

# Teeman luominen WordPress-julkaisualustalle



Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinna, Tietojenkäsittelyn koulutusohjelma

Kevät 2017

Mikko Kuusela

Tietojenkäsittelyn koulutusohjelma  
Hämeenlinna

---

**Tekijä** Mikko Kuusela **Vuosi** 2017

**Työn nimi** Teeman luominen WordPress-julkaisualustalle

**Työn ohjaaja/t** Tommi Saksa

---

## TIIVISTELMÄ

Opinnäytetyön tavoitteena oli luoda teema WordPress-sisällönhallintajärjestelmälle. Tavoitteena oli saada aikaan valmis teema, jota yritys voi tulevaisuudessa käyttää. Toimeksiantajana projektissa oli tamperelainen Mesumatto Oy, joka maahantuo ja myy näyttelymattoa erilaisiin tilaisuuksiin. Työn tarkoituksena oli myös tutkia ja vertailla sitä, mikä CSS-framework sopii parhaiten tällaista projektia varten.

Opinnäytetyö sisältää keskeisiä asioita WordPress-sisällönhallintajärjestelmästä ja siinä kerrotaan yleisesti miksi sisällönhallintajärjestelmiä kannattaa käyttää. Teknisessä osuudessa keskitytään pääasiassa WordPressin teemojen koodaustekniikkaan.

CSS-frameworkien vertailussa todettiin, että Kube oli sopivin yksinkertaisen teeman luomiseen. Opinnäytetyössä saatiin valmiiksi yksinkertainen WordPress-teema.

**Avainsanat** WordPress, HTML, CSS, PHP, sisällönhallintajärjestelmä

**Sivut** 29 sivua, joista liitteitä 1 sivu

Degree Programme in Business Information Technology  
Hämeenlinna

---

<b>Author</b>	Mikko Kuusela	<b>Year</b> 2017
<b>Subject</b>	Creating theme for WordPress content management system	
<b>Supervisors</b>	Tommi Saksa	

---

ABSTRACT

The goal of the thesis was to create theme to business to use with WordPress content management system. The client was carpet Importation Company from Tampere. The purpose of the thesis was also compare different CSS frameworks and choose what frameworks suites best for a project like this.

The thesis includes key things about WordPress and a general overview why content management systems should be used. The technical section focuses on the themes coding techniques and creating a simple theme.

In the comparison of CSS frameworks found out that Kube framework is the most suitable for creating a simple theme. A simple WordPress theme was made during this thesis.

**Keywords** WordPress, HTML, CSS, PHP, Content Management System

**Pages** 29 pages including appendices 1 pages

# SISÄLLYS

1	JOHDANTO.....	1
2	HTML-KIELI.....	2
2.1	Historia.....	2
2.2	HTML5.....	2
3	CSS-KIELI.....	3
3.1	Yleistietoa CSS-kielestä.....	5
3.2	CSS ja responsiivisuus.....	6
3.3	Frameworkin valinta minun projektiin.....	7
3.3.1	Bootstrap.....	8
3.3.2	Materializecss.....	8
3.3.3	Kube.....	8
3.3.4	Yhteenveto.....	8
4	JAVASCRIPT-KIELI.....	9
4.1	JQuery Javascript-kirjasto.....	9
4.2	Javascript verkkosivuilla.....	10
5	PHP-KIELI.....	11
6	SISÄLLÖNHALLINTAJÄRJESTELMÄ.....	12
6.1	Sisällönhallintajärjestelmän hyödyt.....	12
6.2	WordPress.....	13
7	PERUSTIETOA WORDPRESS-TEEMAN LUOMISESTA.....	14
7.1	WordPress-kehittäjän perustietoa.....	14
7.1.1	Vimpaimet.....	14
7.1.2	Koukut.....	14
7.2	Teemojen kansiorakenteet.....	15
7.3	Tärkeimmät tiedostot.....	16
7.3.1	Sivupohja-tiedostot.....	17
7.4	Muita yleisiä tiedostoja.....	19
7.5	The loop (silmukka).....	19
8	TOTEUTUS.....	20
8.1	Tavoitteet.....	20
8.2	WordPressin asennus.....	20
8.3	Messumatto-teeman luominen.....	21
8.3.1	Teeman functions.php-tiedosto.....	22
8.3.2	Header-, footer- ja sidebar-tiedostot.....	24
8.3.3	Index.php-tiedosto.....	26
8.3.4	Muita teeman tiedostoja.....	27
9	YHTEENVETO.....	28
	LÄHTEET.....	29

Liitteet

Liite 1      Search.php-tiedosto

## 1 JOHDANTO

Vuosien varrella sisällönhallintajärjestelmät ovat saaneet yhä suurempaa suosiota ja monet yritykset ja yksityishenkilöt ovat ottaneet sellaisen käyttöön. Nyt myös tamperelainen näyttelymattojen maahantuontiyritys tarvitsee uudet verkkosivut, joita on helppo yrityksen itse hallinnoida.

Tämän opinnäytetyön tavoitteena kertoa miten luodaan oma teema WordPress-sisällönhallintajärjestelmään. Aihe rajautuu pääosin WordPressin ympärille, mutta samalla kerron yleisesti verkkosivujen luomisesta ja mitä responsiivinen verkkosivu tarvitsee.

Kerron ensin yleisesti teoriaa web-kehityksessä tarvittavista menetelmistä ja niiden historiasta. Tämän jälkeen käytännön osuudessa näytän, miten WordPress asennetaan ja miten siihen luodaan oma teema.

Valitsin tämän aiheen, koska olen kiinnostunut web-kehityksestä ja minulla on aikaisempaa kokemusta sisällönhallintajärjestelmistä. Aiheessa kiinnostavaa on monet erilaiset koodikielet, johon opinnäytetyön aikana tutustutaan.

Opinnäytetyössä pääasiana on teeman luominen WordPress-julkaisualustalle, mutta samalla tehdään vertailua siitä, mikä CSS-framework sopii parhaiten projektin tueksi. Opinnäytetyössä kerrotaan myös mitä hyötyä sisällönhallintajärjestelmistä on sen käyttäjälle. Suurimpana tutkimuskysymyksenä kuitenkin on se, miten teema luodaan WordPress-julkaisualustalle.

## 2 HTML-KIELI

HTML (Hypertext Markup Language) on merkintäkieli, jonka avulla kuvataan verkkosivujen sisällön rakenne. Verkkosivuille voidaan sijoittaa monenlaista sisältöä, kuten tekstiä, grafiikkaa, lomakkeita, ääntä, videoita ja vaikkapa pelejä. Vaikka internetissä verkkosivut näyttävät erilaisilta, niitä kaikkia yhdistää yksi asia, ne kaikki on sidottu yhteen käyttämällä HTML-kieltä. (Tittel & Minnick 2013.)

### 2.1 Historia

Ensimmäinen HTML-luonnos syntyi vuonna 1991 Tim Berners-Leen toimesta. Vuonna 1994 IETF (Internet engineering Task Force) perusti ensimmäisen HTML-työryhmän ja sen kehitystyötä on HTML2. Vuonna 1996 IETF:n työryhmä lopetettiin ja jatkokehitys siirtyi W3C-järjestölle (World Wide Web Consortium), jonka Tim Berners-Lee perusti vuonna 1994. W3C julkaisi HTML 3.2:n ja HTML 4:n vuonna 1997 (W3 1998). Pari vuotta tämän jälkeen julkaistiin HTML 4.1. Vuonna 2014 HTML5-kielestä tuli W3C-järjestön suositus ja 2016 vuonna HTML 5.1:stä tuli W3C-kandidaatin suositus (W3 n.d.).

### 2.2 HTML5

HTML5 on uusin HTML-versio tällä hetkellä. HTML5 kehittyi paljon edeltäjänsä verrattuna. Uutena asiana kieleen tuli mediaelementtejä, grafiikoita, rakenteellisia elementtejä, lomake-elementtejä, uusia syötetyyppejä ja ominaisuuksia ja paljon muuta (W3schools n.d.).

Alla on esimerkki HTML-sivun rakenteesta. Koko sivusto tulee HTML-tagien väliin. Head-tägien väliin tulee esimerkiksi sivun otsikko, metatiedot ja viitaukset tyyliin. Yleensä Skripteihin viitataan juuri ennen body-tagin lopetusta. Body-tagien väliin tulee HTML-elementit.

```
<!DOCTYPE html>
<html>
<head>
<title>Esimerkkisivu</title>
<link href="tyyli.css" rel="stylesheet" type="text/css">
</head>
<body>

<h1>Tämä on otsikko!</h1>
<p>Tämä on leipätekstiä.</p>


<script src="jquery-3.1.1.min.js"></script>
</body>
</html>
```

### 3 CSS-KIELI

CSS (Cascading Style Sheets) on tyylikieli, jota käytetään muokkaamaan HTML-elementtien sijaintia ja tyyliä verkkosivuilla. HTML sitoo sivun elementit toisiinsa, kun taas CSS on se, joka antaa elementeille tyyliä, kuten vaikkapa värit, varjostukset, sijainnit ja yms.

CSS-kieltä voidaan kirjoittaa suoraan HTML-tiedostoon, mutta usein CSS-tyyleille tehdään oma, CSS-päätteinen tiedosto, johon viitataan HTML-tiedoston head-tagien välissä.

