




**Web Research**

# **WORDPRESS CUSTOM THEME DEVELOPMENT**

Verhoeven Wout

Karel de Grote Hogeschool



# Wordpress Custom Theme Development

## Table of Contents

<b>Wordpress Custom Theme Development .....</b>	<b>2</b>
<b>1) Wat is Wordpress? .....</b>	<b>3</b>
<b>2) Wordpress.com vs Wordpress.org? .....</b>	<b>3</b>
<b>3) Wordpress installeren.....</b>	<b>4</b>
<b>4) Wordpress Dashboard .....</b>	<b>4</b>
<b>5) Wordpress Admin Bar.....</b>	<b>5</b>
<b>6) Wordpress Posts en Pages .....</b>	<b>5</b>
<b>7) Wordpress Theme development.....</b>	<b>6</b>
<b>8) Underscores theme files .....</b>	<b>7</b>
<b>9) Back to the basics .....</b>	<b>7</b>
<b>9.1) Start .....</b>	<b>8</b>
<b>9.2) Opdelen in sections .....</b>	<b>8</b>
<b>9.3) The loop.....</b>	<b>10</b>
<b>9.4) Single.php .....</b>	<b>10</b>
<b>9.5) Pagination.....</b>	<b>11</b>
<b>9.6) Comments.....</b>	<b>11</b>
<b>9.7) Functions .....</b>	<b>13</b>
<b>9.7.1) Enqueue Scripts en Stylesheets .....</b>	<b>13</b>
<b>9.7.2) Global Custom Fields creëren .....</b>	<b>14</b>
<b>9.7.3) Post Thumbnail Creëren .....</b>	<b>15</b>
<b>9.7.4) Custom post types.....</b>	<b>15</b>
<b>10) Custom post types: Advanced .....</b>	<b>17</b>
<b>11) Conclusie .....</b>	<b>21</b>

## 1) Wat is Wordpress?

Wordpress is een online, open source tool om makkelijk websites te maken en te beheren. Wordpress is geschreven in PHP en is naar eigen zeggen het meest krachtige Content Management System (CMS) vandaag op de markt.

Wordpress geeft je als gebruiker een administratiepagina waar je alle aspecten van je Wordpress website kan beheren en aanpassen. Via de administratiepagina kan je makkelijk het uiterlijk van je website aanpassen (gebruik makende van 'Themes'), alsook functionaliteiten toevoegen (gebruik makende van 'Plugins').

Meer dan 26% van de websites op het internet maken gebruik van Wordpress. Dit opnieuw omdat het de gebruiker veel vrijheid biedt, alsook het de developer in staat stelt de klant deze vrijheid/mogelijkheid te bieden om nadien aanpassingen te maken.

Aangezien dit een vaak gevraagde vacature/requirement is lijkt het me interessant om me tijdens dit vak te verdiepen in het developen in Wordpress, meerbepaald in het maken van custom 'Themes' en 'Plugins'.

## 2) Wordpress.com vs Wordpress.org?

Als je nieuw bent met Wordpress en aan de slag wilt gaan bestaat er nogal verwarring over waar te starten, je hebt 2 mogelijkheden wordpress.com vs wordpress.org. De vraag die je jezelf moet stellen is 'Wie host de website?'.

Wordpress.org

Je download en installeert Wordpress zelf op je eigen aangekochte domeinnaam.

Wordpress.com

Hier regelt Wordpress dit allemaal voor jou. Je hoeft geen domeinnaam aan te kopen, Wordpress te installeren, of je webserver beheren.

Nadeel:           -Je domeinnaam gaat standard net '.Wordpress.com' eindigen  
                      -Je kan geen custom Themes, Plugins of custom PHP gebruiken op je site.

Echter bestaan hier betalende 'Premium Packages' voor waarbij deze nadelen niet meer van kracht zijn.

### 3) Wordpress installeren

Na het aanschaffen van een domeinnaam kan je Wordpress gaan installeren op je server. Vele hostingbedrijven bieden hier autoinstallers of instructies voor over hoe dit te doen. Echter heeft dit weer een aantal beperkingen, we opteren dus deze manueel te installeren.

Voordelen hierbij zijn:

- Het voorkomt dat Wordpress database tables met default labels worden geïnstalleerd
- Het biedt je de mogelijkheid je 'wp-config.php' file aan te passen (heeft grote voordelen)
- Het stelt je in staat te werken volgens 'best security practices' voor je Wordpress site

(Zorg ervoor dat je bij het kiezen van een domeinnaam zeker een database op je server hebt staan. Wordpress maakt hier namelijk intensief gebruik van en zonder een kan je niet verder.)

Stappen om je wordpress site te installeren:

- Download Wordpress via <http://www.wordpress.org>
- Unpack de zip file
- Transfer de inhoud van de zip file naar je domein
- Maak een database aan, en geef het een naam
- Surf naar je domein, je zal een installer-wizard zien
- Selecteer je taal
- Voer je eerder gekozen database naam, username en password in  
(Database host staat in je hosting gegevens (vul localhost als je lokaal draait))
- Op de volgende pagina klik 'Run the install'
- Vul je site title in, maak een username/password aan om in de toekomst in te loggen op je Wordpress site

Je Wordpress site is nu geïnstalleerd en je kijkt naar het inlogscherf van je Wordpress site. Vul je zopas gekozen username/password in, bij een geslaagde login zie je het dashboard.

### 4) Wordpress Dashboard

Je website beheren doe je via het Wordpress Dashboard.

Je kan het dashboard oproepen door achter je domeinnaam '/wp-admin' te plaatsen, bijvoorbeeld <http://www.mijnwebsite.be/wp-admin>, waarna je op een inlogscherf terecht komt en daarna op het dashboard.

Op het dashboard zie je verschillende menu's staan.

Het menu links stelt je in staat om pages, posts, media, theme's,... toe te voegen of aan te passen.

