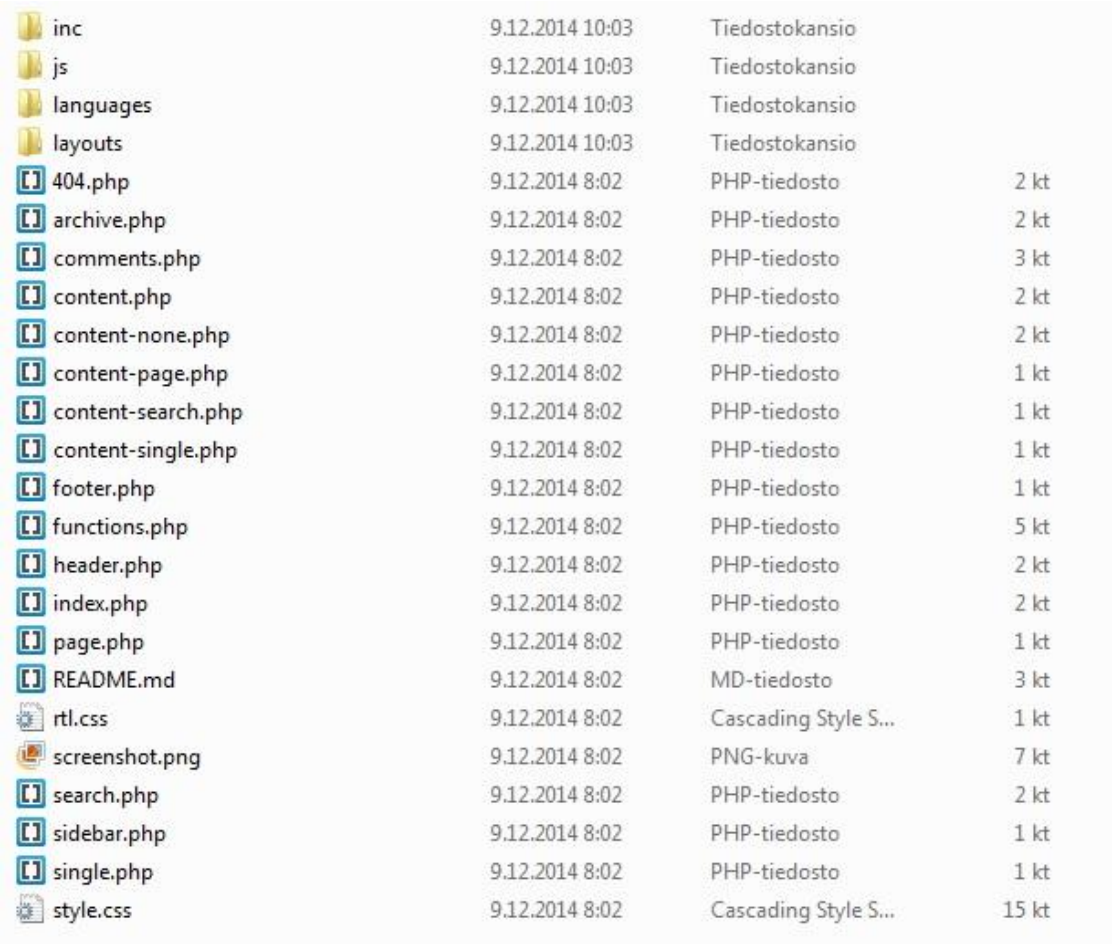
Mikä on teema? Äkkiseltään voisi kuvitella, että teema on vain yksi tyylitiedosto jossa määritellään sivuston ulkoasu. WordPressin verkossa toimivan ohjekirjan, Codexin, mukaan WordPress-teema on kokoelma tiedostoja, jotka yhdessä luovat graafisen käyttöliittymän ja yhtenäisen suunnittelumallin blogille. WordPressin kotisivuilla sijaitsevassa teemakirjastossa ilmaiseksi ladattavia teemoja on yli 3000 kpl (WordPress.org 2014b).

Huomasin tätä opinnäytetyötä tehdessäni kuitenkin, että sopivan valmiin WordPress-teeman löytäminen voi olla todella hankalaa, koska toiminnot, varsinkin ilmaisissa teemoissa, ovat yleensä melko rajallisia. Valmiin teeman muokkaaminen ilman minkäänlaista kokemusta teemoista voi myös osoittautua melko hankalaksi. Siksi suosittelenkin aloittamaan teemoihin tutustumisen ihan ikiomasta teemasta omine ulkoasuineen ja toiminnallisuuksineen.

Perehdyn tässä kappaleessa teeman ohjelmointitekniseen puoleen, en niinkään ulkoasun suunnitteluun. En rakenna teemaa alusta alkaen itse, vaan käytän Underscores (”_s”) nimistä valmista pohjaa teemalle. Underscores-teemapohjan saa vapaasti ladattua osoitteesta http://underscores.me. Underscoresin sivuilla annetaan teemalle nimi. Underscores generoi tämän nimen perusteella kaikki tarvittavat tiedostot toimivaan teemaan ja pakkaa sen zip-pakettiin, joka ladataan koneelle. Esimerkkiteemani nimi onkin kekseliäästi ”my-first-theme”.

2.3.1 Rakenne

Underscores generoi my-first-theme-kansion, jossa on 20 tiedostoa ja 4 alikansiota. Kansiosta löytyy oudompien php-tiedostojen lisäksi myös tutumman kuuloisia tiedostonimiä, kuten index.php, header.php, footer.php sekä style.css. Tässä opinnäytetyössä perehdytään enemmän juuri näihin php-tiedostoihin sekä functions.php -tiedostoon. Kuvassa 7 näkyy Underscoresin luoman teemapohjan rakenne.



inc	9.12.2014 10:03	Tiedostokansio	
js	9.12.2014 10:03	Tiedostokansio	
languages	9.12.2014 10:03	Tiedostokansio	
layouts	9.12.2014 10:03	Tiedostokansio	
404.php	9.12.2014 8:02	PHP-tiedosto	2 kt
archive.php	9.12.2014 8:02	PHP-tiedosto	2 kt
comments.php	9.12.2014 8:02	PHP-tiedosto	3 kt
content.php	9.12.2014 8:02	PHP-tiedosto	2 kt
content-none.php	9.12.2014 8:02	PHP-tiedosto	2 kt
content-page.php	9.12.2014 8:02	PHP-tiedosto	1 kt
content-search.php	9.12.2014 8:02	PHP-tiedosto	1 kt
content-single.php	9.12.2014 8:02	PHP-tiedosto	1 kt
footer.php	9.12.2014 8:02	PHP-tiedosto	1 kt
functions.php	9.12.2014 8:02	PHP-tiedosto	5 kt
header.php	9.12.2014 8:02	PHP-tiedosto	2 kt
index.php	9.12.2014 8:02	PHP-tiedosto	2 kt
page.php	9.12.2014 8:02	PHP-tiedosto	1 kt
README.md	9.12.2014 8:02	MD-tiedosto	3 kt
rtl.css	9.12.2014 8:02	Cascading Style S...	1 kt
screenshot.png	9.12.2014 8:02	PNG-kuva	7 kt
search.php	9.12.2014 8:02	PHP-tiedosto	2 kt
sidebar.php	9.12.2014 8:02	PHP-tiedosto	1 kt
single.php	9.12.2014 8:02	PHP-tiedosto	1 kt
style.css	9.12.2014 8:02	Cascading Style S...	15 kt

KUVA 7. Underscoresin generoima my-first-theme-kansio

Index.php on tiedosto, jonka sisältö näytetään oletuksena WordPress-sivuston etusivulla. Siihen haetaan WordPressin get-funktioilla header.php ja footer.php. Blogi-tyylisessä sivustossa index.php -tiedostoon kannattaa hakea myös sidebar.php, jossa sijaitsee vimpainalue haulle, arkistolle, kategorioille ja muille blogin ominaisuuksille. Kuvassa 8 näkyvät Underscoresin index.php:n oletussisältö sekä get-funktiot.

```
get_header(); ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

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

            <?php /* Start the Loop */ ?>
            <?php while ( have_posts() ) : the_post(); ?>

                <?php
                    /* Include the Post-Format-specific template for the content
                     * If you want to override this in a child theme,
                     * then include a file called content-___.php
                     * (where ___ is the Post Format name)
                     * and that will be used instead.
                     */
                    get_template_part( 'content', get_post_format() );
                ?>

            <?php endwhile; ?>

            <?php my_first_theme_paging_nav(); ?>

        <?php else : ?>

            <?php get_template_part( 'content', 'none' ); ?>

        <?php endif; ?>

        </main><!-- #main -->
    </div><!-- #primary -->

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

KUVA 8. Underscores teeman index.php muokkaamattomana

Index.php on siis teeman oletussivun sivupohja, eli template. Sivupohja on sivun html-rakenne eli miten tieto näytetään sivulla. Uuden sivupohjan luominen on melko helppoa, se tarvitsee tunnistuakseen vain header-osan, minkä jälkeen kirjoitetaan sivulle haluttu koodi. Tiedostot voidaan sijoittaa joko teemakansion juureen, tai niille voidaan tehdä oma kansionsa (WordPress Codex 2014d). Sivupohja otetaan käyttöön WordPressin hallintapaneelissa uutta sivua luodessa. Kuvassa 9 näkyy esimerkki sivupohjasta.

```php
<?php
/*
 * Template Name: My First Custom Page Template
 * Description: Just testing.
 */

get_header(); ?>

<h1><?php the_title(); ?></h1>

<div class="custom_page_content">
    <?php

        if (have_posts()) : while (have_posts()) : the_post();

        the_content();

        endwhile; endif;

    ?>
</div>

<?php
get_footer();
?>
```

KUVA 9. Sivupohja tarvitsee nimen tunnistuakseen hallintapaneelissa

Header.php-tiedostosta löytyy html:ää osaavalle tuttujen tagien, kuten html, head, meta jne. lisäksi myös muutama WordPressin funktio, kuten wp_head(), jolla haetaan wp_head -koukkuun lisätty sisältö, esim. css-tiedostot (kuva 10).