Alla on esimerkki CSS-tiedoston linkittämisestä HTML-tiedostoon:

```
<link href="tyyli.css" rel="stylesheet" type="text/css">
```

Alla on esimerkki tyylin määrittämisestä CSS-tiedostossa. Tämä määritys muuttaa sivun taustaväriä punaiseksi:

```
body {  
background-color: red;  
}
```

Tyylin voi määrittää HTML-tiedostoon myös ilman CSS-tiedoston linkittämistä. Se tapahtuu HTML-tiedostossa style-tagien välissä:

```
<style>  
body {  
background-color: red;  
}  
</style>
```

CSS-tyylejä voi kirjoittaa myös suoraan HTML-elementtiin. Alla on esimerkki tyyli ominaisuuden käyttämisestä HTML-elementissä. Esimerkissä kappaleen tekstin väri muutetaan punaiseksi.

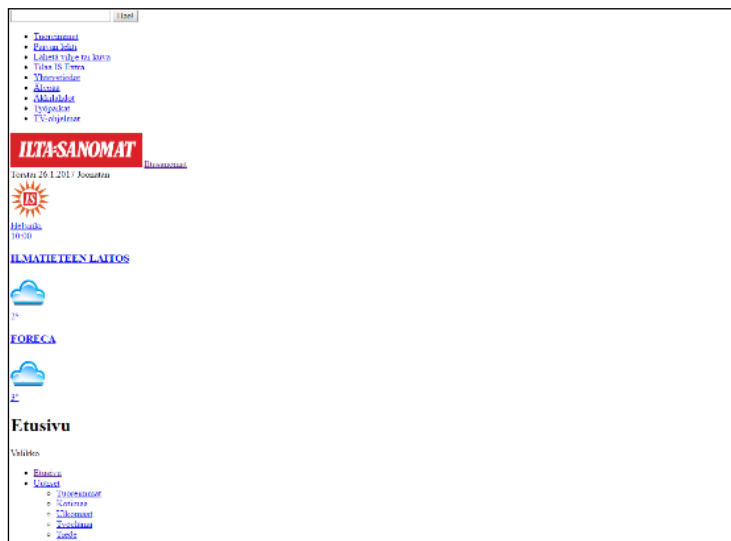
```
<p style="color: red;">Tämä on esimerkki kappale. </p>
```



Alla olevissa kuvissa (Kuva 1 & 2) näkyy miltä sivusto näyttää CSS-tyylien kanssa ja ilman tyylejä. Selviää, että kaikki muotoilut tulevat CSS-tyylien avulla.



kuva 1. Sivun CSS-tyyliä, kuvakaappaus.



kuva 2. Sivun ilman CSS-tyylejä, kuvakaappaus.

### 3.1 Yleistietoa CSS-kielestä

CSS1 julkaistiin ensimmäisen kerran vuonna 1996 ja siitä tuli W3C:n (World Wide Web Consortium) suositus. CSS1 antoi mahdollisuuden muuttaa fonttien muotoilua, taustavärejä, elementtien asemointia, leveyksiä, korkeuksia ja reunaviivoja. (Wium Lie & Bos 2005)

Kun CSS1 julkaistiin, monet selaimet eivät tukeneet CSS-tyylejä. Ensimmäinen kaupallinen selain, joka tuki CSS-tyylejä oli elokuussa 1996 julkaisu Internet Explorer 3. Internet Explorer 3 tuki suurinta osaa väreistä, taustoista ja tekstiominaisuuksista, mutta se ei tukenut laatikoita kuten esimerkiksi divejä, joiden avulla nykyään rakennetaan verkkosivujen ulkoasua. (Wium Lie & Bos 2005)

CSS2 julkaistiin W3C:n suosituksena vuonna 1998 (Wium Lie & Bos 2005). Se tehtiin CSS1-pohjalle ja siihen tuli paljon uusia ominaisuuksia kuten z-index, jolla voidaan määrittää päällekkäisyyksiä elementtien välillä. Muita tälläkin hetkellä tärkeitä web-ulkoasun muodostamisessa olevia uudistuksia olivat elementtien sijainnin muuttaminen, tekstien varjot, reunat, kursorit ja paljon muuta. (Hissom n.d.)

CSS3:n kehitys aloitettiin pian CSS2-julkaisun jälkeen ja ensimmäinen toimiva luonnos julkaistiin vuonna 2001. Koska CSS3 on niin suurikokoinen määrittely, se on julkaistu monessa eri moduulissa. Esimerkiksi taustoille ja reunoille on omat määrittämisensä. Samoin kuin animaatioille, jotka ovat täysin uusi CSS-määrittely. CSS3 kuitenkin sisältää myös kaikki vanhatkin määrittelyt (CSS3 introduction).

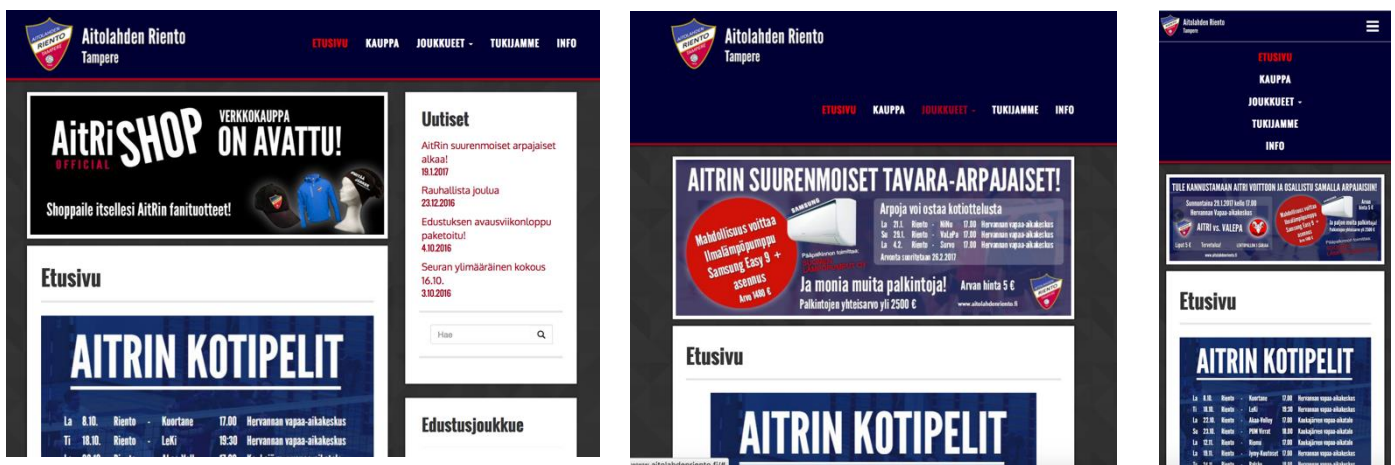
Tärkeimpiä CSS3-moduuleita ovat seuraavat:

- Valitsimet
  - Laatikkomallit
  - Taustat ja reunat
  - Teksti efektit
  - 2D ja 3D muunnokset
  - Animaatiot
  - Monipalstainen ulkoasu
  - Käyttöliittymä
  - Media Query.
- (CSS3 introduction.)

### 3.2 CSS ja responsiivisuus

Nykypäivänä verkkosivuja ei katsota enää pelkästään tietokoneelta, vaan katseluun käytetään paljon myös puhelimia ja tabletteja. Responsiivisten verkkosivujen tavoitteena on näyttää hyvältä ja mukautua kaikille laitteille. Pääosin responsiivinen ulkoasu toteutetaan CSS-tyylejä käyttäen, mutta apuna voidaan käyttää myös Javascriptiä. (Leiniö 2015)

Kuvassa 3 on esimerkki verkkosivujen skaalautuvuudesta selaimella. Kun sivu on skaalattu pienimmäksi, sen navigointipalkki muuttuu mobiililaitteelle sopivaksi valikoksi.



kuva 3. Responsiivinen ulkoasu verkkosivuilla, kuvakaappaukset.

CSS3-versiossa esiteltiin media queryt, joiden avulla voidaan määrittää CSS-tyylejä eri kokoisille näytöille. Tämä ominaisuus on keskeinen osa, kun luodaan responsiivista ulkoasua.

Alla on esimerkki media queryn käytöstä. Jos näytön leveys on alle 400 pikseliä, CSS-luokka "laatikon" leveys muuttuu 100 prosenttiin.

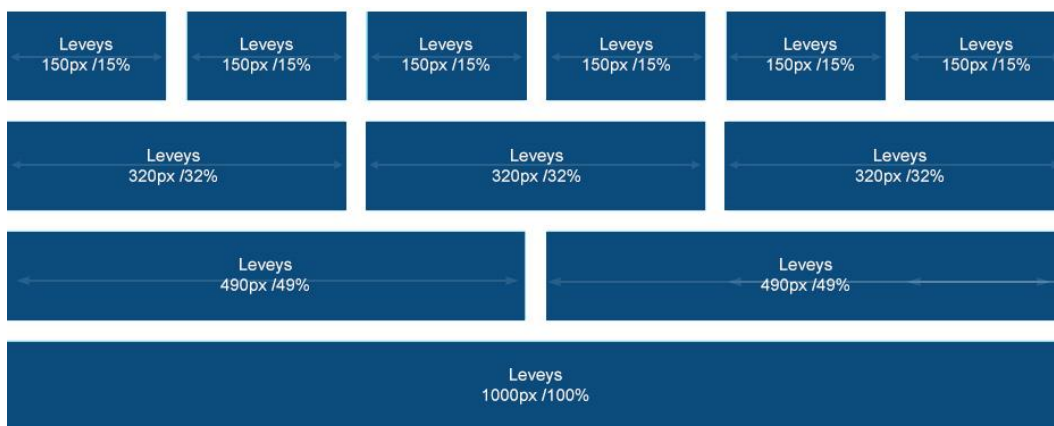
```
@media only screen and (max-width: 400px) {

    .laatikko {
        width: 100%;
    }

}
```

Monet responsiiviset verkkosivut käyttävät grid-tekniikkaa, jonka avulla sisältöä voidaan jakaa palstoihin (kuva 4). Gridien avulla verkkosivujen ulkoasusta saadaan tasapainoinen ja hyvännäköinen (Woods 2014). Gridit tehdään CSS-tyyleillä ja yleisin järjestelmä on 12 palstan järjestelmä, joka tarkoittaa sitä, että jopa 12 palstaa voidaan asettaa vierekkäin sivulle. Suosituimmista CSS-frameworkeista on valmiina responsiiviset gridit. Gridiä käytetään, koska se helpottaa responsiivisen ulkoasun luomista ja on osa sen kivijalkaa.