Het menu boven stelt je in staat om 'live' aanpassingen te maken aan je website, je hoeft hierdoor dus niet de hele tijd heen en weer (van wp-admin naar je site en terug) te navigeren maar je kan alles live/real-time editen. Dit menu noemt men de 'Admin Bar'.

## 5) Wordpress Admin Bar

Als je je website bezoekt terwijl je als administrator bent ingelogd zal je bovenaan continu een menu zien, dit is de admin bar. De functie van de admin bar is om je makkelijker toegang te geven tot de Wordpress functionaliteiten.

In het menu zie je je site naam, klik je hierop kan je terug naar het dashboard, of vanuit het dashboard terug naar je site. Ook kan je met deze knop op een snelle manier naar themes, widgets en menus navigeren.

Naast je site naam zie je 'customize' staan, deze knop brengt je naar de Wordpress Customizer, hierin kan je basis veranderingen maken aan het uiterlijk van je website.

Verder zie je nog een aantal iconen staan om snelle handelingen uit te voeren zoals bvb. een post toe te voegen.

Je kan de admin bar uitschakelen door in je gebruikersinstellingen de instelling 'Show Toolbar when viewing site' uit te vinken.

## 6) Wordpress Posts en Pages

Een Wordpress Post:

Dit is het 'blog' aspect van je site, het zijn vaak artikels of informatie die op reguliere basis worden toegevoegd/aangepast. Posts worden gesorteerd in omgekeerd chronologische volgorde (laatste post staat eerst), en kunnen bedrukt worden met tags of categorieën.

Een Wordpress Page:

Is gelijkaardig aan een post, enkel heeft deze eerder statische informatie. Zie het al seen 'About' of 'Contact' pagina. Pages worden niet gesorteerd en kunnen niet bedrukt worden met tags of categorieën. Pages kunnen hiërarchie hebben, met parent en child pages.

## 7) Wordpress Theme development

Zoals eerder besproken wordt het uiterlijk van je Wordpress website bepaald aan de hand van 'themes'. Je kan deze gratis of betalend downloaden, maar vaak wordt deze ook custom gemaakt.

Wat je ook kan doen is een bestaand theme aanpassen. Je doet dit best practice volgens een 'child theme'. Hierbij ga je zaken die jij wilt aanpassen van de parent theme overschrijven in de child theme.

Je wilt bijvoorbeeld de style.css file aanpassen van het standaard geïnstalleerde theme 'TwentyFourteen'. Je maakt een nieuwe map aan, zet hierin een file 'style.css', en plaats er deze code in.

```
/*  
Theme name: Child theme  
Theme URI: http://www.verhoevenwout.be  
Description: http://www.verhoevenwout.be  
Author: Wout  
Author URI: http://www.verhoevenwout.be  
Template: twentyfourteen  
Version: 0.1  
*/  
@import url('../twentyfourteen/style.css');
```

Je hebt nu de file 'style.css' van het TwentyFourteen theme overschreven.

Valt het je ook op dat deze code eigenlijk in comment staat? Dit is enorm belangrijk want op deze manier weet Wordpress hoe het theme heet, wie de author is, wat de template is, ..., allemaal zaken die in de beschrijving van dit theme zullen gebruikt worden.

Echter opnieuw ben je op deze manier erg beperkt, wij opteren dus om echt 'van scratch' te beginnen met je nieuwe theme.

Om een Wordpress theme te doen functioneren is een bepaalde mappenstructuur/bepaalde files/bepaalde metadata nodig. In zijn absolute eenvoud heb je 'index.php', 'header.php', 'content.php' en 'footer.php'.

Je zal echter al snel merken dat meer files nodig zijn zoals 'functions.php', 'single.php', 'sidebar.php', ... Om deze niet telkens opnieuw te hoeven schrijven bestaan er 'Wordpress Blank Themes' of 'Starter Themes'. Met deze files heb je de absolute backbone om een wordpress theme te maken zonder dat deze je eigen code in de weg loopt.

Een erg populair starter theme is 'Underscores', je kan deze downloaden op 'underscores.me'. Na het ingeven van wat basis informatie (author, url, etc), krijg je een zip file, die je vervolgens uitpakt en in de map 'wp-content/themes' plaatst. Als je nu op je dashboard onder het tabblad 'themes' gaat kijken zie je je nieuwe underscores theme geïnstalleerd.

Om je theme een thumbnail te geven zoals de andere themes verander je simpelweg de file 'screenshot.png' in de map van je nieuwe theme.

## 8) Underscores theme files

Wordpress werkt zeer modulair, het heeft verschillende files voor verschillende pagina's of verschillende delen van een pagina. Als eerste wordt 'index.php' opgeladen. Je zal zien dat hierin wordpress functions staan als 'get\_header()', 'get\_footer()', 'get\_sidebar()', deze eerste functie roept dus hoogst waarschijnlijk de file 'header.php' aan en voegt deze toe aan de file.

In het midden zie je code staan die gaan over 'the loop'. Dit is de core functionaliteit van wordpress. Het is als het ware een 'while-loop' die data uit het backend-gedeelte van wordpress haalt.

Kijken we echter naar een post, dan wordt deze opgeroepen door de file 'single.php'. Deze file lijkt sterk op de 'index.php' file, echter heeft deze ook een stuk code die de comments-sectie inlaadt. Kijken we dan weer naar een pagina (statische content), wordt deze opgeroepen door de file 'page.php'. Je ziet dus dat iedere file een bepaald doel heeft.

## 9) Back to the basics

Ik ondervond in het begin zelf vele problemen met het werken via underscores, ook het implementeren van 'Bootstrap' en andere zaken bleek een hele opgave te zijn. Daarom back to the basics.

Maak een mapje aan in de 'themes' folder van je wordpress project. Plaats hier 2 files; 'index.php' en style.css. plaats in de css file volgende code:

```
/*  
Theme Name: MijnThema  
Author: Wout Verhoeven  
Description: Mijn nieuw Wordpress Thema met Bootstrap  
Version: 0.0.1  
Tags: bootstrap  
*/
```

Deze code is nodig om je theme te doen verschijnen in het wordpress dashboard (onder themes uiteraard).