```php
?><!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
<meta charset="<?php bloginfo( 'charset' ); ?>">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="profile" href="http://gmpg.org/xfn/11">
<link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">

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

<body <?php body_class(); ?>>
<div id="page" class="hfeed site">
    <a class="skip-link screen-reader-text" href="#content">
        <?php _e( 'Skip to content', 'my-first-theme' ); ?></a>

    <header id="masthead" class="site-header" role="banner">
        <div class="site-branding">
            <h1 class="site-title">
                <a href="<?php echo esc_url( home_url( '/' ) ); ?>" rel="home">
                    <?php bloginfo( 'name' ); ?></a></h1>
            <h2 class="site-description">
                <?php bloginfo( 'description' ); ?></h2>
        </div><!-- .site-branding -->

        <nav id="site-navigation" class="main-navigation" role="navigation">
            <button class="menu-toggle" aria-controls="menu" aria-expanded="false">
                <?php _e( 'Primary Menu', 'my-first-theme' ); ?></button>
            <?php wp_nav_menu( array( 'theme_location' => 'primary' ) ); ?>
        </nav><!-- #site-navigation -->
    </header><!-- #masthead -->

    <div id="content" class="site-content">
```

KUVA 10. header.php-tiedostossa käytetään WordPressin funktioita tiedon hakuun

Footer.php:ssä näytetään sivun alaosan sisältö sekä haetaan wp_footer -koukkuun lisätyt sisällöt, kuten javascript tiedostot (kuva 11). Kuten huomataan, niin nämä kolme tiedostoa: index.php, header.php ja footer.php eroavat perinteisestä html-rakenteesta vain WordPressin php-funktioilla.

```
    </div><!-- #content -->

    <footer id="colophon" class="site-footer" role="contentinfo">
        <div class="site-info">
            <a href="<?php echo esc_url( __(
                'http://wordpress.org/', 'my-first-theme' ) ); ?>">
                <?php printf( __( 'Proudly powered by %s', 'my-first-theme' ),
                'WordPress' ); ?></a>
            <span class="sep"> | </span>
            <?php printf( __( 'Theme: %1$s by %2$s.', 'my-first-theme' ),
            'my-first-theme',
            '<a href="http://underscores.me/" rel="designer">Underscores.me</a>' );
            ?>
        </div><!-- .site-info -->
    </footer><!-- #colophon -->
</div><!-- #page -->

<?php wp_footer(); ?>

</body>
</html>
```

KUVA 11. footer.php:ssä haetaan wp_footer-koukkuun lisätyt tiedot

Funktioihin liittyen uutena asiana WordPressin teemoittajalle tulee functions.php-tiedosto. Functions.php-tiedostolla saadaan muutettua WordPressin käyttäytymistä, eli se toimii vähän samalla tavalla kuin lisäosa (kuva 12). Nimensä mukaisesti siihen voidaan määritellä funktioita, joita halutaan jälkeenpäin käyttää omassa teemassa, mutta myös kaikki WordPressin sisäänrakennetut funktiot ovat käytettävissä.

```php
/*
* Create custom post type for movie reviews.
* ----------------------------------------
* Execute our create_post_type function when
* init hook is triggered (WordPress has finished loading).
**/
add_action( 'init', 'create_post_type' );
function create_post_type() {
  register_post_type( 'movie_review', //<-- Post type name.
    array(
      'labels' => array(
        'name' => __( 'Elokuva-arvostelut' ), //<-- New menu item in admin panel.
        'singular_name' => __( 'Elokuva-arvostelu' )
      ),
      'public' => true,
      'has_archive' => true,
      'supports' => array( 'title', 'editor', 'comments',
                           'excerpt', 'custom-fields', 'thumbnail' )
    )
  );
}
```

KUVA 12. Luodaan teemallemme uusi julkaisutyyppi functions.php-tiedostossa

Jokaisella teemalla on oma functions.php-tiedostonsa, joka sisältää kyseistä teemaa palvelevia toiminnallisuuksia. Määrittelemäsi toiminnallisuudet siis katoavat, kun vaihdat teemaa (WordPress Codex 2014b).

2.3.2 Ulkoasu

Kuten aikaisemmin jo mainitsin, en aio keskittyä tässä opinnäytetyössä teeman ulkoasun suunnitteluun, vaan kerron miten valmis html-teema saadaan liitettyä WordPressiin. Käytän tässä ohjeessa BootstrapTasten Squad-teemaa, joka löytyy osoitteesta http://bootstrapzero.com/bootstrap-template/squad. Squad on Bootstrap frameworkin päälle tehty, täysin responsiivinen yhden sivun teema.

Aloitetaan kopioimalla Squad-teeman sisältö edellisessä kappaleessa luotuun my-first-theme-kansioon. Näiden kahden teeman kansiorakenteet eroavat hieman toisistaan (esim. tyylitiedostoja sijaitsee useammassa eri kansiossa), jos haluat muuttaa kansiorakennetta haluamaksesi, muista tarkistaa lataamasi teeman tiedostoviitteiden eheys. Seuraavaksi jaetaan Squad-teeman index.html -tiedosto WordPressin header-, footer- ja index.php-tiedostoihin (liite 2). Index.html:ssä linkitetyt tyyli- ja scriptitiedostot linkitetään WordPressissä oikeaoppisesti functions.php:n kautta, niitä

ei siis kopioida suoraan sivulle (Wpbeginner 2013). Tiedostoja liitettäessä niille annetaan nimi ja polku. Lisäksi valinnaisia parametrejä ovat riippuvuudet, versionumero sekä linkitetäänkö tiedosto footer.php:hen (liite 1).

Ulkoasu on nyt liitetty my-first-theme-teemaamme. Voit siis suunnitella teemasi ulkoasun kokonaan erillään WordPressistä ja liittää sen jälkikäteen. Koska index.php-tiedoston sivupohja on suoraan kopioitu staattisesta html-teemasta, sisältöä ei voida vielä WordPressin kautta tuottaa, vaan sivustolla on tällä hetkellä ainoastaan Squad-teeman mukana tullutta html:ään kovakoodattua sisältöä. Seuraavaksi lisätään teemaan ominaisuus sisällön muokkaamiseen.

2.3.3 Sisältö ja asetukset

Options Framework on ilmainen apuohjelma teeman muokkaamiseen. Sillä voidaan muokata niin ulkoasua, kuin myös sisältöä. Sen ominaisuuksiin kuuluvat mm. WYSIWYG-editori, kuvan lataus, sekä värivalitsin (Price 2014). Options Framework on saatavilla niin lisäosana, kuin omana teemanaankin. Yhdistän tässä opinnäytetyössä Options Framework Themen omaan teemaani.

Options Framework Theme on ilmainen niin ei-kaupallisille, kuin kaupallisillekin sivustoille (Price 2014). Ohjelman saa ladattua osoitteesta https://github.com/devinsays/options-framework-theme. Options Framework Theme liitetään omaan teemaamme kopioimalla Options Frameworkin inc-kansio ja options.php my-first-themen juureen. Lisäksi lisätään pätkä koodia functions.php-tiedostoon (kuva 13). Tämän voit suoraan kopioida Options Frameworkin functions.php-tiedostosta.

```
/*
 * Loads the Options Panel
 *
 * If you're loading from a child theme use stylesheet_directory
 * instead of template_directory
 */

define( 'OPTIONS_FRAMEWORK_DIRECTORY', get_template_directory_uri() . '/inc/' );
require_once dirname( __FILE__ ) . '/inc/options-framework.php';
require_once get_template_directory() . '/options.php';
```

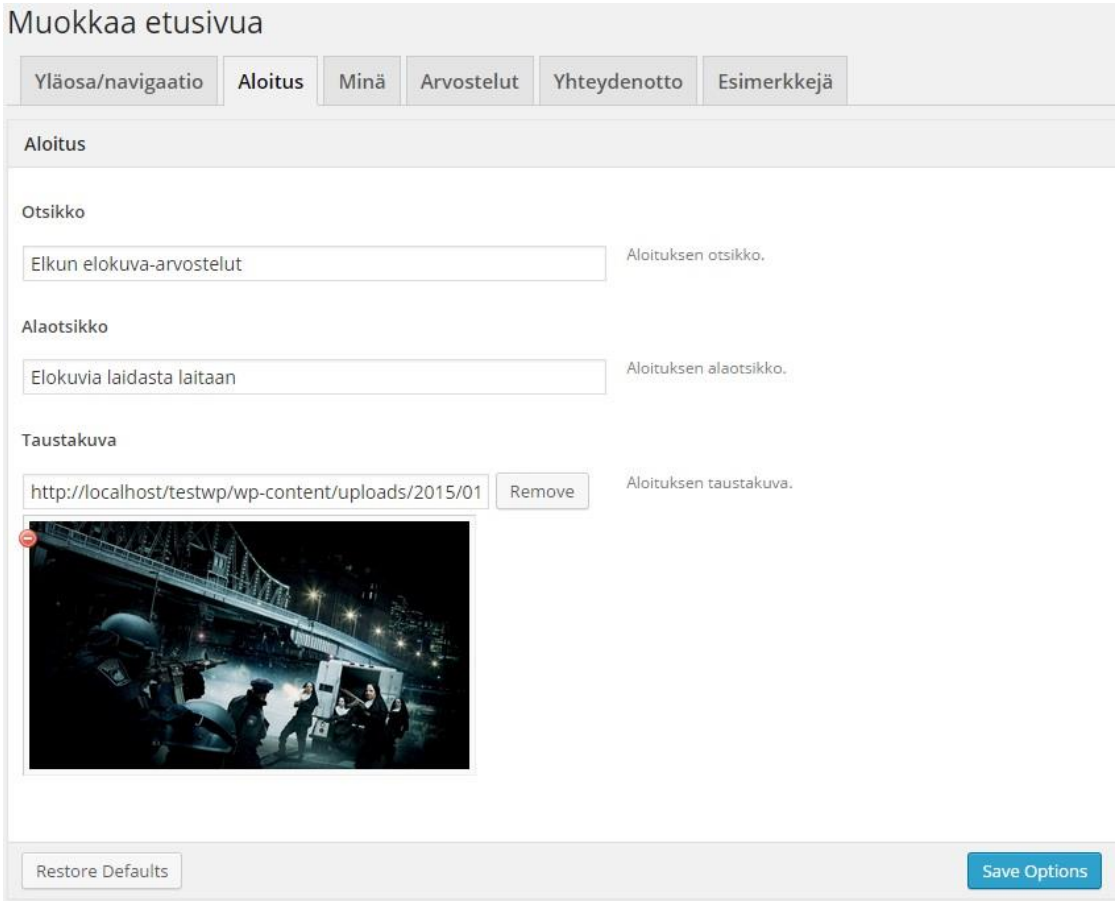
KUVA 13. Otetaan Options Framework käyttöön my-first-themen functions.php:ssä

Suomen kielistä WordPressiä käytettäessä halutaan myös muuttaa WordPressin hallintapaneeliin ilmestyvän Theme Options -valikon nimi. Se tapahtuu class-options-framework-admin.php-tiedostossa, joka sijaitsee inc-kansiossa olevan includes-kansion alla (kuva 14).