Alla on kuva 4, joka kuvastaa gridien käyttöä verkkosivuilla.



kuva 4. Gridit verkkosivuilla, kuvakaappaus.

### 3.3 Frameworkin valinta minun projektiin

Markkinoilla on suuri määrä valmiita CSS-frameworkeja, joita voi käyttää nopeuttaakseen verkkosivujen luomista. Monet verkkosivut ovat rakenteeltaan saman tyyppisiä. CSS-frameworkit tarjoavat verkkosivujen perusrakenteen, jotta kehittäjän ei tarvitse aloittaa verkkosivujen tekemistä alusta asti (Awwwards team 2016). CSS-frameworkit ovat valmiiksi luotuja tyylitiedostoja ja usein ne sisältävät myös Javascript-tiedoston. Ne ovat usein MIT-lisenssin alla, mikä tarkoittaa, että niitä saa käyttää vapaasti. Tässä kappaleessa vertailen muutamia varteenotettavia vaihtoehtoja ja tämän avulla teen päätöksen, mitä käytän projektissani.

Valintaan vaikuttaa moni asia, kuten se miten helppo frameworkista on saada oman näköinen. Yksi tärkeä tekijä on myös frameworkin koko. Projektissani teen pienet kevyet sivut yritykselle, joten suuret käyttöliittymään perustuvat frameworkit voi jättää pois vertailusta.

### 3.3.1 Bootstrap

Bootstrap on tällä hetkellä suosituin framework verkkosivujen rakentamista varten. Se sisältää CSS-tiedoston ja Javascript-tiedoston, jolla sivuihin saadaan dynaamisuutta.

Bootstrap toimii hyvin kaikilla laitteilla. Siitä löytyy myös valmiit LESS- ja Sass-tiedostot, joiden avulla pystyy nopeasti luomaan yksilölliset verkkosivut (Bootstrap n.d.). Bootstrap sisältää siis kaiken mitä isompikin verkkosivu tarvitsee, joten ei ihme, että se on suosituin.

### 3.3.2 Materializecss

Materializecss on Googlen luoma ja suunnittelema framework. Se on moderni framework, joka perustuu googlen Material Design –tyylikieleen, joka kehitettiin vuonna 2014. Se sisältää näyttäviä, pehmeitä animaatioita ja siirtymiä. Kuten myös Bootstrapissa, tämä framework sisältää myös Javascript-tiedoston, jonka avulla saa lisää dynaamisuutta tarvittaessa. (Materializecss n.d.)

Materializecss tarjoaa lähes kaiken mitä verkkosivu tarvitsee. Yksi asia tosin mietityttää tämän valinnassa, tuleeko verkkosivuista liian ”googlemaiset”, vaikka tyylejä pystyykin muuttamaan näppärästi ja nopeasti Sass-kie-len avulla.

### 3.3.3 Kube

Jos verrataan aikaisemmin esiteltyihin frameworkeihin, Kube on selvästi kaikista kevytrakenteisin. Kube on typografiaan ja minimalistisuuteen keskittyvä framework (Kube n.d.). Kaikki elementit ovat yksinkertaisesti tyyli-teltyjä. Vaikka Kube onkin yksinkertainen ja kevyt, silti se sisältää lähes kai-ken tarvittavan pienten ja keskikokoisten sivujen luomiseen.

Yksi asia, joka Kubesta puuttuu, on valikko, mutta onneksi internetistä löy-tyy runsaasti valmiita valikoita, joita voi käyttää frameworkin tukena. Tätä frameworkia kannattaa käyttää, kun rakennetaan yksinkertaista ulkoasua verkkosivuille.

### 3.3.4 Yhteenveto

Vaikka Materializecss ja ja Bootstrap ovat yhtä hyviä vaihtoehtoja sivujen luomiseen kuin Kube, päätän silti käyttää projektissani yksinkertaista Kube-frameworkia. Uskon, että se sopii parhaiten pienten ja kevyiden verkkosivujen luomiseen. Vaikka Kube ei sisälläkään valikkoa, minulle ei ole ongelma etsiä tai luoda sellaista itse.

## 4 JAVASCRIPT-KIELI

Javascript on kevytrakenteinen ohjelmointikieli, jolla verkkosivuista saadaan vuorovaikuttisempia. Javascript toimii kaikilla suosituimmilla selaimilla. Javascriptin käyttämisestä on myös hyötyä. Esimerkiksi vähemmän palvelinpuolen liikennettä, sillä vaikkapa lomakkeiden tarkistukset voidaan tehdä ilman, että tieto lähetetään palvelimelle. Javascript merkitään verkkosivuille Script-tagien väliin. (Tutorialspoint, n.d.)

### 4.1 JQuery Javascript-kirjasto

JQuery on Javascriptin avulla kehitetty ominaisuusrikas kirjasto (jQuery n.d.). JQueryn avulla voidaan tehdä täysin samoja asioita kuin Javascriptillä, mutta tekeminen on paljon nopeampaa, eikä se vaadi yhtä paljon ohjelmointia. Vaikka Javascriptin ja JQueryn koodikieli on erilaista, ne ovat silti molemmat Javascript-koodia. (Wilde, 2013.)

JQuery-koodi kirjoitetaan verkkosivuille samalla tavalla kuin normaali Javascript, eli script-tagien väliin, mutta se tarvitsee vielä viittauksen JQuery-kirjastoon.

Alla on esimerkki siitä, miten Javascriptillä lisätään tyyli HTML-elementtiin.

```
<script>
var d = document.getElementsByClassName("tyylimaaritys");
var i;
for (i = 0; i < d.length; i++) {
    d[i].className = d[i].className + " selected";
}
</script>
```

Alla oleva koodi tekee saman asian kuin edellinen esimerkki, mutta tämä on tehty JQueryä käyttäen.

```
<script src="jquery-3.1.1.min.js"></script>
<script>
$(".tyylimaaritys").addClass( "selected" );
</script>
```

Esimerkeistä selviää, että JQueryllä sama toiminto saadaan tehtyä paljon nopeammin ja pienemmällä vaivalla kuin normaalilla Javascriptillä.

## 4.2 Javascript verkkosivuilla

Javascriptin käyttäminen ei ole uusi asia verkkosivuilla. Sitä on käytetty jo pitkän aikaa. Se sai aikanaan huonon maineen, koska hakkerit lisäsivät haitallisia scriptejä omille sivuilleen. Nykyään Javascriptin käyttäminen on hyvin yleistä ja lähes jokaisella sivulla sitä on jossain muodossa.

Tutorialspointin (n.d.) mukaan Javascriptillä sivusta saadaan vuorovaikutteinen käyttäjän kanssa ja sillä pystytään lisäämään sivun dynaamisuutta. Esimerkiksi verkkosivuilla olevat kuvasarjat tai muuttuvat elementit on tehty Javascriptiä käyttäen. Usein myös lomakkeiden validoinnit on tehty Javascriptillä, ja netistä löytyy runsaasti valmiita koodeja lomakkeiden validointia varten. (Wilde, 2013.)

Verkkosivulle löytyy myös runsas määrä valmiita JQuery-lisäosia, joita voi hyödyntää omissa projekteissa.

## 5 PHP-KIELI

PHP (Hypertext Preprocessor) on ilmainen ja usein verkkosivuilla käytetty ohjelmointikieli, joka toimii palvelinpuolella (PHP, n.d.). Se tarkoittaa sitä, että koodi suoritetaan verkkosivujen palvelimella, kun taas esimerkiksi Javascript suoritetaan jokaisen kävijän omalla selaimella. PHP:n toimivuus ei siis ole selaimesta kiinni, vaan se toimii kaikilla selaimilla samalla tavalla. (Tutorialspoint, n.d.)

PHP-kieli on yleistä, kun tehdään web-pohjaisia sovelluksia, jossa pitää saada yhteys tietokantaan. Esimerkiksi WordPress ja monet muut sisällönhallintajärjestelmät on luotu PHP-kielillä.

Koska PHP-kieli ajetaan palvelimella, siitä ei jää jälkeä, jos joku tarkastelee sivuston lähdekoodia selaimella. PHP on hyödyllinen esimerkiksi silloin, kun pitää hakea tietoa tietokannasta verkkosivulle tai vaikkapa tehdä käyttäjätunnuksen ja salasanan validointi. PHP-kielen avulla on myös tehty valmiita sovelluskehysjä, joiden avulla voidaan nopeuttaa koodausprosessia. (Tutorialspoint, n.d.)

Rasmus Lerdof kirjoitti vuonna 1994 pienen kokoelman C-kielisiä skriptejä ja nimesi sen nimellä Personal Home Page Tools. Hän julkaisi kokoelmansa vuonna 1995 nimellä PHP/FI eli Personal Home Page / Forms Interpreter. Uusi versio siitä julkaistiin vuonna 1997 ja se sai tuhansia käyttäjiä ympäri maailmaa. Andi Gutmans ja Zeev Suraski kehittivät sen pohjalta PHP 3:n ja lyhenne sai nykyisen merkityksensä eli PHP: Hypertext Preprocessor. Vuonna 2000 julkaistiin PHP 4 ja vuonna 2004 julkaistiin PHP 5. (PHP n.d.) PHP 6 sisälsi merkistöstandardeihin liittyviä ongelmia, joten se sai huonon maineen ja sen kehittämistä ei jatkettu. Nykyisin uusi versio on PHP 7, mutta PHP 5:n kehittämistäkin jatketaan vielä. (Geniar 2016.)