Ziezo, je hebt je eigen theme gemaakt! Uiteraard is dit zeer basic, en nog helemaal niet dynamisch.

## 9.1) Start

We gaan van start met een basic staticwebsite gemaakt in Bootstrap, de bedoeling is deze nu dynamisch aanpasbaar te maken via wordpress.

In index.html van onze staticwebsite staat een link naar een css file.

```
<link href="style.css" rel="stylesheet">
```

Je zal merken dat dit in ons wordpress project niet zal werken, dit omdat de file fout wordt gelokaliseerd. Ipv naar 'blog.css' te zoeken moet de file eigenlijk worden gezocht in 'wp-content/themes/MijnThema/'.

We lossen dit probleem op door die code te vervangen door:

```
<link href="<?php bloginfo('template_directory');?>/style.css" rel="stylesheet">
```

Hetzelfde concept geldt voor images, javascript, en meeste van de andere files die zich in je themes folder bevinden, buiten php-files.

## 9.2) Opdelen in sections

Momenteel staat alles in je 'index.php' file, maar uiteraard willen we een header, footer en sidebar op iedere pagina. In wordpress delen we daarom alle delen op in aparte files namelijk: 'header.php', 'footer.php', 'sidebar.php' en 'content.php'.

### Header.php static

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5 <meta charset="utf-8">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <meta name="description" content="">
9 <meta name="author" content="">
10
11 <title>Blog Template for Bootstrap</title>
12 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
13 <link href="<?php bloginfo('template_directory');?>/blog.css" rel="stylesheet">
14 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
15 <!--[if lt IE 9]>
16 <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
17 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
18 <![endif]>
19 </head>
20
21 <body>
22 <div class="blog-masthead">
23 <div class="container">
24 <nav class="blog-nav">
25 <a class="blog-nav-item active" href="#">Home</a>
26 <a class="blog-nav-item" href="#">New features</a>
27 <a class="blog-nav-item" href="#">Press</a>
28 <a class="blog-nav-item" href="#">New hires</a>
29 <a class="blog-nav-item" href="#">About</a>
30 </nav>
31 </div>
32 </div>
33
34 <div class="container">
35
36 <div class="blog-header">
37 <h1 class="blog-title"><a href="<?php bloginfo('wpurl');?>"><?php echo get_bloginfo('name'); ?></a></h1>
38 <p class="lead blog-description"><?php echo get_bloginfo('description'); ?></p>
39 </div>
40
```

### Header.php dynamic

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5 <meta charset="utf-8">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <meta name="description" content="">
9 <meta name="author" content="">
10
11 <title>Blog Template for Bootstrap</title>
12 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
13 <link href="<?php bloginfo('template_directory');?>/blog.css" rel="stylesheet">
14 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
15 <!--[if lt IE 9]>
16 <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
17 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
18 <![endif]>
19 <?php wp_head();?>
20 </head>
21
22 <body>
23
24 <div class="blog-masthead">
25 <div class="container">
26 <nav class="blog-nav">
27 <a class="blog-nav-item active" href="#">Home</a>
28 <!-- 'title_li' to show as list -->
29 <?php wp_list_pages('title_li='); ?>
30 </nav>
31 </div>
32 </div>
33
34 <div class="container">
35
36 <div class="blog-header">
37 <h1 class="blog-title"><a href="<?php bloginfo('wpurl');?>"><?php echo get_bloginfo('name'); ?></a></h1>
38 <p class="lead blog-description"><?php echo get_bloginfo('description'); ?></p>
39 </div>
40
```

Zoals je kan zien maken we al meteen een aantal aanpassingen aan onze header.php file.

Zo heb ik als eerste '<?php wp\_head();?>' toegevoegd, dit wordt geplaatst om achteraf bepaalde zaken die nog in de header moeten komen te appenden.

Ook heb ik alle <li> verwijderd en vervangen door <?php wp\_list\_pages();?>, dit toont zoals je al aanvoelt alle pages die zich in je wordpress site bevinden. ('&title\_li=' toont de data in een <li> vorm). Hieronder zie je de code van de andere files.

Ook is de blog-header <div> dynamisch content gaan opvragen.



## Footer.php

```
1 </div> <!-- /.container -->
2
3 <footer class="blog-footer">
4 <p>Blog template built for <a href="http://getbootstrap.com">Bootstrap</a> by <a href="https://twitter.com/mdo">
  @mdo</a>.</p>
5
6 <p>
7   <a href="#">Back to top</a>
8 </p>
9 </footer>
10
11 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
12 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
13 <?php wp_footer(); ?>
14 </body>
15 </html>
```

## Index.php

```
1 <?php get_header(); ?>
2
3 <div class="row">
4
5   <div class="col-sm-8 blog-main">
6
7     <?php
8     if ( have_posts() ) : while ( have_posts() ) : the_post();
9
10       get_template_part( 'content', get_post_format() );
11
12     endwhile; endif;
13   <?>
14
15   </div> <!-- /.blog-main -->
16
17   <?php get_sidebar(); ?>
18
19 </div> <!-- /.row -->
20
21 <?php get_footer(); ?>
```

Let op hoe hier terug de verschillende delen van de website worden opgeroepen.

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

De while lus die je ziet komt later bod.

## Page.php

```
1 <?php get_header(); ?>
2
3 <div class="row">
4   <div class="col-sm-12">
5
6     <?php
7     if ( have_posts() ) : while ( have_posts() ) : the_post();
8
9       get_template_part( 'content', get_post_format() );
10
11     endwhile; endif;
12   <?>
13
14   </div> <!-- /.col -->
15 </div> <!-- /.row -->
16
17 <?php get_footer(); ?>
```

Page.php wordt gebruikt als je op een page klikt. In dit geval is deze zeer gelijk aan index.php, met het enige verschil dat de sidebar niet getoond zal worden.

