What does it take to develop a Wordpress site.

I had a very interesting conversation at my coffee shop the other day. It revolved around Wordpress and how easy it is (or isn't) to modify and customize. In the 10+ years I've been using it things have gotte to the point where you don't really need to know PHP as much as you used to.

In this post we'll cover the following aspects of customizing a Wordpress theme and associated tools and functionality:

- CSS customizations
- Local changes using filters and actions
- Creating child themes
- Customizing templates
- Using Wordpress REST API to create a completely different front end
- When it's best to create a theme from scratch

We'll also talk about the minimal skillset necessary for creating a Wordpress theme with the understanding that it will depend on the changes you want to make.

My personal biases against Wordpress business model

I make no secret of the issues I have against Wordpress and their business model. In GPL and why I don't develop for WordPress I've documented my opposition against the heavy handed approach Matt Mullenweg and Automattic took against an independent theme developer.

In short, because I release all my code under MIT, I would not be able to offer my code, plugins or themes in Wordpress.org. They will not host code released under licenses other than GPL even if they are more permissive.

I am philosophically opposed to GPL/LGPL and Affero general public licenses. I should be the one deciding the license I release my code under, not the Free Software Foundation or Automattic.

That said, *not wanting* to develop for Wordpress doesn't equal *not knowing* how to develop for Wordpress. Sharing this post doesn't equal endorsing their policies.

CSS cutomizations

The easiest way to change the looks of a Wordpress app is to change the theme's CSS. Dev Tools comes in very handy when you're trying to figure out the specific elements to style. For this section I'm presenting 3 different cases

Changing the fonts for a theme could be done with something like this. Note that we import the fonts in the CSS rather than enqueuing the fonts in `functions.php` like we'll do later in the post. These styles will apply to the entire content.

```
@import  "https://fonts.googleapis.com/css?family=Handlee";
@import  "https://fonts.googleapis.com/css?family=Lato:400,700,400italic,7
@import  "https://fonts.googleapis.com/css?family=Source+Code+Pro:400,300'

body {
    font-family: Lato, Verdana, Helvetica, sans-serif;
    font-size: 16px;
/* 1rem = 16px */
    font-weight: 300;
}

h1,
h2,
h3 {
    clear: both;
    font-family: 'Handlee', cursive !important;
    margin-bottom: .25em;
    text-rendering: optimizeLegibility;
}
'Handlee'
```

When changing the font for h1 tags only when they appears in single post pages we need to figure out the class associated with the content that you want to change. To change the h1 tag in the title we need to change the `.entry-title` class.

```
h1.entry-title {
  clear: both;
  font-family: 'Handlee', cursive !important;
  margin-bottom: .25em;
  text-rendering: optimizeLegibility;
}
```

Adding styles for manually entered CSS is simple. I write embeds for Youtube and Vimeo videos using a syntax like the one below

```
<div class="video">
<iframe width="560" height="315"
  src="https://www.youtube.com/embed/FaOSCASqLsE?rel=0"
  frameborder="0" allowfullscreen></iframe>
</div>
```

And use the following CSS to make sure the videos take the full available screen width.

```
.video iframe {
  clear: both;
  display: block;
  margin: 1em auto;
  max-width: inherit;
  text-align: center
}
```

When working with these types of customization we need to be very, very careful when dealing with specificity issues. Check Estelle Weyl's post on CSS specificity for more information on how to work specificity.

Local changes using hooks and filters

`functions.php` is a theme specific customization file set aside for local customizations. Rather than updating entire templates we can update with actions and filters to customize and change the way

The idea is to build a function that does what we want and then use the add_filter function and pass the following parameters:

- The name of the filter
- The code to execute then the filter is triggered
- An integer for priority
- An integer for the number of argument

What does this mean in the real world? We'll look at two examples of how to customize Wordpress. We'll use a filter to change the way WP inserts images into a post using more semantically rich HTML5 that is easier to style later on.

```
function html5_insert_image($html, $id, $caption, $title, $align, $url) {
  $html5 = "<figure id='post-$id media-$id' class='align-$align'>";
  $html5 .= "<img src='$url' alt='$title' />";
  if ($caption) {
    $html5 .= "<figcaption>$caption</figcaption>";
  }
  $html5 .= "</figure>";
  return $html5;
}
add_filter( 'image_send_to_editor', 'html5_insert_image', 10, 9 );
```

The second example uses an action to add the parent's theme CSS to save ourselves from copying the CSS to a child theme. Actions

- Create a PHP function that should execute when a specific WordPress event occurs, in your plugin file.
- Hook this function to the event by using the add_action() function.

```
<?php
function enqueue_parent_styles() {
    wp_enqueue_style( 'pare'parent-style'_template_directory_uri().'/style
}
add_action( 'wp_enqueue_scripts', 'enqueue_parent_styles' );
'/style.css'?>
```

Rather than make the article way longer than it needs to be by giving you a full list of the hooks available for `add_action` and `add_filter` I'll refer you to the Wordpress Code Reference for Hooks.