Alla on esimerkki PHP-kielen käyttämisestä verkkosivulla. Tämä koodi tulostaa verkkosivuille tekstin "Moikka maailma!".

```
<!DOCTYPE html>
<html>
<body>
<?php echo "Moikka maailma!"; ?>
</body>
</html>
```



## 6 SISÄLLÖNHALLINTAJÄRJESTELMÄ

Kun ihmiset alkoivat luoda verkkosivuja vuosia sitten, ei ollut mahdollisuutta käyttää sisällönhallintajärjestelmää, koska sellaisia ei vielä ollut. Silloin varsinkin isojen sivujen päivitys oli raskasta, koska jokainen muutos piti tehdä erikseen jokaiselle sivulle, jos esimerkiksi vaikka haluttiin muuttaa sivun nimeä. (Amrit 2016.)

Sisällönhallintajärjestelmällä tarkoitetaan ylläpitotyökalua, joka helpottaa verkkosivujen sisällön muokkaamista tai tuottamista. Se nopeuttaa verkkosivujen kehittämistä, sillä kaikki toiminnallisuus on jo toteutettu valmiiksi. Käytännössä siis kehittäjä luo tai ostaa teeman sisällönhallintajärjestelmän päälle. (Casting 2015.)

Sisällönhallintajärjestelmän, kuten esimerkiksi WordPressin, avulla myös sellainen henkilö, jolla ei ole tuntemusta web-kehityksestä, voi luoda omat verkkosivut tai verkkokaupan käyttämällä valmista teemaa (Moveable Online 2013).

### 6.1 Sisällönhallintajärjestelmän hyödyt

Sisällönhallintajärjestelmät tuovat paljon hyötyä sen käyttäjälle. Sivusto pysyy organisoituna ja siistinä, eikä käyttäjän tarvitse miettiä miltä koodi näyttää. Se helpottaa myös isoja yrityksiä, joilla saattaa olla monia eri julkaisijoita. Jos sivuston ulkoasuun halutaan tehdä muutoksia, se käy näppärästi muutamaa koodinpätkää muokkaamalla tai parhaimmillaan jopa ilman koodin muokkaamista. (Moveable Online 2013.)

Sisällönhallintajärjestelmän avulla verkkosivuille voidaan luoda hakupalkki, koska sivuston sisältö on tallennettu tietokantaan, josta voidaan hakea tietoa eri hakusanoilla. Staattisilla verkkosivuilla taas tietoa ei yleensä ole tallennettu tietokantaan. (Amrit 2016.)

Sivujen päivittäminen onnistuu sisällönhallintajärjestelmän avulla kaikilla mahdollisilla älylaitteilla, jossa on selain. Käyttäjä voi päivittää sivuja, milloin ja missä vain. (Amrit 2016.)

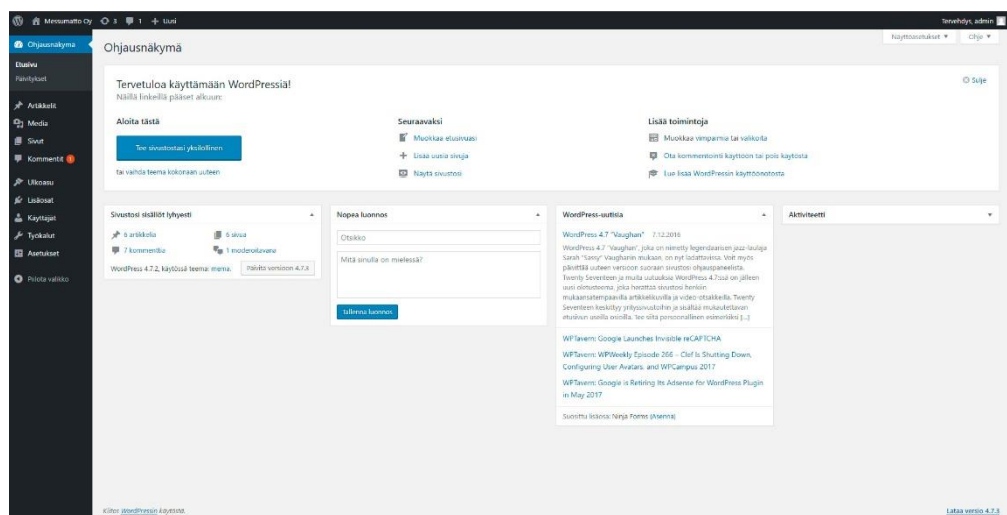
Käyttäjien rooleja ja oikeuksia voidaan muokata. Jollakulla käyttäjällä voi olla täydet käyttöoikeudet sivujen muokkaukseen, mutta toisella onkin vain pääsy artikkeleiden kirjottamiseen. (Amrit 2016.)

Lisäosien käyttö on helppoa, varsinkin jos sisällönhallintajärjestelmänä on jokin suosituimmista järjestelmistä. Esimerkiksi kuvagallerian voi nopeasti hakea sivuille lisäosista. Staattisissa sivuissa joutuisi itse koodaamaan tai etsimään netistä valmiita koodeja, joita yrittää upottaa omaan verkkosivuunsa. (Amrit 2016.)

## 6.2 WordPress

WordPress on selainpohjainen ja avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä. Se on yksi suosituimmista, ellei jopa suosituin blogien ja verkkosivujen hallintatyökalu. WordPress-alustalle löytyy paljon valmiita teemoja ja lisäosia, joten sen käyttämiseen ei välttämättä tarvitse lainkaan ohjelmointitaitoja. (Syed Balkhi 2015.)

Vuonna 2001 Michel Valdrighi aloitti B2 cafelogin kehittämisen. Pari vuotta tämän jälkeen Matt Mullenweg ja Mike Little muokkasivat B2:ta ja siitä syntyi WordPress. 2004 vuonna WordPress esitteli ensimmäisen kerran lisäosat ja vuonna 2005 teemat. Siitä lähtien WordPressiä on kehitetty ahkerasti tähän päivään saakka ja joka vuosi WordPress saa lisää ominaisuuksia. Tammikuussa 2017 WordPress 4.7 versiota, joka julkaistiin joulukuussa 2016, on ladattu jo 20 921 924 kertaa. (WordPress n.d.)



kuva 5. WordPress sisällönhallintajärjestelmän hallintapaneeli, kuva-kaappaus.

Ylhäällä kuva (kuva 5) WordPress-sisällönhallintajärjestelmän hallintapaneelista. Tämä on ensimmäinen näkymä, jonka käyttäjä näkee, kun kirjautuu muokkaamaan omaa verkkosivuaan.

## 7 PERUSTIETOA WORDPRESS-TEEMAN LUOMISESTA

Tässä osioissa kerron perusasiat, kun luodaan teemaa WordPressille, ja tämän tekstin lukemisen jälkeen oman teeman luominen on helpompi aloittaa. Lukija saa ymmärryksen teemojen rakenteesta ja käyttämisestä. En kerro verkkosivujen luomisesta nollasta alkaen, vaan siitä miten WordPressille luodaan teema.

### 7.1 WordPress-kehittäjän perustietoa

#### 7.1.1 Vimpaimet

Vimpaimet (widgets) ovat tapa näyttää sisältöä sisältöalueen ulkopuolella. Vimpaimia näytetään sivupalkkeissa, jotka on määritetty teemassa. Kun teemaa luodaan, tehdään vimpaimille alueita eli sivupalkkeja. Sivupalkkien ei välttämättä tarvitse sijaita nimenomaan sivuston sivussa, vaan se voi olla ihan missä vain. Kun teema on otettu käyttöön WordPressissä, voidaan sivupalkkeihin lisätä vimpaimia sisällönhallintapuolella.

Vimpainalueilla voidaan näyttää erilaista sisältöä, kuten vaikka viimeisimmät uutiset, joka on yksi WordPressin alkuperäisistä vimpaimista. WordPressin lisäosista löytyy myös paljon lisäosia vimpainalueille.

#### 7.1.2 Koukut

Koukut (hooks) ovat keino, joilla pääsee käsiksi WordPressin valmiisiin ydintoimintoihin. Pienillä koodinpätkillä voidaan muokata ja lisätä teemojen toiminnallisuuksia ja korvata sen mitä WordPress tekisi oletuksena. Koukkuja on kahdenlaisia, action-koukkuja ja filter-koukkuja. (Gordon 2015)

Action-koukut ovat funktioita, joilla voidaan, joko tarttua WordPressin omiin toiminnallisuuksiin tai luoda omia uusia toiminnallisuuksia (Gordon 2015). Koukut lisätään teeman functions.php-tiedostossa. Toiminnallisuuksia lisätään `add_action()`-funktioilla. Jos koukku ei tartu mihinkään valmiiseen funktioon ja se halutaan laukaista jollain kohtaa sivulla, se tapahtuu `do_action()`-funktioilla.