```
static function menu_settings() {

    $menu = array(

        // Modes: submenu, menu
        'mode' => 'submenu',

        // Submenu default settings
        'page_title' => __( 'Muokkaa etusivua', 'theme-textdomain' ),
        'menu_title' => __( 'Muokkaa etusivua', 'theme-textdomain' ),
        'capability' => 'edit_theme_options',
        'menu_slug' => 'options-framework',
        'parent_slug' => 'themes.php',

        // Menu default settings
        'icon_url' => 'dashicons-admin-generic',
        'position' => '61'

    );

    return apply_filters( 'optionsframework_menu', $menu );
}
```

KUVA 14. Muutetaan valikon nimi. Keskitymme muokkaamaan vain etusivua

Options Framework on nyt liitetty teemaamme. Seuraavaksi muokataan Options Frameworkin ominaisuudet teemaamme sopiviksi. Kuten liitteestä 2 nähdään, etusivumme rakenne on jaettu osiin (section). On siis järkevää jakaa Options Frameworkin asetuksetkin välilehtiin näiden osien mukaisesti (kuva 15).



KUVA 15. Sivun asetukset kannattaa jakaa välilehtiin luettavuuden parantamiseksi

Halutut Options Frameworkin ominaisuudet määritellään options.php-tiedostossa. Se sisältää valmiiksi esimerkit kaikista mahdollisista ominaisuuksista. Ominaisuudet ovat taulukoita ja ne vaativat vaihtelevan määrän attribuutteja, kuten esimerkiksi nimen, tyypin ja uniikin id:n (kuva 16).

```
//Intro banner
$options[] = array(
    'name' => __( 'Aloitus', 'theme-textdomain' ),
    'type' => 'heading'
);

$options[] = array(
    'name' => __( 'Otsikko', 'theme-textdomain' ), //<-- option name
    'desc' => __( 'Aloituksen otsikko.', 'theme-textdomain' ), //<-- description
    'id' => 'intro_head', //<-- unique id
    'std' => '', //<-- standard value
    'type' => 'text' //<-- option type
);

$options[] = array(
    'name' => __( 'Alaotsikko', 'theme-textdomain' ),
    'desc' => __( 'Aloituksen alaotsikko.', 'theme-textdomain' ),
    'id' => 'intro_subhead',
    'std' => '',
    'type' => 'text'
);

$options[] = array(
    'name' => __( 'Taustakuva', 'theme-textdomain' ),
    'desc' => __( 'Aloituksen taustakuva.', 'theme-textdomain' ),
    'id' => 'intro_bg_image',
    'type' => 'upload'
);
```

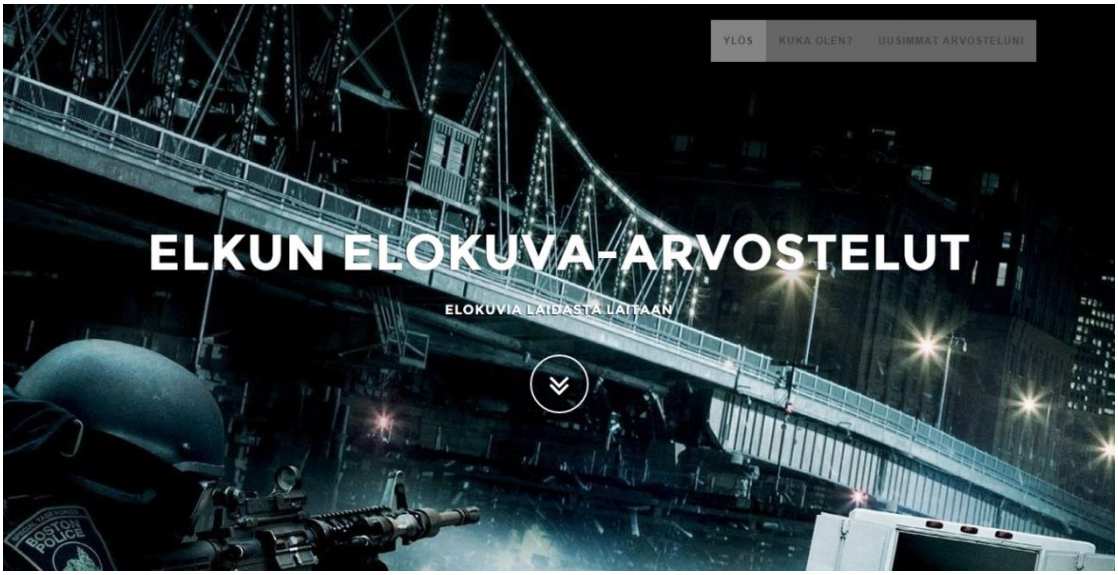
KUVA 16. Välilehti sivustomme aloitus-kohdan muokkaukselle

Options Framework tallentaa annetut sisällöt sekä asetukset tietokantaan ja sivustolle saadaan nyt lisättyä options.php-tiedoston mukaista tietoa hallintapaneelin kautta. Asetukset eivät kuitenkaan vielä näy sivuillamme, vaan joudumme seuraavaksi hakemaan ne sivuillemme tietokannasta. Tässä käytetään asettamiamme id-attribuutteja ja tämän takia uniikki id on tärkeä. Kuvassa 17 haetaan sivustomme aloitus-kohdan tiedot tietokannasta of_get_option()-funktion avulla.

```
<!-- Section: intro -->
    <section id="intro" class="intro"
    style="background: url(<?php echo of_get_option('intro_bg_image', $default); ?>)
            no-repeat
            top
            center;">

        <div class="slogan">
            <h2><?php echo of_get_option('intro_head', $default); ?></h2>
            <h4><?php echo of_get_option('intro_subhead', $default); ?></h4>
        </div>
        <div class="page-scroll">
            <a href="#service" class="btn btn-circle">
                <i class="fa fa-angle-double-down animated"></i>
            </a>
        </div>
    </section>
    <!-- /Section: intro -->
```

KUVA 17. of_get_option()-funktio ottaa sisäänsä kaksi parametriä; options.php:ssä määritetyn id:n sekä vaihtoehtoisen sisällön, mikäli id:llä ei ole sisältöä

Kun tiedonhakufunktiot ovat lisätty etusivumme koodiin, näkyy Option Frameworkissa määrittelemämme sisällöt nyt sivuston etusivulla (kuva 18). Sisältöjä voidaan nyt vaihtaa hallintapaneelista, eikä html- tai php-koodiin tarvitse enää koskea. Tällä tavoin voidaan muuttaa sivustomme käsittelemään vaikkapa kissavideoita.



KUVA 18. Määrittelemämme ominaisuudet näkyvät etusivulla

Options Frameworkilla saadaan muutettua asetuksia ja muokattua sisältöä, mutta sillä ei voida lennosta luoda uutta sisältöä, joka on WordPressin perusidea. Loin aikaisemmin functions.php-tiedostoon uuden julkaisutyypin elokuva-arvosteluille (kuva 12). Eri julkaisuille ei tarvitse välttämättä luoda omia tyyppejä, mutta se helpottaa julkaisujen erottelemista, kun niitä halutaan hakea kannasta. Lisäksi julkaisun asetuksia saadaan muokattua tarpeiden mukaisesti.

Julkaisu hakee sivupohjakseen single-{post_type}.php-tiedoston. Jos tätä ei löydy, käyttää julkaisu WordPressin sivupohjahierarkian mukaisesti single.php-tiedostoa sivupohjanaan (WordPress Codex 2014c). Luomme siis single-movie_review.php-tiedoston julkaisutyyppimme nimen mukaan. Ideana on tuoda julkaisut selaimeen ponnahdusikkunoina Magnific Popup -pluginia ja AJAXia käyttäen, siksi emme tarvitse headeria tai footeria (kuva 19).

```php
<?php
/**
 * The template for displaying all single movie review posts.
 *
 * @package my-first-theme
 */

//get_header(); ?>

    <div id="ajax-content" class="ajax-content-area">


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

            <?php get_template_part( 'content', 'single' ); ?>

            <?php //my_first_theme_post_nav(); ?>

            <?php
                // If comments are open or we have at least one comment,
                // load up the comment template
                if ( comments_open() || get_comments_number() ) :
                    comments_template();
                endif;
            ?>

        <?php endwhile; // end of the loop. ?>


    </div><!-- #ajax-content -->

<?php //get_sidebar(); ?>
<?php //get_footer(); ?>
```

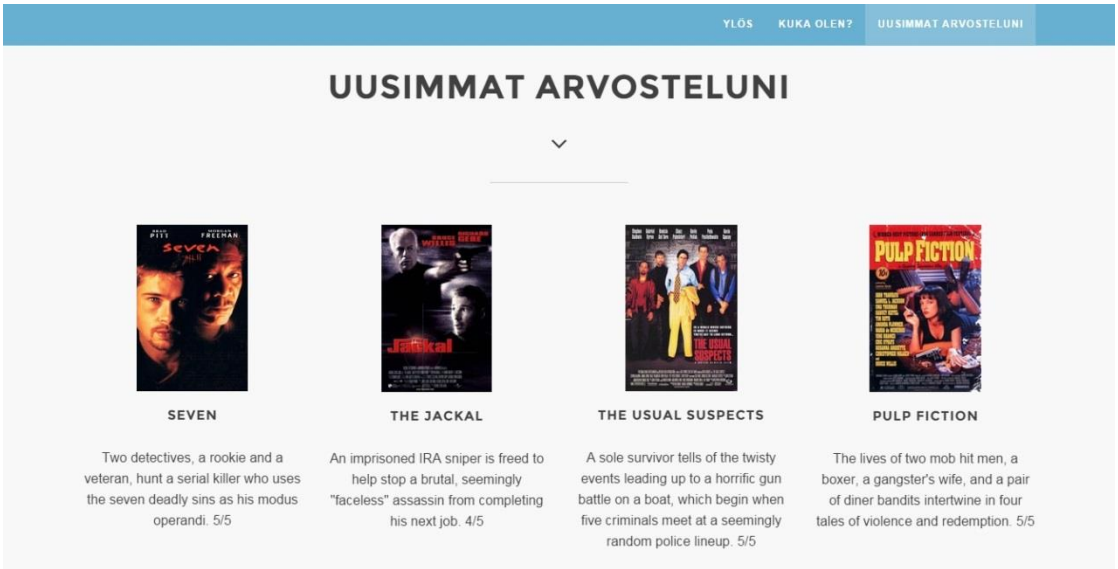
KUVA 19. single-movie_review.php-tiedoston rakenne

Seuraavaksi lisätään teemaamme silmukka hakemaan julkaisuistamme sisältöä etusivulle, kuten otsikko, esittelykuva ja ote tekstistä (kuva 20). Silmukka järjestää julkaisut automaattisesti uusimmasta vanhimpaan. Lisäksi voidaan määritellä kuinka monta julkaisua silmukka hakee.

