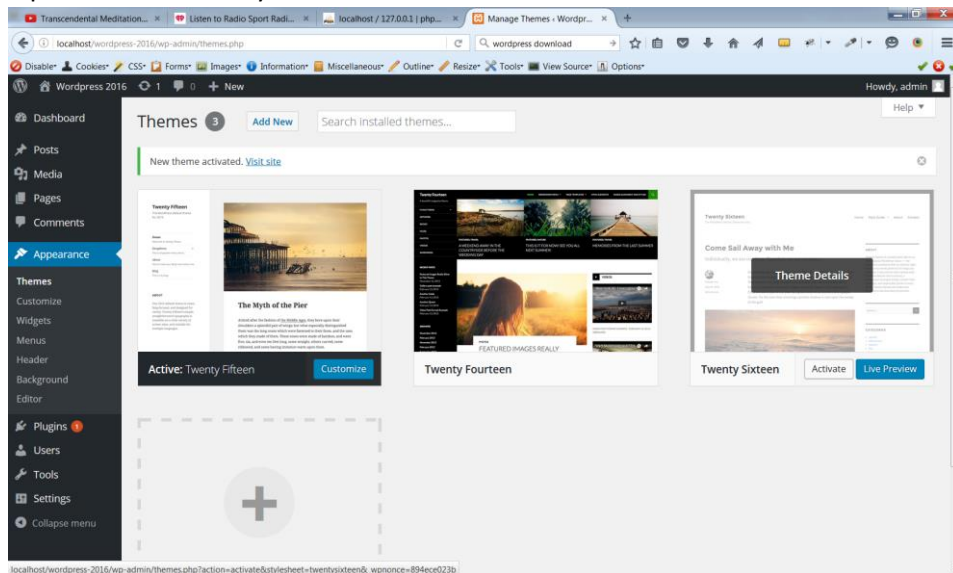


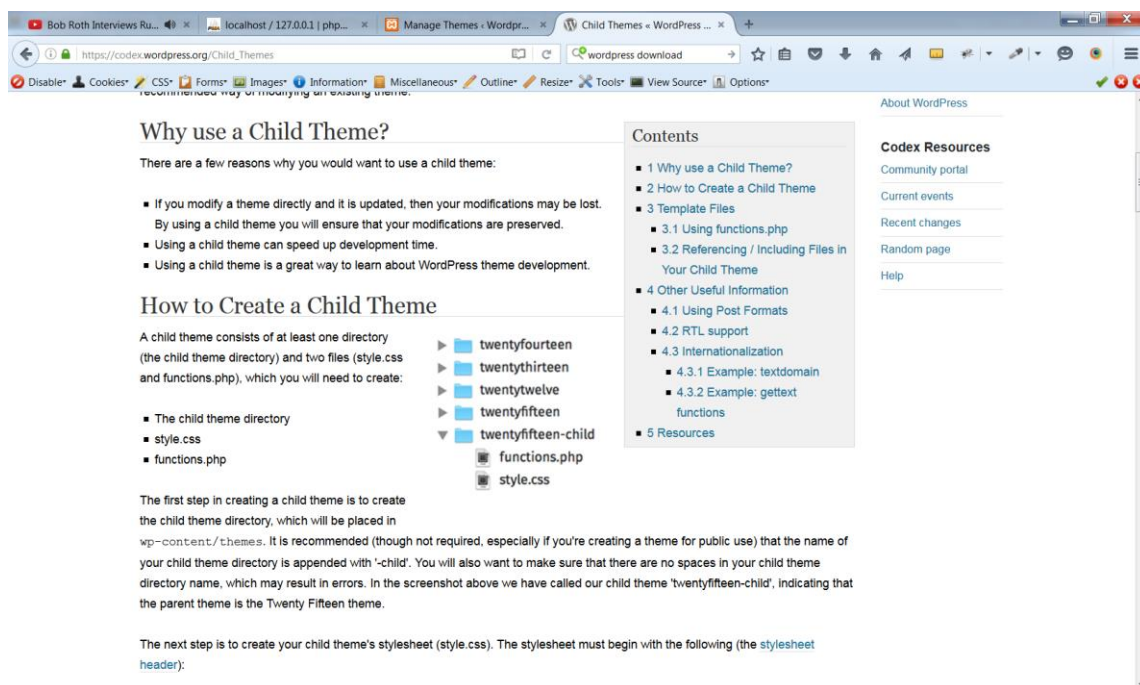
After Installing Wordpress:

1. Login to the dashboard backend GUI and go to Appearance -> Themes.
2. Select a parent theme that you wish to use and click the activate button to activate it.