Alla on esimerkki siitä, miten action-koukku luodaan functions.php-tiedostossa. Funktio lisää `wp_footer`-koukkuun tekstin.

```
function footerTeksti() {  
    echo 'Messumatto Oy';  
}  
add_action('wp_footer', 'footerTeksti');
```

Filter-koukkuja taas käytetään muokkaamaan WordPressin tulostamaa sisältöä. Sen avulla voidaan esimerkiksi suodattaa halutut sanat tai vaikka lisätä haluttu teksti jokaisen artikkelin loppuun. (Gordon 2015)

Alla olevassa esimerkissä muutetaan artikkeleiden otteiden pituutta 15 merkkiin tarttumalla `excerpt_length`-koukkuun. WordPressin oletuksena otteen pituus on 55 merkkiä.

```
function otteen_pituus( $words ) {
    return 15;
}
add_filter( 'excerpt_length', 'otteen_pituus' );
```

## 7.2 Teemojen kansiorakenteet

Teemat löytyvät `wp-content`-kansion sisältä, kun tutkitaan WordPressin kansioita palvelinpuolella. Teeman kansiorakenteen ymmärtäminen helpottaa teemojen tekemistä. Teemojen kansiorakenteet voivat erota toisistaan, mutta niissä kaikissa on tietyt yhtäläisyydet, jotka WordPress vaatii. Kun aloittaa oman teeman tekemisen, kannattaa tutustua WordPressin luomiin teemoihin ja tutkia niiden kansiorakenteita. Niistä saa hyviä vinkkejä omaan teemaan ja niitä ei varmastikaan ole väärin tehty.

Alla on esimerkki (kuva 6) yksinkertaisen teeman kansiorakenteesta. ”`css`”-kansiossa ovat kaikki tyylitiedostot, joita teemassa käytetään. ”`inc`”-kansio sisältää funktioita, jotka liittyvät teemaan, mutta eivät sen ydintoimintaan. ”`js`”-kansio sisältää kaikki javascriptit, joita teemassa on. ”`template-parts`”-kansio sisältää sivupohjien osia, jotka voivat liittyä esimerkiksi sisällön tai hakutulosten näyttämiseen.

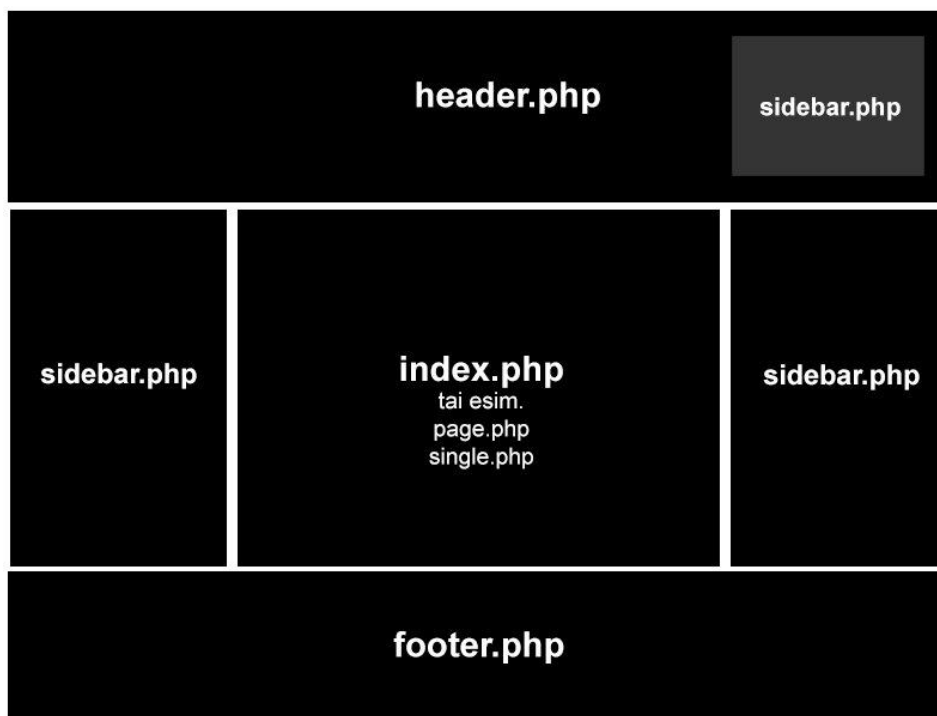
Nimi	Muokkauspäivä	Tyyppi	Koko
css	2.2.2017 0.35	Tiedostokansio	
inc	2.2.2017 0.35	Tiedostokansio	
js	2.2.2017 0.35	Tiedostokansio	
template-parts	2.2.2017 0.35	Tiedostokansio	
404.php	2.2.2017 0.35	PHP-tiedosto	2 kt
archive.php	2.2.2017 0.35	PHP-tiedosto	2 kt
comments.php	2.2.2017 0.35	PHP-tiedosto	3 kt
footer.php	2.2.2017 0.35	PHP-tiedosto	1 kt
functions.php	2.2.2017 0.35	PHP-tiedosto	5 kt
header.php	2.2.2017 0.39	PHP-tiedosto	2 kt
index.php	2.2.2017 0.35	PHP-tiedosto	2 kt
page.php	2.2.2017 0.35	PHP-tiedosto	1 kt
screenshot.png	2.2.2017 0.35	PNG-tiedosto	1 kt
search.php	2.2.2017 0.35	PHP-tiedosto	2 kt
sidebar.php	2.2.2017 0.35	PHP-tiedosto	1 kt
single.php	2.2.2017 0.35	PHP-tiedosto	1 kt
style.css	2.2.2017 0.35	CSS-tiedosto	15 kt

kuva 6. Teeman kansiorakenteesta, kuvakaappaus.

### 7.3 Tärkeimmät tiedostot

Tärkeimpiä tiedostoja teemassa ovat header.php, footer.php, index.php ja style.css. Pelkästään näillä neljällä tiedostolla voisi luoda oman yksinkertaisen teeman.

Alla on kuva 7, josta näkee missä tiedostot voisivat sijaita verkkosivuilla. Sivupalkkeja voi olla myös header.php- tai footer.php –tiedostoissa.



kuva 7. Tiedostojen sijainnit sivuilla, kuvakaappaus.

#### style.css

Style tiedosto on tyylitiedosto, joka vaaditaan jokaiseen WordPress-teemaan. Vaikka tämä onkin tyylitiedosto, ei sinne silti tarvitse laittaa sivun tyylejä. Ainoa asia, joka tähän tiedostoon on pakko laittaa, on teeman tiedot. Ne laitetaan tiedoston yläosaan kommentteina, kuten alla olevasta esimerkistä näkyy.

```
/*
Theme Name: Esimerkkiteema
Author: tekijännimi
Description: Tähän tulee teeman tietoa.
Version: 1.0
Tags: one-column, two-columns, custom-header, custom-menu,
responsive
Text Domain: esimerkkiteema
*/
```

### **header.php**

Header-tiedosto sisältää kaikki tiedot, jotka toistuvat jokaisella sivupohjalla, esimerkiksi metatiedot ja navigointi. Skriptit ja tyylitiedostot tuodaan functions.php-tiedostossa, wp\_enqueue\_scripts()-koukun avulla ja tuodaan header.php-tiedostoon wp\_head()-funktion avulla.

### **footer.php**

Footer-tiedosto on samanlainen tiedosto kuin header, mutta se sisältää kaikki tiedot, jotka ovat sivun alaosassa, kuten vaikka alapalkin. Footer-tiedostoon tuodaan myös skriptit wp\_footer()-funktion avulla.

### **sidebar.php**

Sidebar-tiedosto on sivupalkkia varten luotu tiedosto, johon voidaan WordPress-käyttäjäpuolella lisätä vimpaimia. Sivupalkkeja voi olla erilaisia eri kohdille sivua. Sivupohja-tiedostoon liitetään tämä tiedosto siihen kohtaan mihin sivupalkki halutaan.

### **functions.php**

Funktions-tiedostossa on kaikki teeman ydinfunktiot. Sivun tyylit ja scriptit lisätään funktioiden avulla header-tiedostoon, samoin sivupalkit ja navigoinnit rekisteröidään funktioiden avulla. Functions-tiedostossa voidaan myös lisätä teematukia (theme supports), joiden avulla voidaan lisätä ominaisuuksia teemaan, kuten esimerkiksi artikkelikuvien käyttäminen. Tässä tiedostossa viitataan myös inc-kansiossa oleviin muihin funktioihin.

#### 7.3.1 Sivupohja-tiedostot

Sivupohja-tiedosto on tiedosto, joka sisältää sivun sisällön, ja tässä tiedostossa viitataan header-, footer- ja sidebar -tiedostoihin, jotka tulostuvat tälle sivulle. Sivupohja-tiedostot sisältävät silmukan (loop), joka hakee sivuston sisällön tietokannasta ja näyttää sen sivuilla.

WordPressin teeman hierarkia toimii niin että, WordPress etsii sivupohjia tietyssä järjestyksessä ja jos muita sivupohjia ei löydy silloin se käyttää index.php-tiedostoa, joten index.php-tiedosto on ainoa pakollinen sivupohja-tiedosto. Esimerkiksi, silloin kun teemasta ei löydy muuta sivupohjaa kuin index.php, niin sitä käytetään, mutta jos teemasta löytyy esimerkiksi single.php, silloin sitä käytetään automaattisesti artikkelin sivupohjana.

Muita tärkeitä sivupohjia ovat `page.php`, jota käytetään yksittäisellä sivulla ja `single.php`, jota käytetään artikkelin sivupohjana. Sivupohjia voidaan luoda myös yksittäiseen artikkeliin tai sivuun. Se tapahtuu käyttämällä sivun `id`:tä tai tunnusta. Esimerkiksi ”info”-sivulle voidaan tehdä oma sivupohja tekemällä tiedosto, nimeltä `page-info.php` ja muokkaamalla sitä.

Alla on esimerkki WordPressin sivupohjien hierarkiasta (kuva 8). Esimerkissä on listattuna, mitä sivupohjaa WordPress etsii ensin, kun on kyse artikkeleista tai sivuista.