## Content.php

```
<div class="blog-post">
  <h2 class="blog-post-title"><?php the_title(); ?></h2>
  <p class="blog-post-meta"><?php the_date(); ?> by <a href="#"><?php the_author(); ?></a>
  <?php the_content(); ?>
</div><!-- /.blog-post -->
```

Je ziet hier ook weer dat er op een dynamische manier de title, datum en content worden opgevraagd van een post. Je kan trouwens 'the\_content()' ook veranderen door 'the\_excerpt()', hierdoor toon je enkel de eerste 55 lijnen i.p.v. de gehele post.

## 9.3) The loop

‘The loop’ zagen we al eerder, het is een while loop die gegevens uit de backend gaat halen, het is de meest belangrijke functie in Wordpress.

Op zich is de loop vrij eenvoudig: als er posts zijn, terwijl er posts zijn, toon de post. Alles binnen de loop wordt herhaald.

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

<!-- contents of the loop -->

<?php endwhile; endif; ?>
```

We zagen deze loop eerder in index.php, zoals je kan zien staat er in de loop een functie die de content van die post gaat oproepen, deze file dient nog aangemaakt te worden anders wordt er niets getoont.

```
1 <?php get_header(); ?>
2
3 <div class="row">
4
5   <div class="col-sm-8 blog-main">
6
7     <?php
8       if ( have_posts() ) : while ( have_posts() ) : the_post();
9
10        get_template_part( 'content', get_post_format() );
11
12      endwhile; endif;
13    ?>
14
15  </div> <!-- /.blog-main -->
16
17  <?php get_sidebar(); ?>
18
19 </div> <!-- /.row -->
20
21 <?php get_footer(); ?>
```

Heb je al deze stappen doorlopen, dan zou je nu een werkende wordpress website met custom theme moeten hebben.

## 9.4) Single.php

We gaan een ‘single.php’ file aanmaken, zijn doel is om een individuele post te tonen. Het is dus eigenlijk een duplicaat van ‘page.php’, enkel veranderen we ‘content’ naar ‘content-single’.

We moeten dus een file ‘content-single.php’ creëren, dit is een duplicaat van content.php.

Je kan nu zien dat ‘index.php’ ‘content.php’ gaat ophalen, en ‘content.php’ ‘content-single.php’ gaat ophalen. Door aan de <h2> tag van de ‘content.php’ file een permalink toe te voegen, linken we beide files aan elkaar (bij klik op een post binnen content, ga naar content-single).

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

<div class="row">
  <div class="col-sm-12">

    <?php
      if ( have_posts() ) : while ( have_posts() ) : the_post();
        get_template_part( 'content-single', get_post_format() );
      endwhile; endif;
    ?>

  </div> <!-- /.col -->
</div> <!-- /.row -->

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

In ‘content.php’

```
<a href="<?php the_permalink(); ?>">
```

## 9.5) Pagination

Wil je op je blog verschillende pagina's hebben i.p.v. een gigantisch lange waslijst van posts? Dan kan dit via 'Pagination'. Dit is een stukje code dat ervoor zorgt dat bijvoorbeeld onderaan je 'index.php' file 2 buttons staan die je door de verschillende pages van posts helpen navigeren.

In je 'index.php' file moet deze code komen net na de loop, maar binnen de 'if have\_posts()' statement, aangezien we dit enkel willen tonen als er posts zijn. Vervolgens kan je binnen het dashboard via settings > Reading het aantal getoonde posts per pagina wijzigen via 'Blog pages show at most'. Standaard staat deze op 10. In het voorbeeld hieronder hebben we de paging code in een kleine navigatie gezet zodat deze er snel mooi uit ziet volgens bootstrap.

```

1  <?php get_header(); ?>
2  <div class="row">
3    <div class="col-sm-8 blog-main">
4
5      <?php
6        if ( have_posts() ) : while ( have_posts() ) : the_post();
7          get_template_part( 'content', get_post_format() );
8        endwhile; ?>
9
10       <nav>
11         <ul class="pager">
12           <li><?php next_posts_link( 'Previous' ); ?></li>
13           <li><?php previous_posts_link( 'Next' ); ?></li>
14         </ul>
15       </nav>
16     <?php endif; ?>
17   </div> <!-- /.blog-main -->
18   <?php get_sidebar(); ?>
19 </div> <!-- /.row -->
20
21 <?php get_footer(); ?>

```

Nieuwe 'index.php' file

## 9.6) Comments

Een van de grootste voordelen van server-based content-management-systems en dus ook wordpress is om comments aan te bieden zonder gebruik te maken van een third-party.

We gaan in 'single.php' een comments-sectie aanmaken.

Huidige code 'single.php':

```

if ( have_posts() ) : while ( have_posts() ) : the_post();
  get_template_part( 'content-single', get_post_format() );
endwhile; endif;

```

Met comments-sectie, deze zegt, als de comments enabled zijn, toon dan de comments template

```

if ( have_posts() ) : while ( have_posts() ) : the_post();
  get_template_part( 'content-single', get_post_format() );

  if ( comments_open() || get_comments_number() ) :
    comments_template();
  endif;
endwhile; endif;

```

Verder maken we een nieuwe file genaamd 'comments.php' (standaard comments template).