Also make sure you use some way to namespace your functions to avoid conflict with functions already in Wordpress core or other plugins. Nothing sucks more than your action calling someone else's code (speaking from experience).

Difference between filters, actions and plugins?

Taken from: http://wordpress.stackexchange.com/questions/1007/difference-between-filter-and-action-hooks

Action Hooks

Actions Hooks are intended for use when WordPress core or some plugin or theme is giving you the opportunity to insert your code at a certain point and do one or more of the following:

- Use echo to **inject some HTML** or other content into the response buffer,
- **Modify global variable state** for one or more variables, and/or
- **Modify the parameters** passed to your hook function (assuming the hook was called by do_action_ref_array() instead of do_action() since the latter does not support passing variables by-reference.)

Filter Hooks

Filter Hooks behave very similar to Action Hooks but their intended use is to receive a value and potentially return a modified version of the value. A filter hook could also be used just like an Action Hook i.e. to modify a global variable or generate some HTML, assuming that's what you need to do when the hook is called. One thing that is very important about Filter Hooks that you don't need to worry about with Action Hooks is that the person using a Filter Hook must return (a modified version of) the first parameter it was passed. A common newbie mistake is to forget to return that value!

So what's the Real Difference?

In reality Filter Hooks are pretty much a superset of Action Hooks. The former can do anything the latter can do and a bit more albeit the developer doesn't have the responsibility to return a value with the Action Hook that he or she does with the Filter Hook.

Giving Guidance and Telegraphing Intent

But that's probably not what is important. I think what is important is that by a developer choosing to use an Action Hook vs. a Filter Hook or vice versa they are telegraphing their intent and thus giving guidance to the themer or plugin developer who might be using the hook. In essence they are saying either "I'm going to call you, do whatever you need to do" OR "I've going to pass you this value to modify but be sure that you pass it back."

We can use both filters and actions both in `functions.php` or a plugin. I usually wait until I've accumulated enough related filters or actions (usually 5) to move them to a plugin. There shouldn't be a difference in how the filters or actions work in either implementation.

Creating Child Themes

Before we start customizing our theme's template we'll take a little detour and talk about child themes. Every Wordpress theme can be used as the source of a child theme (also known as the parent theme) but there are frameworks like the Sitepoint Base theme or the Genesis Framework by Studio Press that are specifically made to be used as parent themes.

Child themes completely depend on their parents to work. The parent theme has to be installed in your Wordpress installation for the child to work at all. The child theme will modify and overwrite the parent with the changes you make without changing the parent at all. If you decide to start over all you have to do is remove and recreate the child theme.

There are numerous advantages to using a child theme:

- You don't have to start from scratch. The bulk of the work is already done for you so you can concentrate on the modifications that will work best for your site and content
- Upgrading the parent theme will not affect the child theme or its customizations
- You can use as much or as little of the parent as you want and can make as many or as few changes as you need or want to
- At a minimum a child theme needs three things:

- A folder
- A style sheet
- A local `functions`.php file

The Creation Process

> I normally do this directly on the server using Unix shell tools and programs. If you're not comfortable doing this then it's a good idea to setup a local development environment and then zip the content of your child theme and install it using the theme installer.
>
> If you choose to go the local route Google `creating child theme wordpress` for general instructions.
>
> Your choice

Connect to your host and create your child theme's folder in `wp-content/themes`. I normally ssh to my host to make these changes so, once i've logged in and changed to `path/to/my/blog/wp-content/themes` I run the following commands:

```
-bash-3.2$ mkdir twentyseventeen-child
-bash-3.2$ cd twentyseventeen-child/
```

Next we'll create our style. In the child theme directory (`twentyseventeen-child`) run the following commands to create the `style.css` file and open it for editing:

```
touch style.css
vi style.css
```

press the `i` key to go into interactive mode and copy the content below.

```
/*
 Theme Name:   Twenty Seventeen Child
 Theme URI:    http://example.com/twenty-seventeen-child/
 Description:  Twenty Seventeen Child Theme
 Author:       John Doe
 Author URI:   http://example.com
 Template:     twentyseventeen
 Version:      1.0.0
 License:      MIT
 License URI:  https://caraya.mit-license.org/
 Tags:         light, dark, two-columns, right-sidebar, responsive-layout
 Text Domain:  twenty-seventeen-child
*/
```