**Artikkeli**  
malliosoite.fi/malliartikkeli

1. `single-12.php` (Artikkelin `id`:llä)
2. `single-testiartikkeli.php` (artikkelin nimellä)
3. `single.php`
4. `singular.php`
5. `index.php`

**Sivu**  
malliosoite.fi/etusivu

1. `page-61.php` (Sivun `id`:llä)
2. `page-etusivu.php` (Sivun nimellä)
3. `page.php`
4. `singular.php`
5. `index.php`

kuva 8. Sivupohjien hierarkia.

Samanlainen hierarkia toimii myös muissa sivupohjissa, kuten esimerkiksi kategorioissa, taxonomioissa, tageissa ja muissa. Kun sivupohjia luodaan, on välttämätöntä tutustua WordPressin teemojen hierarkiaan ja siitä löytyvät hyvät ohjeet myös WordPressin sivuilta.

Teemoihin voidaan tehdä myös omia, täysin uusia sivupohjia, joita sitten voidaan käyttää WordPressin puolella, valitsemillaan sivuilla. Sivupohjat luodaan yleensä teeman `page-templates`-kansioon ja ne voidaan nimetä halutulla tavalla, mutta sivupohjan sisällä pitää olla kerrottu sivupohjan nimi. Sivupohjia käytetään, kun halutaan muokata sivun ulkoasua, esimerkiksi jos johonkin sivuun halutaan sisältö 100 %:n leveydelle.

## 7.4 Muita yleisiä tiedostoja

Yleensä teema tarvitsee muitakin tiedostoja, kuin vain tärkeimmät tiedostot, jotta sivuista saadaan toimivammat. Nämä tiedostot eivät kuitenkaan ole välttämättömiä sivujen toimivuuden kannalta.

### Search.php

Search.php-tiedosto on sivupohja-tiedosto, jota käytetään näyttämään hakutulokset.

### Searchform.php

Searchform.php on tiedosto, joka luodaan, jos halutaan muokata teeman oletus hakupalkkia.

### 404.php

404.php-tiedostoa käytetään silloin kun verkkosivuston sisältösivulla ei ole sisältöä.

### Comments.php

Jos artikkelien kommentointi on käytössä teemassa, silloin tätä käytetään tulostamaan kommentit.

## 7.5 The loop (silmukka)

Silmukka on WordPressin oletus mekanismi, jolla voidaan PHP-koodin avulla hakea sivun sisältö tietokannasta sille annettujen tagien avulla (WordPress n.d.). Teemassa voi olla monta erilaista silmukkaa eri tilanteisiin, esimerkiksi artikkelin näyttämiseen ja hakutulosten näyttämiseen omansa. silmukan voi kirjoittaa suoraan sivupohjaan tai ne voidaan kirjoittaa erilliseen php-tiedostoon "template-parts"-kansion sisälle. Jos loopit on luotu "template-parts"-kansioon, niin silloin se pitää tuoda sivupohjaan käyttämällä `get_template_part()`-funktia.

Alla on esimerkki erittäin yksinkertaisesta looppista, joka näyttää otsikon, tekstisisällön ja kirjoittajan nimen. `The_title()`, `the_content()` ja `the_author()` ovat tagejä, joiden avulla saadaan haluttua tietoa haettua tietokannasta.

```
if (have_posts()) :
    while (have_posts()) :
        the_post();
        the_title();
        the_content();
        the_author();
    endwhile;
endif;
```



## 8 TOTEUTUS

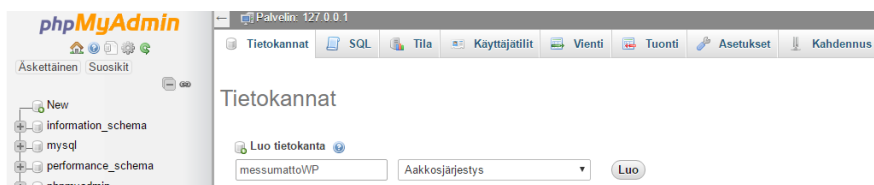
### 8.1 Tavoitteet

Tavoitteena on luoda yksinkertainen WordPress-teema Messumatto Oy – yrityksen käyttöön. Toteutus osuudessa kerrotaan teeman luominen tiedosto kerrallaan, jotta toteutus osuus olisi selkeämpi sisällöltään.

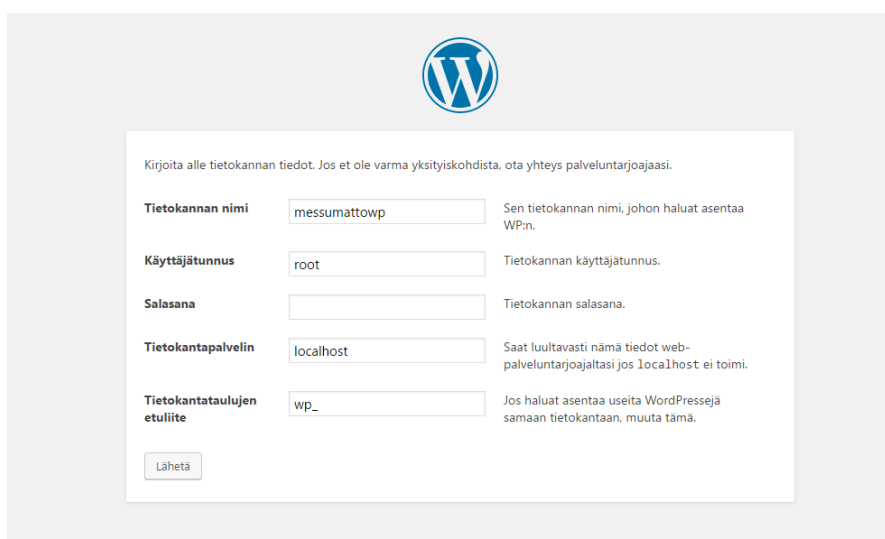
Oikaisen toteutusosuudessa sen verran, että käytän valmista CSS-tyylitiedostoa sivuja luodessa. CSS-tyylitiedostona toimii Imperavin luoma Kube, jota aikaisemmin opinnäytetyössä vertailtiin. Se on yksinkertainen ja kevytrakenteinen tyylitiedosto, joka on valmiiksi mobiiliystävällinen ja joka sopii hyvin tähän projektiin.

### 8.2 WordPressin asennus

WordPressin sivuilta (wordpress.org) voi ladata WordPress-asennusohjelman pakattuna zip-tiedostona. Zip-tiedosto puretaan ja siirretään omalle palvelimelle. Omalla palvelimella on oltava mahdollisuus luoda SQL-tietokantoja (kuva 9). Kun WordPress on siirretty palvelimelle ja tietokanta luotu, siirrytään selaimella siihen osoitteeseen, jossa WordPress sijaitsee. Kun osoitteeseen on siirrytty, WordPress aloittaa automaattisesti asennusohjelman.



kuva 9. Tietokannan luominen phpMyAdminin avulla, kuvakaappaus.



kuva 10. Asennusohjelma kysyy tietokannan tiedot, kuvakaappaus.

Asennusohjelma kysyy ensin tietokannan nimeä, käyttäjätunnuksia ja tietokannan taulun etuliitettä, joka voi olla oletusarvona, jos samaan tietokantaa asennetaan vain yksi WordPress-järjestelmä (kuva 10).

Seuraavana kysytään sivuston nimeä, pääkäyttäjän tunnuksia ja sallitaanko sivuston näkyminen hakukoneissa (Kuva 11).

kuva 11. Asennus ohjelma kysyy tarvittavat tiedot, kuvakaappaus.

Nämä tiedot riittävät ja WordPress viimeistelee asennuksensa, jonka jälkeen voidaan kirjautua WordPressin sisällönhallintapuolelle.

### 8.3 Messumatto-teeman luominen

Ensimmäisenä siirrytään wp-content-kansioon ja luodaan themes-kansion sisälle teemalle kansio nimeltä "messumatto". Tämän jälkeen luodaan messumatto-kansion sisälle muut tarvittavat kansiot kuten, css-, js, img yms. Sitten luodaan style.css-tiedosto, johon lisätään alla olevat teeman tiedot.

```
/*
Theme Name: Messumatto Oy
Author: Mikko Kuusela
Description: Teema, joka on luotu Messumatto Oy yrityksen käyttöön.
Version: 1.0
Tags: one-column, two-columns, custom-header, custom-menu, responsive
Text Domain: messumatto
*/
```

Kun kansiorakenne on kunnossa, otetaan teema käyttöön WordPressissä, jotta nähdään teemaan luodut muutokset nopeasti samalla, kun muokataan teeman koodia.

### 8.3.1 Teeman functions.php-tiedosto

Teeman luomisen voi aloittaa functions.php-tiedostosta. Ensimmäisenä functions-tiedostossa muokataan `after_setup_theme()`-koukkuja ja lisätään teematuet ja rekisteröidään valikot. Tässä vaiheessa on hyvä tietää paljonko sivupalkkeja ja valikoita teemaan tulee. Valikoita ja sivupalkkeja voidaan myös lisätä myöhemminkin.