```

1  <?php if ( post_password_required() ) {
2      return;
3  } ?>
4  <div id="comments" class="comments-area">
5      <?php if ( have_comments() ) : ?>
6          <h3 class="comments-title">
7              <?php
8                  printf( _nx( 'One comment on "%2$s"', '%1$s comments on "%2$s"', get_comments_number(), 'comments title' ),
9                      number_format_i18n( get_comments_number() ), get_the_title() );
10             ?>
11          </h3>
12          <ul class="comment-list">
13              <?php
14                  wp_list_comments( array(
15                      'short_ping' => true,
16                      'avatar_size' => 50,
17                  ) );
18              ?>
19          </ul>
20      <?php endif; ?>
21      <?php if ( ! comments_open() && get_comments_number() && post_type_supports( get_post_type(), 'comments' ) ) :
22          ?>
23          <p class="no-comments">
24              <?php _e( 'Comments are closed.' ); ?>
25          </p>
26      <?php endif; ?>
27      <?php comment_form(); ?>
28  </div>

```

Eerst en vooral creëren we een functionaliteit om te voorkomen dat mensen comments kunnen plaatsen als je je settings hebt ingesteld op 'password protected comments', hierna creëren we een comments div die de comments zal tonen mochten deze aanwezig zijn.

Als we de code volgen wordt in de comment-sectie het aantal comments getoont (get\_comments\_number()), waarna deze ook in lijst vorm worden weergegeven (wp\_list\_comments()). Als de comments gesloten zijn (! comments\_open()) wordt je dit getoont, en op het einde vinden we nog een form om een nieuwe comment te submitten.

Wat betreft het stukje '\_nx( 'One comment on "%2\$s"', '%1\$s comments on "%2\$s"', get\_comments\_number(), 'comments title')', hierbij toont men de eerste parameter als er slechts 1 comment is en de tweede parameter wanneer er meerdere comments zijn. '%1\$s' en '%2\$s' zijn plaatsen waar de strings van de derde en vierde parameter zullen worden geplaatst (in dit geval het nummer, en de titel van de post). Ik heb hetzelfde stukje code ook in de file 'content.php' geplaatst zodat onder iedere posts ook het aantal comments geschreven staat.

## 9.7) Functions

'Functions.php' wordt gebruikt om functionaliteiten aan je site toe te voegen. Plugins en custom functions zijn praktisch hetzelfde trouwens, alle code die je schrijft in deze 'functions.php' file kan gebruikt worden als plugin. Het enige verschil is dat de code in je 'functions.php' file enkel gaat werken als dat theme actief is.

In principe komt een codechunk in de 'functions.php' file telkens neer op dit.

```
function custom_function() {  
    //code  
}  
add_action( 'action', 'custom_function');
```

'Add\_action()' noemt men een 'hook', het is als het ware een eventListener als in javascript dat bovenstaande code zal uitvoeren bij een bepaalde actie (bvb. bij opstart, of bij een post, ...).

### 9.7.1) Enqueue Scripts en Stylesheets

We zagen eerder hoe we scripts en stylesheets toevoegden aan ons wordpress project. Dit door simpelweg een verwijzing in de header, en op het einde van de bodytag bepaalde verwijzingen naar de files te plaatsen. Echter is dit eigenlijk een 'bad-practice', we doen dit beter via de 'functions.php' file.

We maken eerst een nieuwe, gestructureerdere mappenstructuur aan met simpelweg een mapje 'css' en 'js', met hierin onze css/js files (niet 'style.css' en 'functions.js'), alsook 'bootstrap.min.css' en 'bootstrap.min.js'. We voegen daarna deze code toe aan 'functions.php'.

```
1 <?php  
2  
3 // Add scripts and stylesheets  
4 function startwordpress_scripts() {  
5     wp_enqueue_style( 'bootstrap', get_template_directory_uri() . '/css/bootstrap.min.css', array(), '3.3.6' );  
6     wp_enqueue_style( 'blog', get_template_directory_uri() . '/css/blog.css' );  
7     wp_enqueue_script( 'bootstrap', get_template_directory_uri() . '/js/bootstrap.min.js', array('jquery'), '3.3.6',  
8         true );  
9 }  
10 add_action( 'wp_enqueue_scripts', 'startwordpress_scripts' );
```

Wat dit gaat doen is opnieuw de css/js files ophalen, deze keer op de 'best-practice' manier.

wp\_enqueue\_style gaat zoals je zou vermoeden stylesheets ophalen, hetzelfde geldt voor wp\_enqueue\_scripts. Als parameters worden toegevoegd: een id, locatie van de file, een toekomstige array met dependencies en version number.

We gaan nu hetzelfde doen voor 'Google fonts' door via enqueue een font toe te voegen.

```
12 // Add Google Fonts  
13 function startwordpress_google_fonts() {  
14     wp_register_style( 'OpenSans', 'http://fonts.googleapis.com/css?family=Open+Sans:400,600,700,800' );  
15     wp_enqueue_style( 'OpenSans' );  
16 }  
17  
18 add_action( 'wp_print_styles', 'startwordpress_google_fonts' );
```

Deze code is wat anders als de vorige functie, dit omdat we nu met dynamische url's zitten. We moeten deze eerst registreren, en dan enqueue'en.

Bijgevolg kunnen we nu in onze header en index file de dependencies verwijderen.

## 9.7.2) Global Custom Fields creëren

Soms heb je misschien bepaalde custom settings die je graag globaal wil definiëren, aan de hand van het dashboard. Een voorbeeld hiervan is onze sidebar met links naar twitter/github/...

Eerst maken we een sectie aan in het dashboard panel, zet deze code in 'functions.php'.

```
24 // Custom settings
25 function custom_settings_add_menu() {
26     add_menu_page( 'Custom Settings', 'Custom Settings', 'manage_options', 'custom-settings', 'custom_settings_page',
27         null, 99);
28 }
29 add_action( 'admin_menu', 'custom_settings_add_menu' );
```

De verschillende parameters van deze functie zijn: (pageTitle, menuTitle, capability, menuSlug, callable function, iconUrl, position).

We hebben nu op ons dashboard een extra tabblad genaamd 'Custom Settings', echter staat hier nog niets in. Om hiervoor te zorgen voegen we volgende code toe.