```
<div class="row">

    <?php $loop = new WP_Query( array( 'post_type' => 'movie_review',
                                       'posts_per_page' => 4 )); ?>
    <?php while ( $loop->have_posts() ) : $loop->the_post(); ?>

      <div class="col-sm-3 col-md-3">
        <div class="wow fadeInLeft" data-wow-delay="0.2s">
        <div class="service-box">
            <div class="service-icon popup"
                 href="<?php echo get_permalink(); ?>">
                <?php echo get_the_post_thumbnail($post_id,
                                                  array(200,200)); ?>
                                                  <!-- Image size -->
            </div>
            <div class="service-desc">
                <h5 href="<?php echo get_permalink(); ?>">
                    <?php echo get_the_title(); ?></h5>
                <p><?php echo get_the_excerpt(); ?> </p>
            </div>
        </div>
        </div>
    </div>
    <?php endwhile; wp_reset_query(); ?>
```

KUVA 20. Haetaan neljä uusinta elokuva-arvostelua kannasta

Sivullamme näkyy nyt neljä uusinta movie_reviw-julkaisua (kuva 21). Tämä on helppo tapa tuoda uusimmat julkaisut esille. Kaikki julkaisut voidaan hakea vaikkapa erilliselle arkistosivulle.



KUVA 21. Neljä uusinta julkaisua näkyvät etusivulla

Options Frameworkilla voidaan toki myös vaikuttaa WordPressin omiin funktioihin. Etusivumme muuttuu näppärästi elokuva-arvosteluista levy- tai kirja-arvosteluihin.

Tai miksei myös niihin kissavideoihin. Luodaan vain uusi julkaisutyyppi, esim. album_review, ja lisäätään Options Frameworkilla etusivun asetuksiin mahdollisuus vaihtaa silmukassa haettua julkaisutyyppiä (kuva 22).

```
$output = 'names'; // names or objects, note names is the default
$operator = 'and'; // 'and' or 'or'

$post_types = get_post_types( $args, $output, $operator );

$options[] = array(
    'name' => __( 'Valitse julkaisutyyppi', 'theme-textdomain' ),
    'desc' => __( 'Mitkä julkaisut näytetään etusivulla', 'theme-textdomain' ),
    'id' => 'select_post_type',
    'std' => 'two',
    'type' => 'select',
    'options' => $post_types
);
```

KUVA 22. Lisätään options.php:n asetuksiin mahdollisuus vaihtaa julkaisutyyppiä

Options Frameworkilla saadaan siis muokattua sivua todella monipuolisesti. Vaikka se nimensä mukaisesti onkin tarkoitettu lähinnä teeman asetusten muokkaamiseen, en näe syytä miksi sitä ei voisi käyttää myös sisällönhallinnassa. Ominaisuudethan tähän Option Frameworkista löytyvät.

2.4 Vimpaimet

Vimpaimet, eli widgetit, ovat pieniä sivuston osia, joilla voidaan tuoda toiminnallisuuksia ja sisältöä sivustolle. Vimpaimet tarvitsevat toimiakseen teeman joka tukee niitä. Teemassa on siis tällöin yksi tai useampi alue, widget area, jossa vimpaimen sisältö tai toiminnallisuus voidaan esittää. Vimpainten alkuperäinen tarkoitus oli luoda helppokäyttöisyyttä sivun ja teemaan hallintaan, esimerkkinä tästä sivuille vimpaimena lisättävä hakupalkki, joka on siis WordPressin oma vimpain. WordPressin omiin vimpaimiin kuuluvat hakupalkin lisäksi mm. kategoriat sekä tagit. (Wpbeginner 2015.)

Vimpaimet ovat yksi tapa esittää lisäosien toiminnallisuuksia. Vimpain on siis rakennettu lisäosassa ja lisäosan käyttäytymistä voidaan muokata vimpaimen kautta. Esimerkiksi lisäosalla, joka antaa mahdollisuuden luoda sivustosta monikielisen, on vimpain jolla käyttäjä voi vaihtaa sivuston kieltä.

Vimpainalueesta puhutaan yleensä sivupalkkina, eli sidebarina. Alue voi kuitenkin sijaita missä vain sivustolla, tämä täytyy vain määritellä teemaan (Wpbeginner 2015). Seuraavaksi luodaan my-first-theme-teemaamme uusi alue vimpaimille. Itse vimpaimen luontia käsitellään Lisäosat-luvussa, jossa lisäosamme esiintyy vimpaimena.

Vimpainalueen luonti

Vimpainalue luodaan teeman functions.php-tiedostossa. Se on funktio, joka tarttuu widgets_init-koukkuun. Sille annetaan nimi, id sekä html tagit johon yksittäinen vimpain ”kiedotaan” sivustolla esitettäessä (kuva 23). Näin päästään vimpaimeen käsiksi css-tyyleillä tai javascriptillä.

```
function my_first_theme_widgets_init() {
    register_sidebar( array(
        'name'          => __( 'Oma vimpainalueeni', 'my-first-theme' ),
        'id'            => 'my-widget-area',
        'description'   => '',
        'before_widget' => '<div id="%1$s" class="widget my-widget-area">',
        'after_widget'  => '</div>',
        'before_title'  => '<h1 class="widget-title">',
        'after_title'   => '</h1>',
    ) );
}
add_action( 'widgets_init', 'my_first_theme_widgets_init' );
```

KUVA 23. Luodaan teemallemme uusi vimpainalue functions.php-tiedostossa

Uusi vimpainalueemme näkyy nyt WordPressin hallintapaneelissa Ulkoasu-valikon vimpaimet-välilehden alla. Alueelle voidaan nyt vetää ja pudottaa haluttuja vimpaimia, mutta sivustolla ne eivät vielä näy, koska emme ole kutsuneet sitä. Kuten aikaisemmin sanoin, on vimpainalueen ”sidebar”-nimitys hieman hämäävä, koska alue voidaan liittää sivustolle mihin tahansa haluttuun kohtaan (kuva 24).

```
<section id="intro" class="intro"
style="background:
       url(<?php echo of_get_option('intro_bg_image', $default); ?>)
       no-repeat
       top
       center;">

    <div class="slogan">
        <h2><?php echo of_get_option('intro_head', $default); ?></h2>
        <h4><?php echo of_get_option('intro_subhead', $default); ?></h4>
    </div>
    <div class="page-scroll">
        <a href="#about" class="btn btn-circle">
            <i class="fa fa-angle-double-down animated"></i>
        </a>
    </div>
    <?php if ( !function_exists('dynamic_sidebar') ||
                !dynamic_sidebar('my-widget-area') ) : endif; ?>
</section>
```

KUVA 24. Näytetään oma vimpainalueemme sivustomme index.php:n intro-osassa

Vimpain näkyy nyt etusivullamme määrittelemässämme kohdassa. Sen sijaintia ja ulkoasua voidaan muuttaa css-tyyleillä, joita varten määrittelimme before_widget ja after_widget-parametrit, tässä tapauksessa siis html:n div-elementin aukaisun, tunnisteen ja sulkeutumisen.

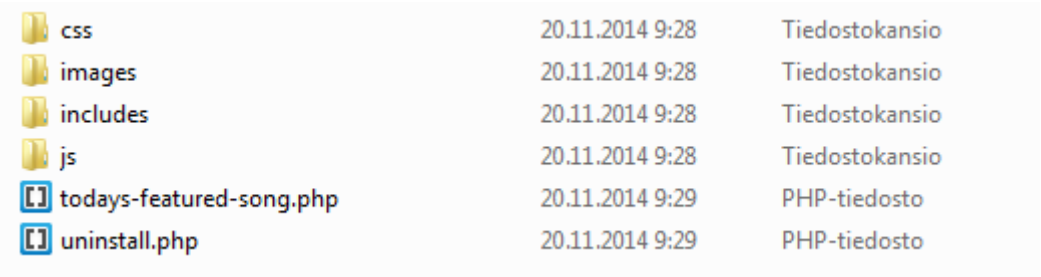
2.5 Lisäosat

Lisäosat, eli pluginit, voivat olla pieniä tai suuria php:lla rakennettuja ohjelmia, jotka liittävät itsensä WordPressin ytimeen. Niillä saavutetaan toimintoja, joita WordPress ei valmiina tarjoa. Nämä toiminnot voivat olla todella yksinkertaisia tai monimutkaisia; mitään rajoituksia ei ole. (Williams ym. 2011, 1–3.) WordPress tarjoaakin laajan kirjaston, josta löytyy käyttäjien tekemiä lisäosia tällä hetkellä yli 30 000 kappaletta (WordPress.org 2015). Rakennan tässä kappaleessa oman lisäosan nimeltään ”Todays Featured Song”. Lisäosa näkyy teemassamme widgettinä, eli vimpaimena.

2.5.1 Rakenne

WordPress-lisäosa voi yksinkertaisimmillaan olla vain yksi php-tiedosto, joka sisältää kaiken lisäosan toiminnallisuuden. Laajempi lisäosa voi koostua useammasta tiedostosta ja kansioista, jotka sisältävät esimerkiksi javascript- ja tyylitiedostoja.

(Williams ym. 2011, 11–13.) Kuvassa 25 on esimerkki laajemman lisäosan kansiorakenteesta.



KUVA 25. Laajemman lisäosan kansiorakenne. Sanat erotellaan WordPressin standardin mukaan tavuviivoilla alaviivojen sijaan

Lisäosat sijaitsevat WordPressin plugins-alihakemistosssa. Lisäosa voidaan ladata sinne WordPressin hallintapaneelilla tai FTP-ohjelmalla. Localhostissa toimittaessa lisäosan voi luonnollisesti siirtää WordPressin alle. Ainoa toiminnallisuus, jonka WordPress vaatii lisäosalta, on niin sanottu ”header”, joka on ensimmäinen osa lisäosan päätiedostoa (kuva 26).