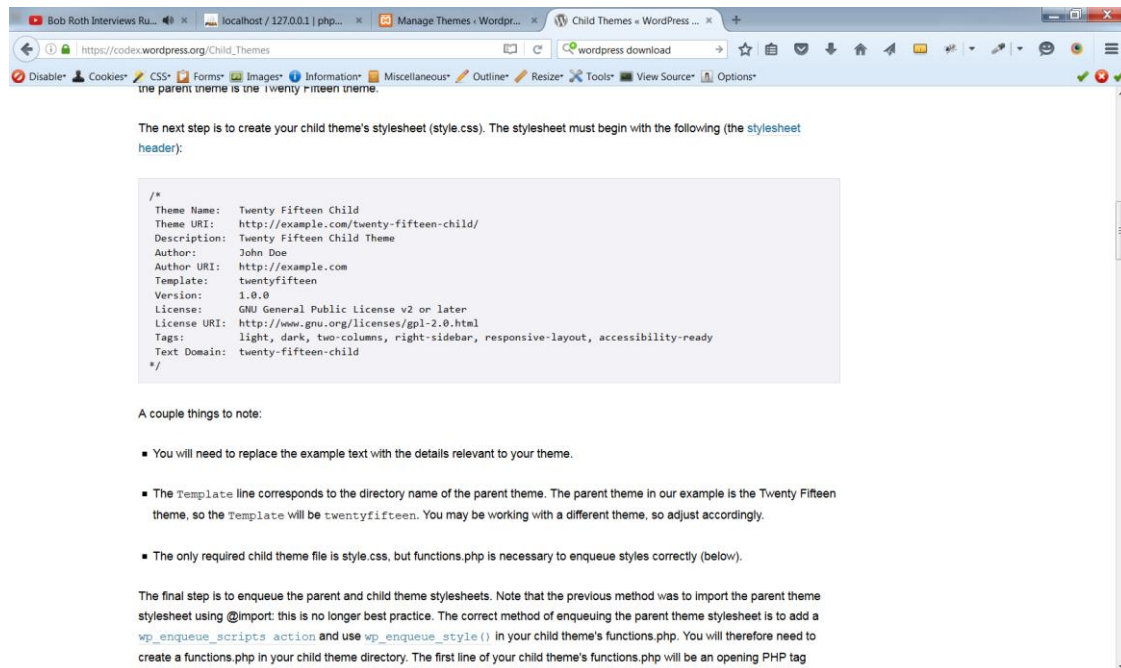


3. For this exercise I have chosen to use 2016 as my parent theme.

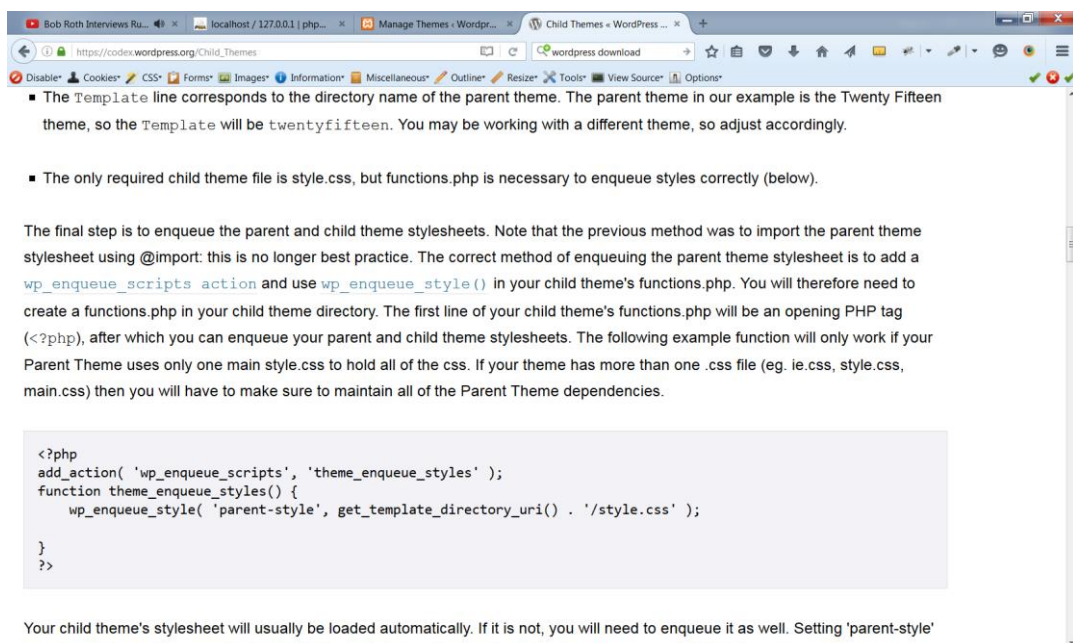
We now need to create a Child Theme in Wordpress. If you google “wordpress child theme” you should be able to see the following page “Child Themes << Wordpress codex”. Click that google link. We are aiming to create a directory like the one displayed below:



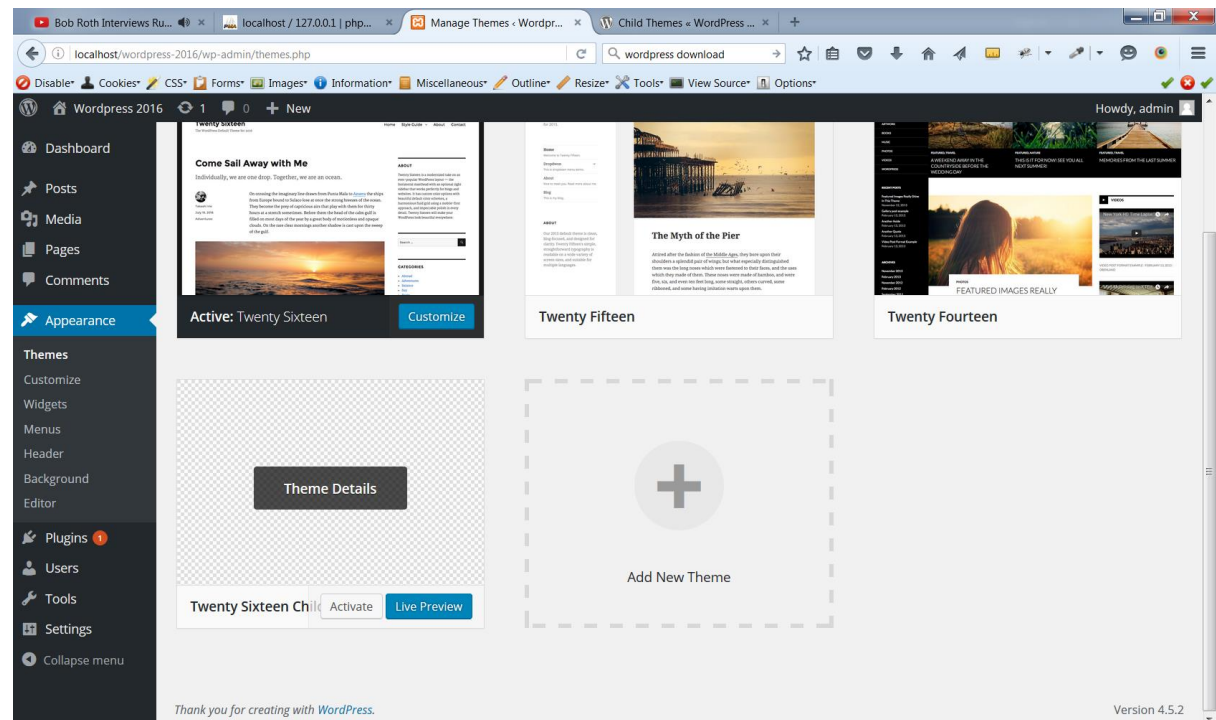
4. Create a **style.css** file in sublime. The key ingredient is to set up a comment section at the top of the style.css file:



5. After saving this file you with the comment from wordpress (above) we will also need to create our own **functions.php** file. This file allows us to use php code to instruct wordpress on how we want to use and modify our child theme.
6. Create a `functions.php` in sublime text. Add the following code snippet to the `functions.php` file and then save the file in your child theme folder:



- After the functions.php file is saved, you should be able to see the child theme as a selectable icon in the wordpress dashboard (GUI area):



.....and your project 'Dir' should look like this:

- If you modify a theme directly and it is updated, then your modifications may be lost.
By using a child theme you will ensure that your modifications are preserved.
- Using a child theme can speed up development time.
- Using a child theme is a great way to learn about WordPress theme development.