```
32 // Create Custom Global Settings
33 function custom_settings_page() { ?>
34     <div class="wrap">
35         <h1>Custom Settings</h1>
36         <form method="post" action="options.php">
37             <?php
38                 settings_fields('section');
39                 do_settings_sections('theme-options');
40                 submit_button();
41             ?>
42         </form>
43     </div>
44 <?php }
45
46 // Twitter
47 function setting_twitter() { ?>
48     <input type="text" name="twitter" id="twitter" value="<?php echo get_option('twitter'); ?>" />
49 <?php }
50
51 //Set up the page to show, accept and save the option fields
52 function custom_settings_page_setup() {
53     add_settings_section('section', 'All Settings', null, 'theme-options');
54     add_settings_field('twitter', 'Twitter URL', 'setting_twitter', 'theme-options', 'section');
55
56     register_setting('section', 'twitter');
57 }
58 add_action( 'admin_init', 'custom_settings_page_setup' );
```

Wat hier allemaal gebeurt is:

- 33. Maak een stukje html met daarin een header, functies die fields, secties en een submit button ophalen.
- 47. Creëer een inputfield waarbij je de value gaat ophalen bij add\_settings\_field.
- 53. Creëer een settingsSection genaamd 'All Settings', steek hierin de theme-options.
- 54. Creëer een settingsField, met hierin het inputveld van vorige functie.
- 58. Bij initialisatie van de admin, voer bovenstaande functie uit.

Nu veranderen we enkel nog in de file 'sidebar.php' de links nog naar 'echo get\_option('github');', en nu zijn je url's dynamisch aanpasbaar gemaakt via het 'Custom Settings' menu.

```
14 <ol class="list-unstyled">
15     <li><a href="<?php echo get_option('github'); ?>">GitHub</a></li>
16     <li><a href="<?php echo get_option('twitter'); ?>">Twitter</a></li>
17 </ol>
```

## 9.7.3) Post Thumbnail Creëren

Wil je gebruik maken van thumbnails op je pagina 'content.php', dan kan dit door in je 'functions.php' file dit stukje code te plaatsen.

```
// Support Featured Images
add_theme_support( 'post-thumbnails' );
```

Je hebt nu normaal in je dashboard de mogelijkheid om binnen een post een 'featured image' te setten, dit is je thumbnail.

Vervolgens passen we onze file 'content.php' aan zodat deze de thumbnail toont.

```
<?php if ( has_post_thumbnail() ) {?>
<div class="row">
  <div class="col-md-4">
    <?php the_post_thumbnail('thumbnail'); ?>
  </div>
  <div class="col-md-6">
    <?php the_excerpt(); ?>
  </div>
</div>
<?php } else { ?>
<?php the_excerpt(); ?>
<?php } ?>
```

Onthoud dat 'the\_excerpt()' slechts 55 karakters van onze post toont, exclusief eventuele foto's in onze post.

## 9.7.4) Custom post types

Wordpress biedt je de mogelijkheid zelf een 'custom post' te maken als de standaard wordpress post opmaak niet van toepassing is op jouw design.

We gaan een simple custom post maken die

Eerst en vooral, voeg deze code toe aan 'functions.php'

```
// Custom Post Type
function create_my_custom_post() {
    register_post_type('my-custom-post',
        array(
            'labels' => array(
                'name' => __('My Custom Post'),
                'singular_name' => __('My Custom Post'),
            ),
            'public' => true,
            'has_archive' => true,
            'supports' => array(
                'title',
                'editor',
                'thumbnail',
                'custom-fields'
            )
        )
    );
}
add_action('init', 'create_my_custom_post');
```

In deze functie creëren we een post genaamd 'My Custom Post', met een slug 'my-custom-post', dit maakt dat als mijn url 'example.be' is, de custom post url 'example.be/my-custom-post' zou zijn.

In 'supports' kan je zien welke functionaliteiten we allemaal toevoegen, namelijk:

- Title, <?php the\_title();?>
- Editor <?php the\_content();?>
- Thumbnail <?php the\_post\_thumbnail();?>
- Custom-fields Fields die je later kan adden en callen.

### Web Research 3

Voor de makkelijkheid maken we een nieuwe page aan via het dashboard en noemen die 'Custom'. Momenteel is deze page echter nog gegevens aan het pullen van 'page.php'.

We maken daarom een nieuwe file aan genaamd 'page-custom.php', en kopiëren hier de code van 'page.php'. Volgens de 'Wordpress Template Hiërarchy' gaat een 'page-name.php' de 'page.php' overschrijven. Zorg er zeker voor dat de 'name' hetzelfde is als je page, anders roepen deze elkaar niet op.

Het enige dat we veranderen is dat deze enkel posts toont van het type 'my-custom-post', we doen dit op deze manier.

```
6      <?php
7          $args = array(
8              'post_type' => 'my-custom-post',
9              'orderby' => 'menu_order',
10             'order' => 'ASC'
11         );
12         $custom_query = new WP_Query( $args );
13         while ($custom_query->have_posts()) : $custom_query->the_post(); ?>
14
15             <div class="blog-post">
16                 <h2 class="blog-post-title"><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
17                 <?php the_excerpt(); ?>
18             </div>
19
20         <?php endwhile; ?>
```

Een custom post loop lijkt een beetje anders dan de normale loop, namelijk deze kan argumenten, of condities aannemen waardoor er gefilterd kan worden. Deze filter bevindt zich in de '\$custom\_query' variabele, de filter argumenten/condities bevinden zich in '\$args'.

Kijken we nu naar de 'custom' pagina, zal je zien dat er enkel posts worden getoont, aangemaakt in onze eigen geschreven 'My Custom Post' sectie.



## 10) Custom post types: Advanced

In dit deel gaan we custom fields toevoegen aan onze posts, ze toevoegen aan de database, en tonen op de frontend van je website, dit zonder gebruik te maken van een plugin.

Eerst creëren we een custom post genaamd 'Your Post', we plaatsen deze code in 'functions.php'.