```
1  <?php
2  /*
3  Plugin Name: Todays featured song
4  Plugin URI: http://poraa.16mb.com/wordpress_plugins/todays_featured_song
5  Description: Shows song of your choice and a youtube link in a widget area on WordPress frontpage
6  Version: 1.0
7  Author: Aapeli Poranen
8  Author URI: http://poraa.16mb.com
9  License: GPLv2
10 */
11 ?>
```

KUVA 26. todays-featured-song.php-tiedoston header-osa

Header-osassa kerrotaan lisäosan nimi, kotisivu, kuvaus, versionumero, tekijä, tekijän kotisivu sekä lisenssi. Ainoastaan lisäosan nimi on pakollinen ohjelman tunnistumiseen WordPressissä.

2.5.2 Toiminnallisuus

Kun tiedosto tai kansio on siirretty WordPressin plugins-kansioon, lisäosan voi nyt ottaa käyttöön ja poistaa käytöstä WordPressin hallintapaneelissa Lisäosat-välilehden alta. Lisäosahan ei tällä hetkellä vielä tee mitään, mutta WordPressia se ei kiinnosta. Seuraavaksi rakennetaan lisäosan toiminnallisuus ja näytetään se vimpaimena.

Lisäosan toiminnallisuuden aktivoimiseen käytetään WordPressin hookkeja eli koukkuja joihin kytkeydytään, kun kyseinen koukku ajetaan WordPressin toimesta. Koukuista kerroin tarkemmin tämän opinnäytetyön kohdassa 3.2. Tässä vimpain-lisäosassa tartutaan widgets_init-koukkuun, joka laukaistaan kun WordPress on rekisteröinyt kaikki omat vimpaimensa. Tämän jälkeen rekisteröidään oma vimpaimemme. Tässä tapauksessa ”widget_todays_featured_song” (Kuva 27).

```
//grab widgets_init action hook to execute your function
add_action('widgets_init', 'register_my_widgets');

//register widget
function register_my_widgets() {
    register_widget('widget_todays_featured_song');
}
```

KUVA 27. Tartutaan widgets_init-koukkuun ja rekisteröidään vimpain

Seuraavaksi luodaan luokka vimpaimellemme. Luokka periytetään WordPressin WP_Widget-luokasta ja luokan nimenä käytetään äsken rekisteröityä vimpaimen nimeä. Tässä tapauksessa siis ”widget_todays_featured_song”. Tämän jälkeen luodaan vimpaimelle asetukset jonka jälkeen luodaan itse vimpain. (kuva 28.)

```
//create class for your widget
class widget_todays_featured_song extends WP_Widget {
    /*create options (classname for html elements and a little
    description which shows on admin panel) then process your
    widget*/
    function widget_todays_featured_song() {
            $my_widget_options = array(
                'classname' => 'widget_todays_featured_song_class',
                'description' => 'Show\'s your featured song of the
                day, and a youtube link for it.'
            );
        $this->WP_Widget('widget_todays_featured_song', 'My
        Featured Song Of The Day', $my_widget_options);
    }//function
```

KUVA 28. Luodaan vimpaimelle luokka ja asetukset jonka jälkeen luodaan itse vimpain

Vimpain on nyt rekisteröity WordPressiin. Kun vimpain aktivoidaan, se näkyy WordPress-hallintapaneelin vimpaimet-välilehden alla. Siinä ei kuitenkaan ole vielä mitään näkyvää sisältöä. Seuraavaksi luodaan lomake, jolla voidaan syöttää ja

tallentaa tietoa vimpaimellemme (kuva 29). Huomaa, että html:n form-tageja ei tarvita, sillä WP_Widget-luokka hoitaa tämän puolestamme.

```
//widget form (settings/information)
    function form($instance) {
        $defaults = array(
            'song' => '',
            'link' => ''
        );
        $instance = wp_parse_args((array) $instance, $defaults);
        $song = $instance['song'];
        $link = $instance['link'];
        ?>
        <p>Song: <input class="widefat" type="text"
                        name="<?php echo $this->get_field_name('song'); ?>"
                        value="<?php echo esc_attr($song); ?>"></p>
        <p>Link: <input class="widefat" type="text"
                        name="<?php echo $this->get_field_name('link'); ?>"
                        value="<?php echo esc_attr($link); ?>"></p>
<?php
    }//function

//save form
    function update($new_instance, $old_instance) {
        $instance = $old_instance;
        $instance['song'] = strip_tags($new_instance['song']);
        $instance['link'] = strip_tags($new_instance['link']);

        return $instance;
    }//function
```

KUVA 29. Luodaan lomake, jossa määritellään vimpaimen näyttämä tieto ja tallennetaan se

Nyt vimpaimessamme on siis lomake, jossa on kaksi kenttää tiedolle; päivän musiikkikappale ja linkki tähän kappaleeseen. Tietoa voidaan päivittää WordPressin hallintapaneelista vimpaimet-välilehden alta. Tieto ei kuitenkaan vielä näy missään muualla kuin hallintapaneelissa. Seuraavaksi rakennetaan koodi tiedon tulostukseen (kuva 30).

```
//display your widget
    function widget($args, $instance) {
        extract($args);
        echo $before_widget;
        $song = empty($instance['song']) ? '$nbsp' : $instance['song'];
        $link = empty($instance['link']) ? '$nbsp' : $instance['link'];

        echo '<h5>Päivän biisivalintani.</h5>';
        echo '<p><a target="_blank" href="'.$link.'">'.$song.'</a></p>';
        echo $after_widget;
    }//function
}//class
```

KUVA 30. Tulostetaan vimpaimen tieto

Huomaa, että kuvassa 30 ei määritellä mihin tieto tulostuu, vaan se määräytyy sen alueen (widget area) mukaan, johon olen liittänyt vimpaimesi. Tästä lisää kohdassa 2.3.4.

3 WORDPRESS MULTISITE

WordPressin versiosta 3.0 lähtien käyttäjillä on ollut mahdollisuus luoda sivujen verkko käyttäen WordPressin multisite-ominaisuutta. Tällä ominaisuudella voidaan siis luoda yhden WordPress-asennuksen alle useita sivustoja. Sivut voivat olla itsenäisiä omine tyyleineen ja ominaisuuksineen, tai ne voivat käyttää esim. yhteisiä teemoja ja lisäosia. Sivustot ovat virtuaalisia, mikä tarkoittaa sitä, että eri sivustoilla ei ole serverillä omia erillisiä kansioitaan, vaan sivustoille on WordPress-asennuksen tietokannassa omat taulunsa. (WordPress Codex 2015a.)

Multisite voidaan ottaa käyttöön jo olemassa olevassa WordPress-asennuksessa, jolloin tiedostoista ja tietokannasta on suositeltavaa ottaa varmuuskopiot. Tässä ohjeessa käsittelen kuitenkin multisite-ominaisuuden käyttöönottoa ns. puhtaalta pöydältä. Teen siis uuden WordPress-asennuksen ja lisään siihen multisite-ominaisuuden. WordPress on suositeltavaa asentaa tässä tapauksessa palvelimen juureen, jotta multisite-ominaisuuden vaatima osoiterakenne säilyy ehjänä ilman erillistä asetusten muokkaamista.

3.1 Multisite-ominaisuuden käyttöönotto

Suosittelen multisite-ominaisuutta tukevan WordPressin asennuksessa ohjatun asennuksen sijaan määrittämään tarvittavat asetukset wp-config.php-tiedostossa, koska wp-config.php:tä tarvitaan tässä tapauksessa myös myöhemmin. Ohjattua asennusta käyttäessäni huomasin, että WordPress ei jostain syystä luonut asetuksia sisältävää wp-config.php-tiedostoa, vaan ilmeisesti lisäsi asetukset suoraan kantaan. Tämä saattoi toki olla joku bugi, tai palvelinpään ongelma.

Kun WordPress on siirretty palvelimelle, määritellään wp-config-sample.php-tiedostosta tietokantayhteyden asetukset oikeiksi. Lisäksi määritellään liuta uniikkeja tunnistautumisavaimia. Ohje tämän tekemiseen on kommentoitu tiedostoon.

Lisäyksenä normaaleihin asetuksiin lisätään wp-config-sample.php-tiedostoon kuvassa 31 näkyvä rivi. Tämän jälkeen tallennetaan tiedosto wp-config.php:nä ja nyt WordPressin asennus voidaan suorittaa loppuun web-käyttöliittymällä. (Huom! Multisite-ominaisuus tarvitsee palvelimelta tuen osoiterakenteen muutoksille esim. Apache:lla mod_rewrite -moduulin.)

```
/* Multisite */
define( 'WP_ALLOW_MULTISITE', true );

/* Siinä kaikki, älä jatka pidemmälle! */

/** WordPress absolute path to the Wordpress directory. */
```

KUVA 31. Otetaan multisite -ominaisuus käyttöön wp-config -tiedoston loppupuolella

Kun multisite-ominaisuus on onnistuneesti otettu käyttöön, ilmestyy Wordpressin hallintapaneeliin työkalut-välilehden alle uusi työkalu verkon asentamiselle. Tässä voidaan valita sivustojen osoiterakenne joko aliverkkotunnuksiksi tai alihakemistomaiseksi rakenteeksi. Aliverkkotunnusrakenne vaatii palvelinpuolelta joitakin ominaisuuksia, kuten ”jokerimerkin” (*) käytön aliverkkotunnuksissa. Tätä siis tarvitaan silloin, kun halutaan luoda sivustoja ennalta määräämättömästi ns. on-demand -tyylillä. (WordPress Codex 2015b). Käytän itse tässä tapauksessa alihakemistorakennetta oman testipalvelimeni rajoitusten vuoksi.

Kun osoiterakenne on valittu, antaa Wordpress muutaman rivin koodia kopioitavaksi wp-config.php sekä htaccess-tiedostoihin. Seuraavaksi lisätään wp-config.php-tiedostoon kuva 32:n mukaiset rivit.