- **Theme name** This is the name that will show up for your theme in the WordPress back end
- **Theme URI** This points to the website or demonstration page of the theme at hand. This or the author’s URI must be present in order for the theme to be accepted into the WordPress directory
- **Description** This description of your theme will show up in the theme menu when you click on “Theme Details”
- **Author** This is the author’s name
- **Author URI** You can put your website’s address here if you want.
- **Template** This part is crucial. Here goes the name of the parent theme, meaning its folder name. Be aware that it is case-sensitive, and if you don’t put in the right information, you will receive an error message!
- **Version** This displays the version of your child theme. Usually, you would start with 1.0.
- **License** This is the license of your child theme. WordPress themes in the directory are usually released under a GPL license. This is only important if you will add your theme to the Wordpress directory.
- **License URI** This is the address where your theme’s license is explained
- **Tags** The tags help others find your theme in the WordPress directory. Thus, if you include some, make sure they are relevant.
- **Text domain** This part is used for internationalization and to make themes translatable. This should fit the “slug” of your theme.

Most of this information is only relevant if you choose to publish your child theme. If you will not publish it, either because it’s a personal project or because it’s for a client then the minial stylesheet header looks like this:

```
/*
 Theme Name:   Twenty Seventeen Child Theme
 Description:  A child theme of the Twenty Seventeen WordPress theme
 Author:       Carlos Araya
 Template:     twentyseventeen
 Version:      1.0.0
*/
```

Once it has been copied press escape and then wq to save your work and quit the editor.

The next step is to create `functions.php` and add a function to enqueue the parent theme's styles and the child styles.

To create the file run these commands.

```
touch functions.php
vi functions.php
```

Copy the content below to the `functions.php` file. Press `i` to enter interactive mode and then paste the script below. This will use Wordpress' system to import the parent theme and the child theme's stylesheets in the correct order to make the cascade work for you.

```
<?php
// function taken from:
// http://justintadlock.com/archives/2014/11/03/loading-parent-styles-for
function my_enqueue_styles() {
  /* If using a child theme, auto-load the parent theme style. */
  if ( is_child_theme() ) {
    wp_enqueue_style( 'parent-style', trailingslashit( get_template_direc
  }

  'parent-style'/* Always load active theme's style.css. */style( 'style'
}
add_action( 'wp_enqueue_scripts', 'my_enqueue_styles' );
```

```
function html5_insert_image($html, $id, $caption, $title, $align, $url, $s
  $src  = wp_get_attachment_image_src( $id, $size, false );
  $url = str_replace(array('http://','https://''style'//','https://'), '//
  $html5 = "<figure>";
  $html5 .= "<img src='$url' alt='$alt' class='size-$size' />";
  if ($caption) {
    $html5 .= "<figcaption class='wp-caption-text'>$caption</figcaption>"
  }
  $html5 .= "</figure>";
  return $html5;
}
add_filter( 'image_send_to_editor', 'html5_insert_image', 10, 9 );
?>"<figure>"''?>
```

Once it has been copied press escape and then wq to save your work and quit the editor.

If you want to change templates from the parent theme in your child you will have to copy the structure from the parent to the child. Because we'll make changes to a template, we'll copy it now. To do so run the following commands from the root of the child theme:

```
mkdir -p template-parts/post
cd template-parts/post
# copies content from parent
cp ../../../twentyseventeen/template-parts/post/*.php .
```

This will make all the post specific templates available to edit in the child theme.

Good reads

- Wordpress Theme Handbook
- Child themes

Customizing Templates

The next step in customizing a Wordpress implementation is to edit a template. Unless you're working in a one-off theme for a client I strongly suggest you work on a child theme like we did above… this will save you from a lot of headaches during development.

In this section we'll assume you created a

dding a link to the [accelerated mobile pages](#) (AMP) version of a post when you're viewing individual posts.

You have to edit the specific template. In the case of the theme I'm using (twentyseventeen) the template to edit is `template-parts/post/content-single`.php. What I want to do is add the link to the header either next to or below the post title. I've taken the full template for this demo.

```
<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
<?php
  if ( is_sticky() && is_home() ) :
    echo twentyseventeen_get_svg( array( 'icon'icon'thumb-tack' ) );
  endif;
?>
<header class="entry-header">
  ___PHP3__<header class="entry-header">lso available in <a href='___PHP4
    ___PHP5___
</header><a href='___PHP4___/amp'><!-- .entry-header -->="post-thumbnail"
    <a href="___PHP7___">
      ___PHP8___
    </a>
  </div><!-- .post-thumbnail -->
<a href="___PHP7___"><!-- .post-thumbnail -->PHP10___
</div><!-- .entry-content -->

___PHP11___
  ___PHP1</div><!-- .entry-content -->

___PHP11___
```

```
  ___PHP12___
___PHP13___

</article><!-- #post-## -->
```

We don’t need to implement AMP functionality. That has been implemented through a third-party plugin that we’ll discuss in more detail in the next section.