Alla olevassa esimerkissä ensimmäisenä varmistetaan, että `messumatto_alustus()`-funktio on olemassa. Se tapahtuu `if`-lausekkeella ja `function_exists()`-funktioilla, joka tarkistaa löytyykö sille parametrina annettua funktiota. Tämä tarkistus on suositeltavaa tehdä jokaisen funktion ympärille functions.php-tiedostossa. Tarkistuksen jälkeen lisätään teematuet ja rekisteröidään valikko, jonka nimeksi on annettu ”päävalikko”. `Messumatto_alustus()`-funktion jälkeen tartutaan `after_setup_theme()`-koukkuun ja lisätään `Messumatto_alustus()`-funktio siihen. `After_setup_theme()`-funktio ajetaan silloin, kun teema ladataan selaimessa.

```

if ( ! function_exists ( 'messumatto_alustus' ) ) :

function messumatto_alustus () {

    // Lisätään hyödylliset teematuet
    add_theme_support ( 'automatic-feed-links' );
    add_theme_support ( 'title-tag' );
    add_theme_support ( 'post-thumbnails' );
    add_theme_support ( 'custom-logo' );

    add_theme_support ( 'html5', array(
        'search-form',
        'comment-form',
        'comment-list',
        'gallery',
        'caption',
    ) );

    //rekisteröidään valikko
    register_nav_menus (
        array(
            'header-menu' => __( 'Päävalikko' )
        )
    );

    // Lisätään mahdollisuus muuttaa taustakuvaa
    add_theme_support ( 'custom-background', apply_filters(
        'messumatto_oy_custom_background_args', array(
            'default-color' => 'ffffff',
            'default-image' => '',
        ) ) );
}
endif;
add_action ( 'after_setup_theme', 'messumatto_alustus' );

```

Alla olevassa esimerkissä rekisteröidään teemaan sivupalkit. Rekisteröiminen tapahtuu `register_sidebar()`-funktiolla, johon voidaan lisätä parametreja listana. Rekisteröinnin jälkeen tartutaan `widgets_init()`-koukkuun ja lisätään `messumatto_sivupalkit()`-funktio siihen.

```
//Sivupalkkien lisääminen teemaan
function messumatto_sivupalkit() {
    register_sidebar( array(
        'name'           => 'Alapalkin vimpain',
        'id'             => 'sidebar-1',
        'description'    => 'Lisää vimpaimia.',
        'before_widget' => '<section class="widget %2$s">',
        'after_widget'  => '</section>',
        'before_title'  => '<h2 class="widget-title">',
        'after_title'   => '</h2>',
    ) );

    register_sidebar( array(
        'name'           => 'Yläpalkki 2',
        'id'             => 'sidebar-2',
        'description'    => 'Lisää vimpaimia',
        'before_widget' => '<section class="widget %2$s">',
        'after_widget'  => '</section>',
        'before_title'  => '<h2 class="widget-title">',
        'after_title'   => '</h2>',
    ) );

    //...
}

add_action( 'widgets_init', 'messumatto_sivupalkit' );
```

Tyylitiedostot lisätään teemaan `functions.php`-tiedostossa ja se tapahtuu tarttumalla `wp_enqueue_scripts()`-koukkuun. Alla olevassa esimerkissä näkyy `messumatto_skriptit()`-funktio, jossa lisätään tyyli- ja skriptit `wp_enqueue_style()`- ja `wp_enqueue_script()`-funktioiden avulla. `Wp_enqueue_script()`-funktion lopussa, skriptille annetaan parametri `"true"`, joka tarkoittaa sitä, että se tulostuu `footer.php`-tiedostossa, `wp_footer()`-funktion avulla.

```
function messumatto_skriptit() {
    //Tyyli tiedostot
    wp_enqueue_style( 'messumatto-oy', get_stylesheet_uri() );
    wp_enqueue_style( 'kube', get_template_directory_uri() .
        '/css/kube.css', array(), '1.0.0', "all" );
    wp_enqueue_style( 'custom', get_template_directory_uri() .
        '/css/custom.css', array(), '1.0.0', "all" );
    wp_enqueue_style( 'font-awesome', get_template_directory_uri() .
        '/css/font-awesome.min.css', array(), '1.0.0',
        "all" );

    //Skriptit

    wp_deregister_script( 'jquery' );
    wp_enqueue_script( 'jquery', get_template_directory_uri() .
        '/js/jquery.js', array(), null, true );
```

```

wp_enqueue_script( 'script', get_template_directory_uri()
. '/js/pushy.min.js', array ( ), 1.0, true);
wp_enqueue_script( 'search', get_template_directory_uri()
. '/js/search.js', array ( ), 1.0, true);
}

add_action( 'wp_enqueue_scripts', 'messumatto_skripit' );

```

### 8.3.2 Header-, footer- ja sidebar-tiedostot

Functions-tiedoston jälkeen kannattaa luoda header.php-tiedosto. Header-tiedostoon lisätään kaikki tarvittava, mitä verkkosivujen head-tagien väliin tulee, kuten esimerkiksi meta-tiedot, otsikko ja tyylit.

Minun teemassani header.php sisältää myös navigoinnin ja hakupalkin, koska se toistuu jokaisella sivulla.

Alla on esimerkki siitä, mitä head-tagien väliin tulee header.php-tiedostossa. Wp\_head()-funktion avulla tuodaan sivun tyylit, jotka lisättiin aikaisemmin functions.php-tiedostossa. Tällä funktiolla tuodaan myös sivun otsikko, koska functions-tiedostoon on lisätty teematuki tittle-tagille.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="<?php bloginfo( 'charset' ); ?>">
<meta content="Messu-ja näyttelymattojen maahantuonti."
name="description">
<meta content="width=device-width, initial-scale=1, user-
scalable=no" name="viewport">
<?php wp_head(); ?>
</head>

```

Alla on esimerkki, miten haku ominaisuus lisätään header.php-tiedostossa. Suunnittelemani sivussa hakulaatikko avautuu uutena ikkunana, kun painetaan hakupainiketta valikosta. Hakupalkki saadaan tuotua get\_search\_form()-funktiolla.

```

<div id="search">
    <div class="closebtn fa fa-close" id="close"></div>
<?php get_search_form(); ?>
</div>

```

Teemassani on teematuki kustomoidulle logolle, joka tarkoittaa sitä, että käyttäjä pystyy itse muuttamaan sivuston logoa. Alla olevassa esimerkissä näkyy, että logo tuodaan valikko palkkiin the\_custom\_logo()-funktiolla.

Alla olevassa esimerkissä näkyy myös, miten valikko tuodaan wp\_nav\_menu()-funktion avulla. Ennen funktion käyttöä, selvitetään, löytyykö "header-menu"-nimistä valikkoa. Se tapahtuu if-lauseella ja

has\_nav\_menu()-funktioilla. Jos valikko on olemassa wp\_nav\_menu()-funktio tuo HTML-listana sivuston sivujen linkit. Tähän funktioon voi lisätä parametreja listana, joiden avulla voidaan muuttaa luokkien nimiä yms.

```
<nav>
<div class="navigation">
  <div class="logo" style="float:left;">
    <!-- Kustomoitu logo -->
    <?php
      if ( function_exists( 'the_custom_logo' ) ) {
        the_custom_logo();
      }
    ?>
  </div>
  <div class="pushy pushy-left" data-focus="#first-link">
    <div class="pushy-content">
      <ul>
        <li class="pushy-link searchicon" id="myBtn">
          <a href="#search"><i aria-hidden="true" class="fa
fa-search"></i></a>
        </li>
        <!-- Valikon tuonti -->
        <?php
          if (has_nav_menu('header-menu')){
            wp_nav_menu(array(
              'theme_location' => 'header-menu',
              'items_wrap' => '%3$s',
              'container' => false,
              'menu_class' => 'pushy-content'
            ));
          }
        ?>
      </ul>
    </div>
  </div>
</nav>
```

Footer.php-tiedostossa on kaikki mitä alapalkkiin tarvitaan. Tässä teemassa alapalkkiin tulee vain yksi sivupalkki, joka näkyy alla olevassa esimerkissä. Sivupalkki on tuotu get\_sidebar()-funktioilla, jolle annetaan parametrimina halutun sivupalkin id-tunnus. Wp\_footer()-funktion avulla saadaan mm. skriptit, jotka oltiin lisätty aikaisemmin functions.php-tiedostossa.

```
<footer>
  <section class="container">
    <?php if ( is_active_sidebar( 'sidebar-1' ) ) { ?>
      <div class="row gutters">
        <div class="col col-12 text-center">
          <?php get_sidebar("1") ?>
        </div>
      </div>
    <?php } ?>
  </section>
</footer>
</div> <!-- Site-wrap päättyy -->
<?php wp_footer(); ?>
</body>
</html>
```

Sidebar.php-tiedostossa testataan ensin, onko sivupalkki aktiivinen eli käytössä teemassa. Jos se on aktiivinen, ajetaan `dynamic_sidebar()`-funktio, joka saa parametrina sivupalkin nimen. `Dynamic-sidebar()`-funktio tulostaa sivupalkin sivulle. Alla on koodi, jonka sidebar-tiedosto sisältää.

```
<?php if ( is_active_sidebar( 'sidebar-1' ) ) : ?>
    <aside class="widget-area" role="complementary">
        <?php dynamic_sidebar( 'sidebar-1' ); ?>
    </aside>
<?php endif; ?>
```

### 8.3.3 Index.php-tiedosto

Seuraavana on `index.php`, joka on teeman tärkein sivupohja-tiedosto. `index.php` sisältää viittauksen header-, footer- ja mahdollisiin sidebar-tiedostoihin. Alla olevassa esimerkissä näkyy, miten header- ja footer-tiedostot tuodaan `get_header()`- ja `get_footer()`-funktioilla. Esimerkissä näkyy myös yksinkertainen silmukka, jolla sivun tekstisisältö tuodaan.