```
/* Multisite */
define( 'WP_ALLOW_MULTISITE', true );
define('MULTISITE', true);
define('SUBDOMAIN_INSTALL', false);
define('DOMAIN_CURRENT_SITE', 'poraa.16mb.com');
define('PATH_CURRENT_SITE', '/');
define('SITE_ID_CURRENT_SITE', 1);
define('BLOG_ID_CURRENT_SITE', 1);

/* Siinä kaikki, älä jatka pidemmälle! */
```

KUVA 32. Palataan takaisin wp-config.php -tiedostoon lisäämään määrittelemämme asetukset

Lisäksi osoiterakenteen muutosta varten palvelinta pitää ohjeistaa muuttamaan osoitteet halutunlaisiksi. Jotta multisite-ominaisuus toimisi, tulee palvelimella olla tähän mahdollisuus. Ainakin Apache-palvelinta käytettäessä muutossäännöt ovat ihan hirvittävää sekamelskaa, mutta onneksi WordPress generoi tarvittavan koodin valmiiksi joka vain liitetään htaccess-tiedostoon (kuva 33).

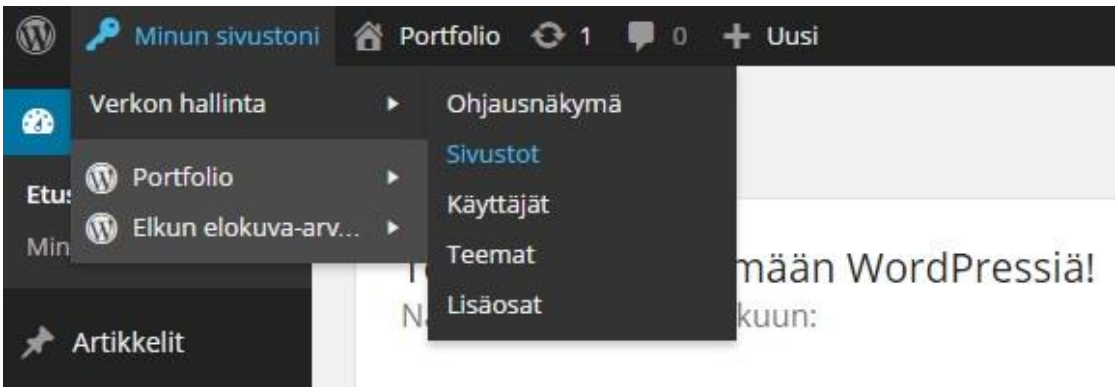
```
# BEGIN WordPress
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]