```

1 <?php
2 function create_post_your_post() {
3     register_post_type( 'your_post',
4         array(
5             'labels' => array(
6                 'name' => __( 'Your Post' ),
7             ),
8             'public' => true,
9             'hierarchical' => true,
10            'has_archive' => true,
11            'supports' => array(
12                'title',
13                'editor',
14                'excerpt',
15                'thumbnail',
16            ),
17            'taxonomies' => array(
18                'post_tag',
19                'category',
20            )
21        )
22    );
23    register_taxonomy_for_object_type( 'category', 'your_post' );
24    register_taxonomy_for_object_type( 'post_tag', 'your_post' );
25 }
26 add_action( 'init', 'create_post_your_post' );
27

```

Eerst registreren we een post type genaamd 'Your Post' met id 'your\_post'. We geven mee dat deze een title, editor, excerpt en thumbnail moet supporten. Ook support deze post 'taxonomies', dit is een manier om posts te groeperen, in dit geval volgens tag en category.

Als we een thumbnail willen gebruiken moeten we nog een extra lijntje code schrijven namelijk: 'add\_theme\_support( 'post-thumbnails' );' Hierbij enable je thumbnails voor je theme.

Vervolgens gaan we ervoor zorgen dat we deze post ergens kunnen tonen, in dit geval zet ik volgende code in 'page.php'.

```

1 <?php
2
3 $args = array(
4     'post_type' => 'your_post',
5 );
6 $your_loop = new WP_Query( $args );
7
8 if ( $your_loop->have_posts() ) : while ( $your_loop->have_posts() ) : $your_loop->the_post();
9     $meta = get_post_meta( $post->ID, 'your_fields', true ); ?>
10
11 <!-- contents of Your Post -->
12
13 <?php endwhile; endif; wp_reset_postdata(); ?>

```

De '\$your\_loop' variabele zorgt ervoor dat enkel de posts worden getoont die door de nieuwe custom post functie zijn opgemaakt. Wat betreft '\$meta', dit is momenteel nog niet relevant, we komen hier later op terug.

In 'page.php' voegen we nog 2 lijntjes code toe (onder contents of your post) zodat we onze post kunnen zien. <?php the\_title();?> en <?php the\_content();?>

Nu we onze custom content kunnen tonen op een page, gaan we 'meta-boxes' aanmaken. Een meta-box wordt gebruikt om custom zaken aan je post toe te voegen zoals een kalender, foto, ...

In 'functions.php' voegen we volgende code toe:

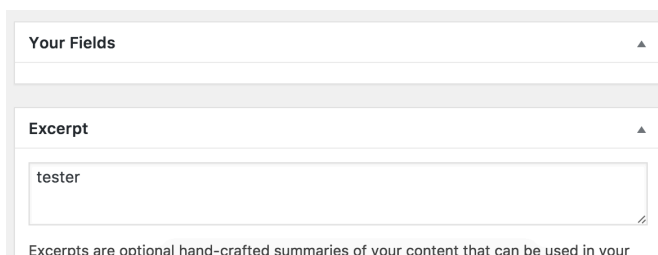
```
29 function add_your_fields_meta_box() {  
30     add_meta_box(  
31         'your_fields_meta_box', // $id  
32         'Your Fields', // $title  
33         'show_your_fields_meta_box', // $callback  
34         'your_post', // $screen  
35         'normal', // $context  
36         'high' // $priority  
37     );  
38 }  
39 add_action( 'add_meta_boxes', 'add_your_fields_meta_box' );
```

De belangrijkste parameters hier zijn:

- '\$title', wat de naam van je metabox bepaald
- '\$screen', wat de locatie van de metabox gaat zijn (waar wordt deze getoont)

Je kan deze metabox aan iedere post toevoegen, echter in dit geval linken we deze aan 'your\_post', de custom post die we eerder maakten.

Ga je nu terug naar je post, dan zie je een extra veld 'Your Fields' staan. Echter is deze momenteel nog leeg en bevat deze dus nog geen functie, het is een lege meta-box.



The screenshot shows a WordPress post editor interface. At the top, there is a new meta-box titled 'Your Fields' which is currently empty. Below it is the 'Excerpt' meta-box, which contains the text 'tester'. At the bottom of the excerpt box, there is a small text that reads: 'Excerpts are optional hand-crafted summaries of your content that can be used in your'.

Vervolgens gaan we ervoor zorgen dat deze fields in de database worden geschreven.

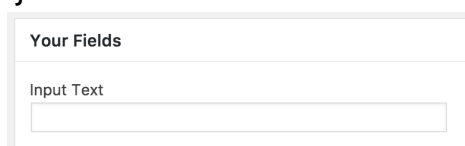
Eerst schrijven we de code om al je custom fields te tonen, we hebben als voorbeeld een textinput in de meta-box geplaatst.

```
41 function show_your_fields_meta_box() {  
42     global $post;  
43     $meta = get_post_meta( $post->ID, 'your_fields', true );  
44  
45     <input type="hidden" name="your_meta_box_nonce" value="<?php echo wp_create_nonce( basename( __FILE__ ) ); ?>">  
46  
47     <!-- All fields will go here -->  
48     <p>  
49         <label for="your_fields[text]">Input Text</label>  
50         <br>  
51         <input type="text" name="your_fields[text]" id="your_fields[text]" class="regular-text" value="<?php echo $meta  
52             ['text']; ?>">  
53     </p>  
54 }
```

Let erop dat de class 'regular-text' gewoon een build-in Wordpress admin style is.

Wat betreft de '[' in 'your\_fields[text]', dit specificeert de inhoud van de input. Deze kan/mag je veranderen. Mocht dit nu een emailadres zijn kan je het veranderen naar bijvoorbeeld '[email]'.

Je meta-box ziet er nu zo uit.



The screenshot shows the 'Your Fields' meta-box in the WordPress post editor. It contains a label 'Input Text' and an empty text input field below it.

Echter kan deze nu nog niets opslaan in de database. Onderstaande code zorgt hiervoor.

Code om je meta-box fields op te slaan in de database:

```

55 function save_your_fields_meta( $post_id ) {
56     // verify nonce
57     if ( !wp_verify_nonce( $_POST['your_meta_box_nonce'], basename(__FILE__) ) ) {
58         return $post_id;
59     }
60     // check autosave
61     if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE ) {
62         return $post_id;
63     }
64     // check permissions
65     if ( 'page' === $_POST['post_type'] ) {
66         if ( !current_user_can( 'edit_page', $post_id ) ) {
67             return $post_id;
68         } elseif ( !current_user_can( 'edit_post', $post_id ) ) {
69             return $post_id;
70         }
71     }
72
73     $old = get_post_meta( $post_id, 'your_fields', true );
74     $new = $_POST['your_fields'];
75
76     if ( $new && $new !== $old ) {
77         update_post_meta( $post_id, 'your_fields', $new );
78     } elseif ( '' === $new && $old ) {
79         delete_post_meta( $post_id, 'your_fields', $old );
80     }
81 }
82 add_action( 'save_post', 'save_your_fields_meta' );

```

Let er goed op dat 'your\_meta\_box\_nonce' overeen komt met de vorige functie's 'your\_meta\_box\_nonce'. Deze functie bekijkt ook of de gebruiker de juiste rechten heeft om fields te updaten.

Vervolgens gaan we meerdere fields toevoegen aan de metabox (er staat al 1 input field in).

We gaan toevoegen: input field, textbox, checkbox, select menu en een image upload.

De eerste 4 inputfields worden verkregen met volgende code:

```

47 <!-- All fields will go here -->
48 <p>
49     <label for="your_fields[text]">Input Text</label>
50     <br>
51     <input type="text" name="your_fields[text]" id="your_fields[text]" class="regular-text" value="<?php echo $
52         meta['text']; ?>">
53 </p>
54 <p>
55     <label for="your_fields[textarea]">Textarea</label>
56     <br>
57     <textarea name="your_fields[textarea]" id="your_fields[textarea]" rows="5" cols="30" style="width:500px;">
58         <?php echo $meta['textarea']; ?></textarea>
59 </p>
60 <p>
61     <label for="your_fields[checkbox]">Checkbox
62     <input type="checkbox" name="your_fields[checkbox]" value="checkbox" <?php if ( $meta['checkbox'] === '
63         checkbox' ) echo 'checked'; ?>
64     </label>
65 </p>
66 <p>
67     <label for="your_fields[select]">Select Menu</label>
68     <br>
69     <select name="your_fields[select]" id="your_fields[select]">
70         <option value="option-one" <?php selected( $meta['select'], 'option-one' ); ?>>Option One</option>
71         <option value="option-two" <?php selected( $meta['select'], 'option-two' ); ?>>Option Two</option>
72     </select>
73 </p>

```

Wat betreft de image upload moet je goed letten dat je de juiste classes aan de verschillende inputs geeft, 'meta-image', 'regular-text' en 'image-upload'.

```

71 <p>
72 <label for="your_fields[image]">Image Upload</label><br>
73 <input type="text" name="your_fields[image]" id="your_fields[image]" class="meta-image regular-text" value="
74 <?php echo $meta['image']; ?>">
75 <input type="button" class="button image-upload" value="Browse">
76 </p>
77 <div class="image-preview"></div>

```

Echter gaat deze snippet niet veel doen, door onderstaand javascript snippet van Foundation toe te voegen, opent er de built in Wordpress Media Gallery, die zorgt dat je kan browsen naar images. Je plaatst deze code recht onder de 'image-preview' div.

```

77 <script>
78 jQuery(document).ready(function ($) {
79
80     // Instantiates the variable that holds the media library frame.
81     var meta_image_frame;
82     // Runs when the image button is clicked.
83     $('.image-upload').click(function (e) {
84         e.preventDefault();
85         var meta_image = $(this).parent().children('.meta-image');
86
87         // If the frame already exists, re-open it.
88         if (meta_image_frame) {
89             meta_image_frame.open();
90             return;
91         }
92         // Sets up the media library frame
93         meta_image_frame = wp.media.frames.meta_image_frame = wp.media({
94             title: meta_image.title,
95             button: {
96                 text: meta_image.button
97             }
98         });
99         // Runs when an image is selected.
100         meta_image_frame.on('select', function () {
101             // Grabs the attachment selection and creates a JSON representation of the model.
102             var media_attachment = meta_image_frame.state().get('selection').first().toJSON();
103             // Sends the attachment URL to our custom image input field.
104             meta_image.val(media_attachment.url);
105         });
106         // Opens the media library frame.
107         meta_image_frame.open();
108     });
109 });
110 </script>

```

Ik ga hier niet verder op in omdat dit een vaak gebruikte Wordpress snippet is.

'Functions.php' is nu klaar.

```

11 <!-- contents of Your Post -->
12 <h1>Title</h1>
13 <?php the_title(); ?>
14
15 <h1>Content</h1>
16 <?php the_content(); ?>
17
18 <h1>Text Input</h1>
19 <?php echo $meta['text']; ?>
20
21 <h1>Textarea</h1>
22 <?php echo $meta['textarea']; ?>
23
24 <h1>Checkbox</h1>
25 <?php if ( $meta['checkbox'] == 'checkbox' ) { ?>
26     Checkbox is checked.
27 <?php } else { ?>
28     Checkbox is not checked.
29 <?php } ?>
30
31 <h1>Select Menu</h1>
32 <p>The actual value selected.</p>
33 <?php echo $meta['select']; ?>
34
35 <h1>Image</h1>
36 <img src="<?php echo $meta['image']; ?>"
37

```

Je kan zien dat we deze fields oproepen door gebruik te maken van \$meta["].

Hierbij is het belangrijk dat datgene binnen de [], overeenkomt met de benamingen wat in je 'functions.php' file staat, meerbepaald in de functie: 'show\_your\_fields\_meta\_box'.

De nieuwe custom posts worden nu getoont op een page.

## **11) Conclusie**

Wordpress is een sterk wapen als je een solid cms system wil gebruiken voor je website. Echter vind ik de leercurve om eigen theme's te maken groot, alsook is het lastig terug helemaal van nul te hoeven beginnen bij het leren developen van een Wordpress website (nieuwe syntax, Wordpress functions, action triggers, ...).

Al bij al ben ik wel blij mij hierop te hebben verdiept, aangezien dit een vaak gevraagde skill is bij front-end developers.