How to Create a Child Theme

A child theme consists of at least one directory (the child theme directory) and two files (style.css and functions.php), which you will need to create:

- The child theme directory
- style.css
- functions.php

- twentyfourteen
- twentythirteen
- twentytwelve
- twentyfifteen
- ▼ twentyfifteen-child
 - functions.php
 - style.css

- 2 How to Create a Child Theme
- 3 Template Files
 - 3.1 Using functions.php
 - 3.2 Referencing Your Child Theme
- 4 Other Useful Information
 - 4.1 Using Post Thumbnails
 - 4.2 RTL support
 - 4.3 Internationalization
 - 4.3.1 Example
 - 4.3.2 Example functions
- 5 Resources

- Congratulations! You have now created a child theme that you can override with your child themes style.css file. You also have a functions.php file that will allow you to do powerful things with wordpress. As we shall see.....

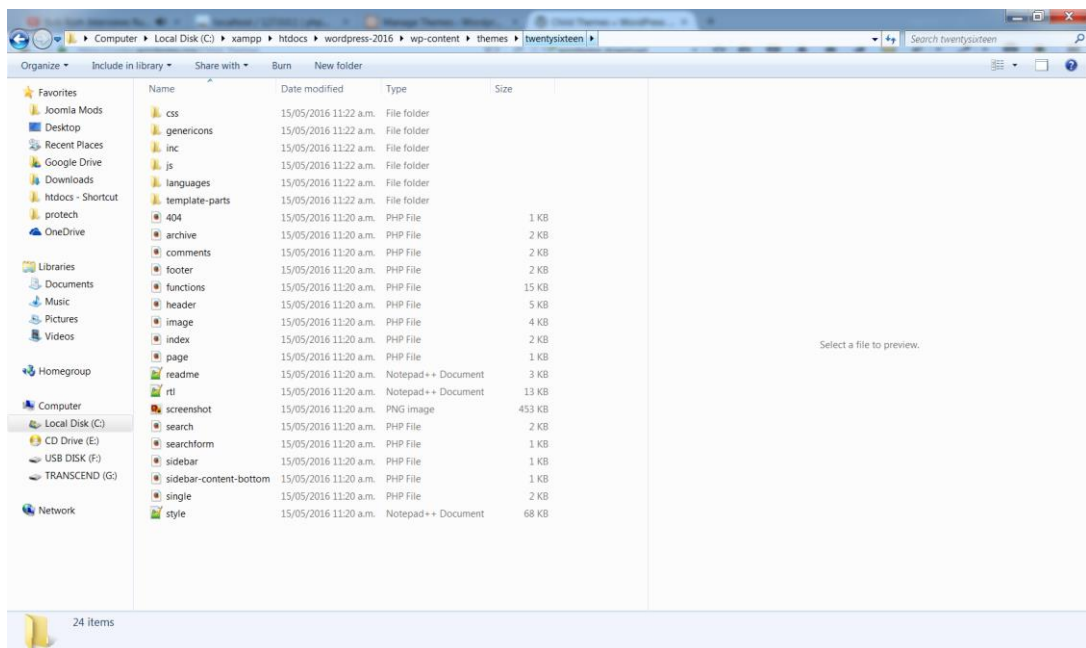
The Power of Child Themes:

The theory behind Wordpress and most CMS systems like Joomla and Drupal, is that they can be updated. This allows the CMS brands (Wordpress, Joomla etc.) to update your CMS version if viruses and bugs are discovered over time.

My recommendation is to avoid updating your CMS at all costs! If you absolutely have to, then it is ok, because you have created a child theme. The child theme means you have a section of the site that is totally protected from updates. As we will see, the child theme can override the parent theme if we use it correctly.