# add a trailing slash to /wp-admin
RewriteRule ^([_0-9a-zA-Z-]+/)?wp-admin$ $1wp-admin/ [R=301,L]

RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^ - [L]
RewriteRule ^([_0-9a-zA-Z-]+/)?(wp-(content|admin|includes).*) $2 [L]
RewriteRule ^([_0-9a-zA-Z-]+/)?(.*\.php)$ $2 [L]
RewriteRule . index.php [L]
# END WordPress
```

KUVA 33. htaccess-tiedoston tulisi nyt näyttää tältä

Multisite-ominaisuus pitäisi olla nyt käytössä. Wordpress tarjoaa linkin uudelleenkirjautumiselle, jonka jälkeen hallintapaneelin yläpalkkiin ilmestyy uusi Minun sivustoni -valikko.

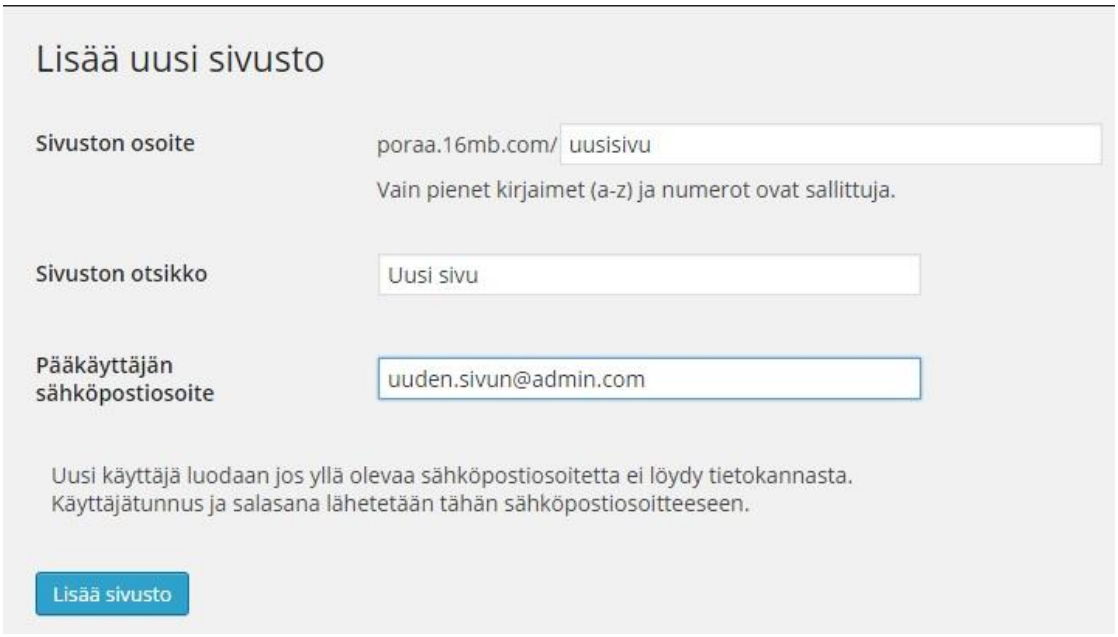
3.2 Sivustot

Wordpress-asennuksen pääkäyttäjä, eli verkon järjestelmänvalvoja voi nyt luoda sivustoja Minun sivustoni -valikon alta ”sivustot”-kohdasta. Järjestelmänvalvoja hallinnoi myös verkon teemoja, lisäosia sekä käyttäjiä (kuva 34). Teemat ja lisäosat ovat tässä tapauksessa verkon yhteisiä, ja alikäyttäjien täytyy tyytyä niihin teemoihin ja lisäosiin, joita järjestelmänvalvoja on lisännyt verkostoon.



KUVA 34. Multisite-verkon valikko Wordpressin hallintapaneelissa

Uudelle sivustolle annetaan osoite, otsikko sekä uuden sivun pääkäyttäjän sähköpostiosoite, johon kyseisen sivuston hallintapaneelin salasana lähetetään (kuva 35). Salasana generoidaan automaattisesti WordPressin toimesta ja käyttäjätunnus on luodun sivun polku. Esim. jos sivun osoite on http://poraa.16mb.com/uusisivu/, on käyttäjätunnus tällöin ”uusisivu”.



KUVA 35. Uuden sivuston luonti multisite-verkon hallintapaneelissa

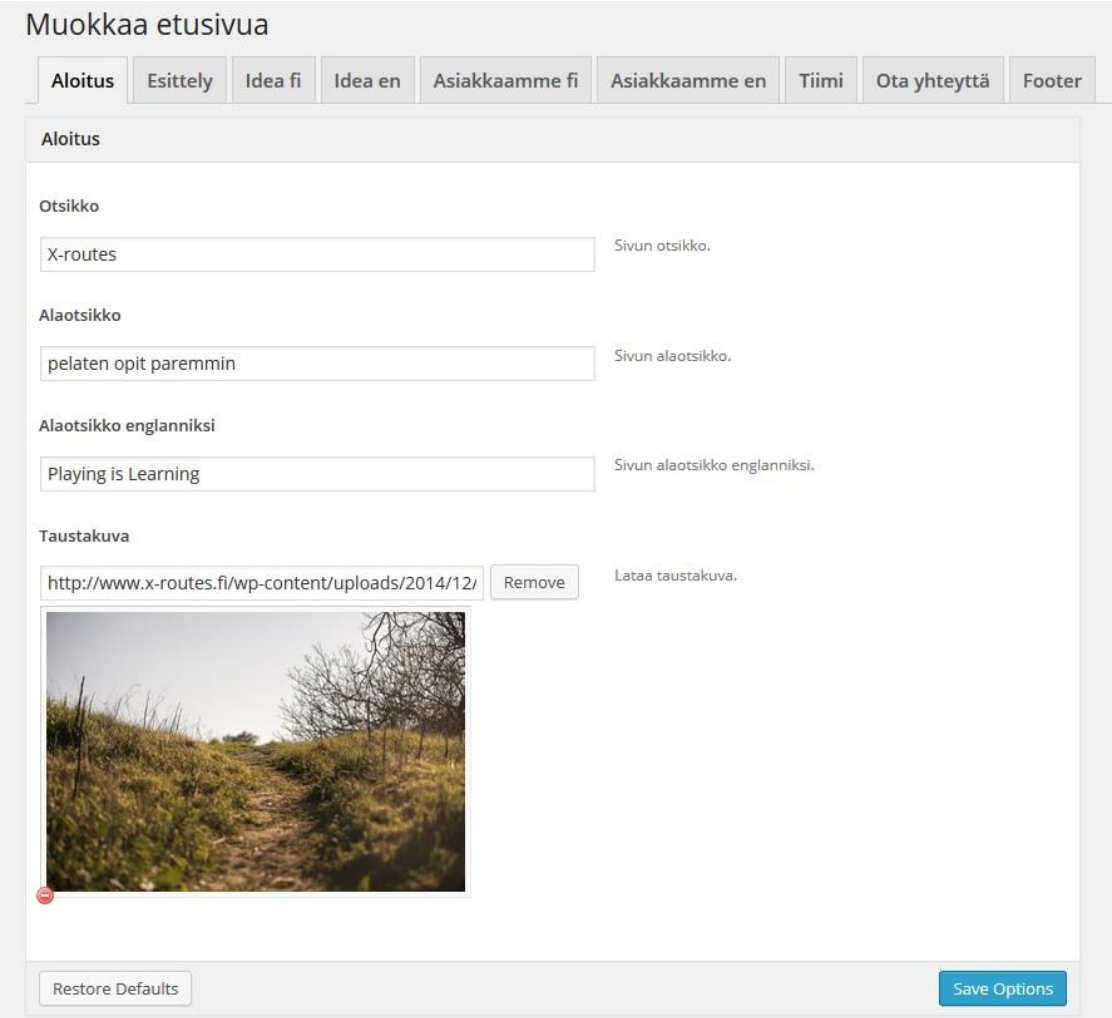
Uuden sivuston käyttäjä voi muuttaa salasanansa jälkikäteen hallintapaneelista, mutta käyttäjänimeä ei kuitenkaan voida vaihtaa. Verkon järjestelmänvalvoja voi halutessaan antaa käyttäjälle järjestelmänvalvojan oikeudet, jolloin uuden sivuston käyttäjä voi itse esimerkiksi muuttaa sivuston polkua.

Kuten aikaisemmin sanoin, myös teemojen, lisäosien sekä käyttäjien hallinnointi vaatii verkon järjestelmänvalvojan oikeudet. Järjestelmänvalvoja voi lisätä multisite-verkkoon teemoja sekä lisäosia, joita alikäyttäjät voivat halutessaan ottaa käyttöön. Oman teeman tai lisäosan lisääminen sivustolle ei siis onnistu ilman järjestelmänvalvojan oikeuksia. Sivuston sisällönhallintaan käyttäjällä on kuitenkin täydet oikeudet. (WordPress Codex 2015c.)

4 X-ROUTES

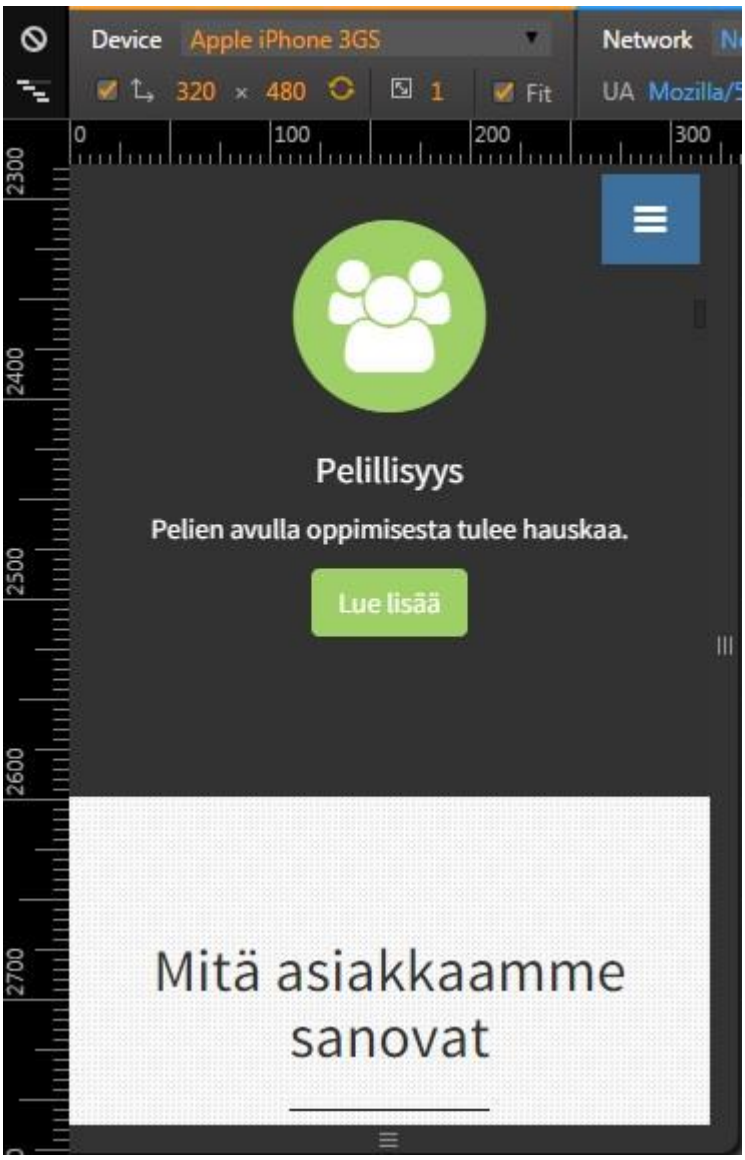
Tässä opinnäytetyössä esiteltyjä tekniikoita sovelsin myös Observis Oy:lle tekemässäni verkkosivussa. Sivu on landing page -tyylinen verkkosivu, jossa kerrottaan mm. tuotteen ideasta. X-routes on opetuksellisuuteen ja liikuntaan tähtäävä mobiilipeli kouluille, jonka tarkoituksena on tuoda ulkona liikkuminen osaksi opetusta. X-routesilla opettajat voivat luoda reittejä, joita oppilaat seuraavat omilla älypuhelimillaan. Reiteille on sijoitettu kiintopisteitä, joiden kohdalla oppilaiden on tarkoitus vastata kyseiselle kiintopisteelle liitettyyn kysymykseen. Sivusto löytyy osoitteesta http://www.x-routes.fi.

Sisällönhallintaan sivusto käyttää pelkästään Options Frameworkia (kuva 36), mutta esimerkiksi idean esittelyyn käytetyn ”karuselli”-toiminnon sisällön olisi voinut toteuttaa WordPressin julkaisuina (kuva 12), jotka olisivat haettu karusellin osiksi etusivulle. Ajattelin kuitenkin pitää etusivun sisällön Options Frameworkin hallinnassa sivuston mahdollista laajenemista varten, jolloin esimerkiksi uutisia sekä muita artikkeleita hallittaisiin WordPressin omilla toiminnoilla.



KUVA 36. X-routesin etusivun sisällönhallinta tapahtuu Options Frameworkilla

Ulkoasuna sivustolla käytettiin valmista responsiivisen (kuva 37) Bootstrap frameworkin päälle rakennettua html-teemaa, jota muokkasin hieman ja liitin sen Undescrores-palvelulla luotuun WordPress-teemaani. Kuvien lisäksi ulkoasun grafiikassa on käytetty Font Awesomen ikoneita. Font Awesomen ikonit ovat css-tyylitiedostossa määriteltyjä fonttien tapaan skaalautuvia vektorigrafiikoita, joita voidaan käyttää esim. painikkeina tai tekstin tukena.



KUVA 37. X-routes.fi iPhone 3:n näytöllä

Toiminnallisuuksia sivulla on aikaisemmin mainitun karusellin lisäksi javascriptillä toteutettu liukuva siirtyminen sivun sisäisillä linkeillä, AJAX-tekniikalla toimiva lomakkeen lähetys sekä sosiaalisen median syöte. Palautelomakkessa käytettiin PHPMailer-luokkaa ja sosiaalisen median syötteestä vastaa Codecanyonin jQuery Social Stream -lisäosa.

Sivun toteutus onnistui yleisesti ottaen hyvin. Kaikki tavoitteet saatiin toteutettua ja toimeksiantajat olivat tyytyväisiä. Suurin ongelmakohta oli sivuston responsiivisuus. Ennen kuin löysin Bootstrap frameworkin oli suunnittelemiani ulkoasuratkaisuja hankala saada toimimaan laitteella kuin laitteella. Myös sivuston värimaailma osoittautui ongelmalliseksi. Sivustossa päädyttiin lopulta käyttämään Observis Oy:n

värimaailmaa, joka ei tahdo sopia tuotteen ilmeeseen. Uuden värimaailman luonti onkin työn alla ja se tulee tulevaisuudessa muuttumaan.

5 YHTEENVETO

Oman WordPress-sivuston luominen siis onnistuu melko kivuttomasti, kunhan vain tietää mitä tekee. Ensiksi täytyy ymmärtää WordPressin rakenne, miten esimerkiksi ulkoasun ja sivupohjat sisältävä teema on erillään itse WordPressin julkaisujärjestelmästä. Tämä sisäistäminen vei itseltänikin aikaa ja hankalin osa tätä opinnäytetyötä olikin simppelin ja ymmärrettävän rakenteen luominen teemalle.

Toki teeman voi rakentaa eritavalla kuin itse tein. Olivathan erimerkkini vain yhden sivun teemoja. Kun käytin osaan sisällön hallinnasta Options Frameworkkia, samanlaisen ulkoasun sivustolle olisi voinut rakentaa esimerkiksi pelkillä julkaisuilla, jotka olisivat silmukan kautta kasattu yhdelle sivulle omiin muotteihinsa. Tapoja siis on monia.

WordPressille löytyy paljon valmiita laajennuksia, kuten teemoja ja lisäosia. Harvoin kuitenkaan mikään teema vastaa asiakkaan tarpeita suoraan, joten sen muokkaaminen on yleensä edessä kuitenkin. Tämän takia mielestäni hyvä tapa tutustua teemoihin onkin rakentaa itse oma. Yli 30 tuhannesta lisäosasta luulisi löytyvän sopiva melko moneen tarpeeseen, mutta ainahan voidaan tehdä parempia jo olemassa olevien lisäosien korvaajia, taikka välttää maksullisia lisäosia rakentamalla niitä itse.

Myös WordPressin multisite-ominaisuus osoittautui todella käytännölliseksi. Monet yksityiseen käyttöön hinnoitellut webhotellit sisältävät 1 tai 2 tietokantaa, ja täten useamman WordPress-sivuston luonti ilman multisite-ominaisuutta ei onnistu. Voisin esimerkiksi kuvitella tätä käytettävän opetustilanteessa, jolloin opettaja loisi WordPress-verkoston, johon oppilaat voisivat itse luoda teemoja ja lisäosia. Tällöin luokkaan ei tarvitsisi asentaa erikseen kolmeakymmentä WordPressiä.

Opinnäytetyö yleisesti ottaen onnistui melko hyvin, mutta lähti käyntiin melko hitaasti johtuen toimeksiantoni alussa vertailemistani eri toteutustavoista. Tutustuin opinnäytetyöni edetessä samalla myös responsiivisten sivustojen tekoon tarkoitettuun Bootstrap frameworkkiin, jota aionkin tulevaisuuden projekteissani hyödyntää

kaikissa mahdollisissa tilanteissa. Kaiken kaikkiaan opinnäytetyöni oli todella mielenkiintoinen ja opettavainen. Sain paljon uusia ideoita ja näkökulmia verkkosivustojen sekä -palveluiden toteutukseen ja WordPress tulee todellakin olemaan näissä mukana.

LÄHTEET

Cohen, Matt 2011. Introduction to Hooks: a basic WordPress building block. Blogi. http://wpcandy.com/teaches/how-to-use-wordpress-hooks/. Päivitetty 4.5.2011. Luettu 3.2.2015.

Price, Devin 2014. Options Framework Theme. Blogi. http://wptheming.com/options-framework-theme. Ei päivitystietoa. Luettu 12.12.2014.

Williams, Brad, Richard, Ozh & Tadlock, Justin 2011. Professional WordPress Plugin Development. Indianapolis: Wiley Publishing, Inc.

WordPress.org 2014a. About. WWW-dokumentti. https://wordpress.org/about/. Ei päivitystietoa. Luettu 27.11.2014.

WordPress.org 2014b. Themes Directory. WWW-dokumentti. https://wordpress.org/themes/. Ei päivitystietoa. Luettu 2.12.2014.

WordPress.org 2015. Plugin Directory. WWW-dokumentti. https://wordpress.org/plugins/. Ei päivitystietoa. Luettu 4.1.2015.

WordPress Codex 2014a. Themes. WWW-dokumentti. http://codex.wordpress.org/Themes. Ei päivitystietoa. Luettu 27.11.2014.

WordPress Codex 2014b. Functions File Explained. WWW-dokumentti. http://codex.wordpress.org/Functions_File_Explained. Ei päivitystietoa. Luettu 10.12.2014.

WordPress Codex 2014c. Post Type Templates. WWW-dokumentti. http://codex.wordpress.org/Post_Type_Templates. Ei päivitystietoa. Luettu 12.12.2014.

WordPress Codex 2014d. Page Templates. WWW-dokumentti. http://codex.wordpress.org/Page_Templates. Ei päivitystietoa. Luettu 13.12.2014.

WordPress Codex 2015a. Create A Network. WWW-dokumentti. http://codex.wordpress.org/Create_A_Network. Ei Päivitystietoa. Luettu 25.1.2015.

WordPress Codex 2015b. Before You Create A Network. WWW-dokumentti. http://codex.wordpress.org/Before_You_Create_A_Network#Server_Requirements. Ei päivitystietoa. Luettu 25.1.2015.

WordPress Codex 2015c. Multisite Network Administration. WWW-dokumentti. http://codex.wordpress.org/Multisite_Network_Administration#User_Access. Ei päivitystietoa. Luettu 30.1.2015.

WordPress Codex 2015d. Plugin API / Filter Reference / the_content. WWW-dokumentti. http://codex.wordpress.org/Plugin_API/Filter_Reference/the_content. Ei päivitystieto. Luettu 2.2.2015

Wpbeginner 2013. How to Properly Add JavaScripts and Styles in WordPress. Blogi. http://www.wpbeginner.com/wp-tutorials/how-to-properly-add-javascripts-and-styles-in-wordpress/. Päivitetty 2.7.2013. Luettu 11.12.2014.

Wpbeginner 2015. What is: Widgets. WWW-dokumentti. http://www.wpbeginner.com/glossary/widgets/. Ei päivitystieto. Luettu 4.2.2015

LIITE 1.

Yksisivuinen liite