```
<!--Tuodaan header.php-tiedosto -->
<?php get_header(); ?>

<section class="smallshowcase">
    <?php if ( have_posts() ) :?>
        <h1 class="title"><?php single_post_title(); ?></h1>
    <?php endif; ?>
</section>

<section class="container">
    <div class="row gutters mainteksti">
        <div class="col col-12 text-center">
            <!--Silmukka, jolla tuodaan sisältö -->
            <?php
                if ( have_posts() ) : while ( have_posts() ) : the_post();
                    the_content();
                endwhile;
            <!--Silmukka päättyy -->

                else :
                    echo "Sisältöä ei löytynyt.";
                    get_search_form();
                endif;
            ?>

        </div>
    </div>
</section>

<!--Tuodaan footer.php-tiedosto -->
<?php get_footer(); ?>
```

### 8.3.4 Muita teeman tiedostoja

404.php on tiedosto, joka näytetään, kun mitään sivua ei löydy. Tiedosto tulostaa "sivua ei löytynyt" ja sen jälkeen näyttää hakupalkin, joka tuodaan get\_search\_form()-funktioilla. Alla on 404.php-tiedoston koodi.

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

<section class="smallshowcase">
  <h1 class="title">Sivua ei löytynyt</h1>
</section>

<section class="container">
  <div class="row mainteksti">
    <div class="col col-4 push-center text-center">
      <h3>Kokeile hakea jotain muuta </h3>
      <?php get_search_form(); ?>
    </div>
  </div>
</section>

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

Search.php-tiedosto (liite 1) tulostetaan kun sivuston kävijä tekee haun sivuilta. Tiedostossa on silmukka, joka tulostaa hakutulokset ja näyttää otteen jokaisesta tuloksesta. Silmukka on käytännössä samanlainen kuin index.php-tiedostossa. Jos hakutuloksia ei ole, näytetään viimeisimmät uutiset ja hakupalkki.

Teemaan voisi vielä tehdä runsaasti muita sivupohjia, mutta koska messumatto Oy:n verkkosivuilla ei ole artikkeleita, on turhaa tehdä erikseen niitä varten sivupohjia.



## 9 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda teema WordPress-sisällönhallintajärjestelmälle. Yksinkertaisen teeman luominen onnistui hyvin CSS-frameworkin avulla, jota vertailtiin opinnäytetyön aikana. Opinnäytetyössä tehty teema on vielä erittäin yksinkertainen, mutta siitä on hyvä jatkaa kehitystä ja mahdollisesti saada vielä laaja kokonaisuus. Jatkokehityksenä teemaan tehdään etusivulle oma sivupohja ja lisätään mahdollisuuksia muuttaa teemaan ulkoasua WordPressin mukauttimen avulla. Tulevaisuudessa teeman avulla luodaan uudet Messumatto Oy -yrityksen uudet verkkosivut.

Vaikka minulla oli aikaisempaakin kokemusta teemojen luomisesta, opin silti paljon uutta, kuten esimerkiksi syvennyin teemojen hierarkiaan paremmin kuin aikaisemmin. Sain myös kiinnostavaa tietoa sisällönhallintajärjestelmien hyödyistä ja toivon, että se oli myös lukijalle hyödyllistä.

Tutkimuskysymyksiin vastaaminen onnistui ja lukija saa vastauksen kaikkiin tutkimuskysymyksiin. Tämän opinnäytetyön pohjalta lukijalle selviää, mitä asioita on hyvä ottaa huomioon omaa teemaa luodessa.

Tämän opinnäytetyön avulla, aiheesta kiinnostunut henkilö saa hyvän pohjan teemojen kehitykseen ja oman teeman kehittämisen aloittaminen on helpompaa.

## LÄHTEET

Amrit, R. (2016). 15 CMS benefits. Blogijulkaisu 27.9.2016. Haettu 28.1.2017 osoitteesta <https://www.raycreationsindia.com/cms-benefits/>

Awwwards team (2016). What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. Blogijulkaisu 20.2.2016. Haettu 26.1.2017 osoitteesta <http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html>.

Balkhi, S. (2015). Why You Should Use WordPress? Blogijulkaisu 7.6.2015. Haettu 30.1.2017 osoitteesta <http://www.wpbeginner.com/why-you-should-use-WordPress/>

Bootstrap (n.d.). Bootstrap. Haettu 8.2.2017 osoitteesta <http://get-bootstrap.com/>

Casting, C. (2015). What is CMS? Blogijulkaisu 27.8.2015. Haettu 30.1.2017 osoitteesta <http://learn.onemonth.com/what-is-cms>

Geniar, M. (2015). PHP 6: The Missing Version Number. Blogijulkaisu 18.1.2015. Haettu 30.1.2016 osoitteesta <https://ma.ttias.be/php6-missing-version-number/>

Gordon, Z. (2015). WordPress Hooks: Actions, Filters, and Examples. Blogijulkaisu 4.2.2015. Haettu 8.2.2017 osoitteesta <http://blog.teamtreehouse.com/hooks-wordpress-actions-filters-examples>

Hissom, A. (n.d.). History of HTML. Haettu 24.1.2017 osoitteesta <http://amyhissom.com/HTML5-CSS3/history.html>

JQuery (n.d.). What is JQuery. Haettu 27.1.2017 osoitteesta <https://jquery.com/>

Kube (n.d.). Kube. Haettu 8.2.2017 osoitteesta <https://impe-ravi.com/kube>

Leiniö, T. (2012). Mitä on responsiivinen design? Blogijulkaisu 19.7.2012. Haettu 28.1.2017 osoitteesta <https://www.sofokus.com/blogi/mita-on-responsiivinen-design/>

Materializecss (n.d.). Materializecss. Haettu 8.2.2017 osoitteesta <http://materializecss.com/>

Meyerweb (n.d.). New properties in CSS2. Haettu 12.1.2017 osoitteesta <http://meyerweb.com/eric/articles/webrev/199802a1.html>

Moveable Online (2013). 7 Advantages of Using a CMS to Run Your Site. Blogijulkaisu 29.10.2013. Haettu 24.1.2017 osoitteesta <http://moveableonline.com/blog/2013/10/29/7-advantages-using-cms-run-site/>

PHP (n.d.). What is PHP. Haettu 30.1.2017 osoitteesta <http://php.net/manual/en/intro-what-is.php>

PHP (n.d.). History of PHP. Haettu 30.1.2017 osoitteesta <http://php.net/manual/en/history.php.php>

Tittel, E. & Minnick, C. (2013). Beginning HTML5 and CSS3 For Dummies. Hoboken: John Wiley & Sons, Inc.

Tutorialspoint (n.d.). PHP introduction. Haettu 30.1.2017 osoitteesta [https://www.tutorialspoint.com/php/php\\_introduction.htm](https://www.tutorialspoint.com/php/php_introduction.htm)

Tutorialspoint (n.d.). What is Javascript? Haettu 20.1.2016 osoitteesta [https://www.tutorialspoint.com/javascript/javascript\\_overview.htm](https://www.tutorialspoint.com/javascript/javascript_overview.htm)

Wium Lie, H. & Bos, B. (2005). Cascading Style Sheets, designing for the Web, Third Edition. Addison-Wesley Professional.

W3 (1998). History of HTML. Haettu 18.1.2017 osoitteesta <https://www.w3.org/People/Raggett/book4/ch02.html>

W3 (n.d.) All standards and drafts Haettu 26.2.2016 osoitteesta [https://www.w3.org/TR/#tr\\_HTML](https://www.w3.org/TR/#tr_HTML)

W3schools (n.d.). HTML5 new elements. Haettu 25.1.2017 osoitteesta [http://www.w3schools.com/html/html5\\_new\\_elements.asp](http://www.w3schools.com/html/html5_new_elements.asp)

Wilde, B. (2013). JQuery vs. JavaScript: What's the Difference Anyway? Blogijulkaisu 31.10.2013. Haettu 20.1.2017 osoitteesta <https://blog.udemy.com/jquery-vs-javascript/>

WordPress (n.d.). History. Haettu 28.1.2017 osoitteesta <https://codex.wordpress.org/History>

Woods, S (2014). 10 TOP PRINCIPLES OF EFFECTIVE WEB DESIGN. Blogijulkaisu 4.3.2014. Haettu 30.1.2017 osoitteesta <http://shortiedesigns.com/2014/03/10-top-principles-effective-web-design/>

WordPress (n.d.) The loop. Haettu 17.2.2017 osoitteesta <https://developer.wordpress.org/themes/basics/the-loop/>

## Search.php-tiedosto

```

<?php get_header(); ?>

<section class="smallshowcase">
    <?php if ( have_posts() ) : // Jos tuloksia löytyy?>
        <h1 class="title">Haku tulokset</h1>
    <?php else :?>
        <h1 class="title">Ei hakutuloksia</h2>
    <?php endif; ?>

</section>

<section class="container">
    <div class="row gutters mainteksti">
        <div class="col col-12 text-center">
            <div class="content-area">

                <?php if (have_posts()) : // Jos tuloksia löytyy ?>

                    <!-- Silmukka -->
                    <?php while (have_posts()) : the_post(); ?>
                    <h2><a href="<?php the_permalink() ?>"><?php the_title(); ?></a></h2>
                    <p><?php echo wp_trim_words( get_the_excerpt(), 30, '...' ); ?></p>
                    <hr />
                    <?php endwhile; ?>
                    <!--Silmukka Päättyy -->

                    <?php else : //Jos tuloksia ei löytynyt ?>
                    <p>Valitettavasti haullasi ei löytynyt mitään.</p>
                    <?php get_search_form(); ?>
                </div>
                <h3>Uusimmat uutiset:</h3>
                <ul>
                    <?php
                    $args = array(
                        'numberposts' => '10',
                        'post_status' => 'publish'
                    );

                    $recent_posts = wp_get_recent_posts( $args );

                    <!--Tuodaan viimeisimmät uutiset -->
                    <foreach( $recent_posts as $recent ) {
                    <echo ' <li><a href="' . get_permalink($recent["ID"]) . "'>' . $re-
                    cent["post_title"] . '</a></li>';
                    }
                    ?>
                </ul>

                <?php endif; ?>
            </div>
        </div>
    </div>
</section>
</hr>
<?php get_footer(); ?>

```