When to use third party plugins?

In the previous section we added a link to the AMP version of single posts but we didn’t have to worry about how to convert the posts into AMP, I used a plugin from Automattic to generate the AMP content. It will add code and customizable templates to make AMP work with your content; you can then choose to customize the code or use it as is.

This is a good point to stop and ask ourselves why do we need plugins and how safe are we installing third party plugins (particularly form a security stand point).

First things first. **Only download plugins from sites you trust**. Plugins can modify the way your Wordpress installation works and cause all sorts of unsafe and unpredictable consequences.

Next. **Make sure that you actually need the features you’re installing plugins for**. Just like with all web content the more plugins you have installed the bigger an impact on performance.

Lastly. **Test the plugins in a development server**. Before putting anything in a production server you need to test it in development server that mirrors as closely as possible your production environment. If you can afford it it would also help to have third party reviews of your plugins and how they interact with your code.

Using Wordpress REST API to create a completely different front end

> WordPress is moving towards becoming a fully-fledged application framework, and we needed new APIs. This project was born to create an easy-to-use, easy-to-understand and well-tested framework for creating these APIs, plus creating APIs for core.
>
> …
>
> The API exposes a simple yet powerful interface to WP Query, the posts API, post meta API, users API, revisions API and many more. Chances are, if you can do it with WordPress, WP API will let you do it.
>
> from the wp-api plugin page

The API is included in the core Wordpress installation as of version 4.7 and newer. There is a plugin available for versions between 4.4 and 4.6

So why would I use this API rather than the traditional Wordpress API? The API gives you the freedom to create your UI and UX in whatever language you want and in whatever way your users will be most comfortable with. To test the API, and give myself a good refresher on Polymer, I created a Polymer element that fetches the latest 10 posts from my blog and converts them to HTML.

The code looks like this:

```
<template>
    <!-- iron-ajax action -->
    <iron-ajax
      auto
      url="http://<iron-ajax
      auto
      url="http://rivendellweb.net/wp-json/wp/v2/posts"
      params='{"page": 1, "_embed": 1}'
      handle-as="json"
      last-response="{{posts}}"
      debounce-duration="300"><!-- local DOM for your element -->s-contai
      <template is="dom-repeat" items="{{posts}}" as="post">
        <paper-card heading tab-index='0'>
          <h1>
            <marked-element markdown="[[post.title.rendered]]">
              <div class="markdown-html" tabindex="0"></div>
            </marked-element>
          </h1>
          <div class="post-meta">
     <template is="dom-repeat" items="{{posts}}" as="post">st update: [[p

            <p>Posted by: [[post._embedded.author.0.name]] in [[post._emb
          </div> <!-- closes post-meta </p><!-- closes post-meta -->d-con
            <a href="[[post._embedded.wp:featuredmedia.0.link]]"><img src

            <template is="dom-if" if="[[post.content.protected]]">

              <h1>Private Post</h1>

              <p>This is a protected post and the content is not availabl
            </template>
            <marked-element markdown="[[post.content.rendered]]">
              <div class="markdown-html" tab-index='0'></div>
            </marked-element>
          </div> <!-- closes card-content -<a href="[[post._embedded.wp:f
</template>
</template><!-- closes cards-container -->
</template>
```

Examples of apps built with the REST API

- REST API example apps
- WordPress Posts in React

When it’s best to create a theme from scratch

The biggest and most complex customization option is to build a theme from scratch. I seldom start completely from scratch but use something like Sitepoint’s base theme, underscores. I used to work with Thesis but dropped it when the development became too complicated to do without using individual blocks and being forced into visual design.

Before you jump into this adventure consider the following list of pros and cons that I consider essential when considering if you want to develop a theme from scratch or not:

Pros

- You can use whatever technology stack you’re most comfortable with (I uses SCSS for my styles and have used Coffeescript to write scripts before converting them to ES5)
- You have total control over the looks and styles of your site. You can create as many or few templates as you want

Cons

- You need to know (or know where to find information about) how to fully build templates and the PHP Way to do things
- You still need to have a design ready to go. Tania’s article referenced later uses a Bootstrap template as the basis for a theme. Zurb Foundation also provides a blog template using their platform.

Rather than tell you how to do it, mostly because it would take several posts just as long as this one to cover in the level of detail that this needs. Tania Rascia has a very good series of articles covering how to build a theme from scratch

Closing

Customizing a Wordpress theme can be as complicated as you need it to be. You can work on different levels of customization to achieve your desired result. With the suite of technologies we’ve discussed the possibilities are only limited by your imagination.