```
function my_first_theme_scripts() {

    //get_template_directory_uri() gets the root folder of the theme
    wp_enqueue_style( 'bootstrap-core',  get_template_directory_uri() .
                      '/css/bootstrap.min.css' );

    wp_enqueue_style( 'fontawesome',  get_template_directory_uri() .
                      '/font-awesome/css/font-awesome.min.css' );

    wp_enqueue_style( 'squad-animate',  get_template_directory_uri() .
                      '/css/animate.css' );

    wp_enqueue_style( 'magnific-css',  get_template_directory_uri() .
                      '/css/magnific-popup.css' );

    wp_enqueue_style( 'squad-style',  get_template_directory_uri() .
                      '/css/style.css' );

    wp_enqueue_style( 'squad-colors',  get_template_directory_uri() .
                      '/color/default.css' );

    //get_style_sheet_uri() points to the style.css file in theme's root folder
    wp_enqueue_style( 'my-first-theme-style', get_stylesheet_uri() );

    /*the last three parameters are optional (if scripts depends on something,
     *script version number, and if script is placed in footer.php)*/
    wp_enqueue_script( 'my-first-theme-jquery', get_template_directory_uri() .
                       '/js/jquery.min.js', array(), false, true );

    wp_enqueue_script( 'bootstrap-javascript', get_template_directory_uri() .
                       '/js/bootstrap.min.js', array(), false, true );

    wp_enqueue_script( 'jquery-easing', get_template_directory_uri() .
                       '/js/jquery.easing.min.js', array(), false, true );

    wp_enqueue_script( 'jquery-scrollTo', get_template_directory_uri() .
                       '/js/jquery.scrollTo.js', array(), false, true );

    wp_enqueue_script( 'squad-wow', get_template_directory_uri() .
                       '/js/wow.min.js', array(), false, true );

    wp_enqueue_script( 'magnific-popup', get_template_directory_uri() .
                       '/js/jquery.magnific-popup.min.js', array(), false, true );

    wp_enqueue_script( 'squad-custom', get_template_directory_uri() .
                       '/js/custom.js', array(), false, true );

    wp_enqueue_script( 'my-first-theme-navigation', get_template_directory_uri() .
                       '/js/navigation.js', array(), '20120206', true );

    wp_enqueue_script( 'my-first-theme-skip-link-focus-fix',
                      get_template_directory_uri() .
                       '/js/skip-link-focus-fix.js', array(), '20130115', true );

    if ( is_singular() && comments_open() && get_option( 'thread_comments' ) ) {
        wp_enqueue_script( 'comment-reply' );
    }
}
add_action( 'wp_enqueue_scripts', 'my_first_theme_scripts' );
```

LIITE 2(1).

Monisivuinen liite

```
<?php
/**
 * The header for our theme.
 */
?><!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
<meta charset="<?php bloginfo( 'charset' ); ?>">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="profile" href="http://gmpg.org/xfn/11">
<link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">

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

<body id="page-top" data-spy="scroll" data-target=".navbar-custom">
    <!-- Preloader -->
    <div id="preloader">
      <div id="load"></div>
    </div>

    <nav class="navbar navbar-custom navbar-fixed-top" role="navigation">
        <div class="container">
            <div class="navbar-header page-scroll">
                <button type="button" class="navbar-toggle"
                        data-toggle="collapse" data-target=".navbar-main-collapse">
                    <i class="fa fa-bars"></i>
                </button>
                <a class="navbar-brand" href="index.html">

                </a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse navbar-right navbar-main-collapse">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#intro">Ylös</a></li>
        <li><a href="#about">
            <?php echo of_get_option('about_head', $default); ?>
            </a></li>
        <li><a href="#service">
            <?php echo of_get_option('review_head', $default); ?>
            </a></li>
      </ul>
            </div>
            <!-- /.navbar-collapse -->
        </div>
        <!-- /.container -->
    </nav>
```

LIITE 2(2).

Monisivuinen liite

```
<?php
/**
 * The main template file.
 */
get_header(); ?>

<!-- Section: intro -->
    <section id="intro" class="intro"
    style="background:
           url(<?php echo of_get_option('intro_bg_image', $default); ?>)
           no-repeat
           top
           center;">

        <div class="slogan">
            <h2><?php echo of_get_option('intro_head', $default); ?></h2>
            <h4><?php echo of_get_option('intro_subhead', $default); ?></h4>
        </div>
        <div class="page-scroll">
            <a href="#about" class="btn btn-circle">
                <i class="fa fa-angle-double-down animated"></i>
            </a>
        </div>
        <?php if ( !function_exists('dynamic_sidebar') ||
                    !dynamic_sidebar('my-widget-area') ) : endif; ?>
    </section>
    <!-- /Section: intro -->

    <!-- Section: about -->
    <section id="about" class="home-section text-center">
        <div class="heading-about">
            <div class="container">
            <div class="row">
                <div class="col-lg-8 col-lg-offset-2">
                    <div class="wow bounceInDown" data-wow-delay="0.4s">
                    <div class="section-heading">
                    <h2><?php echo of_get_option('about_head', $default); ?></h2>
                    <i class="fa fa-2x fa-angle-down"></i>

                    </div>
                    </div>
                </div>
            </div>
            </div>
        </div>
```

```
    <div class="container">

    <div class="row">
        <div class="col-lg-2 col-lg-offset-5">
            <hr class="marginbot-50">
        </div>
    </div>
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12 text-center">
            <div class="wow bounceInUp" data-wow-delay="1s">
            <div class="team boxed-white">
                <div class="inner">
                    <h5>
                        <?php echo of_get_option('about_name', $default); ?>
                    </h5>
                    <p class="subtitle">
                        <?php echo of_get_option('about_subtitle', $default); ?>
                    </p>
                    <div class="avatar">
                    <img src="wp-content/themes/my-first-theme/img/team/4.jpg"
                         alt="" class="img-responsive img-circle" />
                    </div>

                </div>
            </div>
            </div>
        </div>
    </div>
    </div>
</section>
<!-- /Section: about -->
```

```
    <!-- Section: services -->
    <section id="service" class="home-section text-center bg-gray">

        <div class="heading-about">
            <div class="container">
            <div class="row">
                <div class="col-lg-8 col-lg-offset-2">
                    <div class="wow bounceInDown" data-wow-delay="0.4s">
                    <div class="section-heading">
                    <h2><?php echo of_get_option('review_head', $default); ?></h2>
                    <i class="fa fa-2x fa-angle-down"></i>

                    </div>
                    </div>
                </div>
            </div>
            </div>
        </div>
        <div class="container">
        <div class="row">
            <div class="col-lg-2 col-lg-offset-5">
                <hr class="marginbot-50">
            </div>
        </div>
        <div class="row">

            <?php $loop = new WP_Query(
                array( 'post_type' => of_get_option('select_post_type', $default),
                                            'posts_per_page' => 4 )); ?>
            <?php while ( $loop->have_posts() ) : $loop->the_post(); ?>

              <div class="col-sm-3 col-md-3">
                <div class="wow fadeInLeft" data-wow-delay="0.2s">
                <div class="service-box">
                    <div class="service-icon popup"
                         href="<?php echo get_permalink(); ?>">
                        <?php echo get_the_post_thumbnail($post_id,
                                                    array(200,200)); ?>
                                                    <!-- Image size -->
                    </div>
                    <div class="service-desc">
                        <h5 href="<?php echo get_permalink(); ?>">
                            <?php echo get_the_title(); ?></h5>
                        <p><?php echo get_the_excerpt(); ?> </p>
                        <?php do_action('my_first_hook'); ?>
                    </div>
                </div>
                </div>
            </div>
            <?php endwhile; wp_reset_query(); ?>
        </div>
        </div>
    </section>
    <!-- /Section: services -->
<?php //get_sidebar(); ?>
<?php get_footer(); ?>
```

```
<?php
/**
 * The template for displaying the footer.
 */
?>

<footer>
        <div class="container">
            <div class="row">
                <div class="col-md-12 col-lg-12">
                    <div class="wow shake" data-wow-delay="0.4s">
                    <div class="page-scroll marginbot-30">
                        <a href="#intro" id="totop" class="btn btn-circle">
                            <i class="fa fa-angle-double-up animated"></i>
                        </a>
                    </div>
                    </div>
                    <p>&copy;Copyright 2014 - Squad. All rights reserved.</p>
                </div>
            </div>
        </div>
    </footer>

<?php wp_footer(); ?>
<script>
$(document).ready(function(){

    $('.popup').magnificPopup({
        type:'ajax',
        closeOnContentClick: false,
        closeOnBgClick: false,
        showCloseBtn: true,
        closeBtnInside: true
    });
});

</script>
</body>
</html>
```