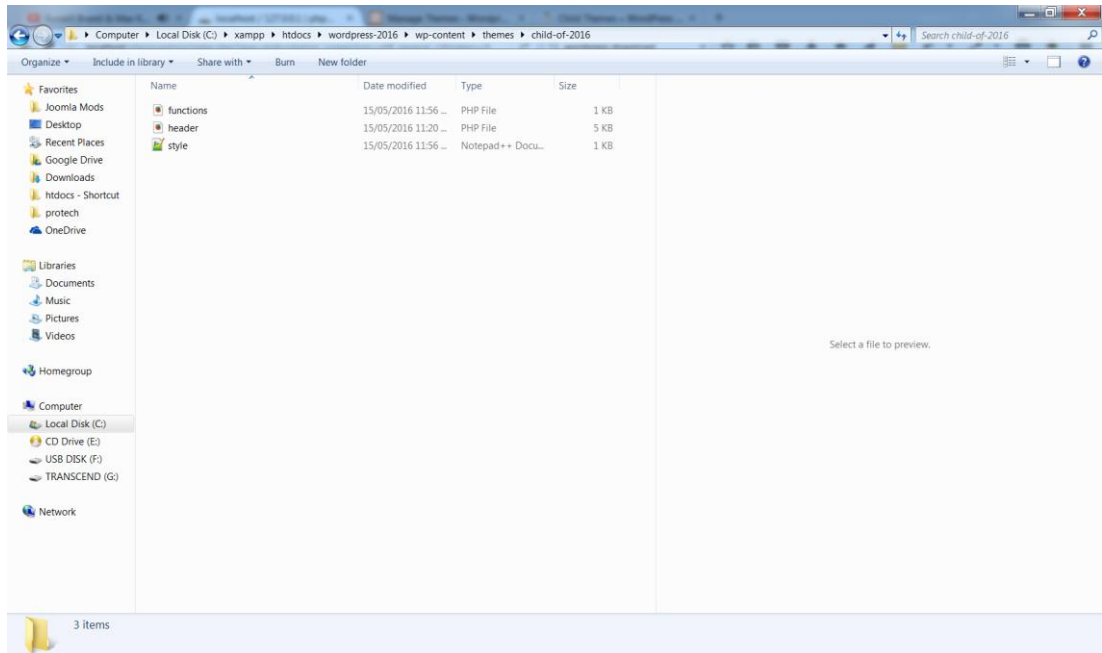
1. Overriding Parent Theme .php Files:

Our parent theme already has many files within it. This is what the file structure of **twentyseventeen** theme looks like:

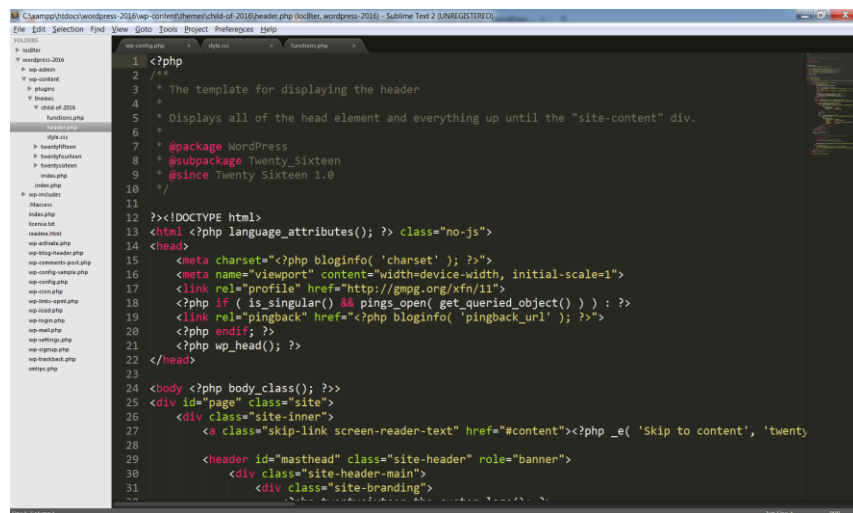


Having a child theme means we can copy parent theme files across to our **child theme folder**. A copied and transferred file, will become the actual file that Wordpress uses to render the site. This is a very powerful tool. We now have direct access to the php and html of various files and can safely override them (re-code) as we see fit. Remember there will always be a backup in your parent theme.

A basic way to see this in action is by copying the **header.php** file to the child theme folder. Go to your **parent theme** (I am using **twentysixteen**) and copy the **header.php** file and paste it into your child theme folder. Your child theme should now look like this:



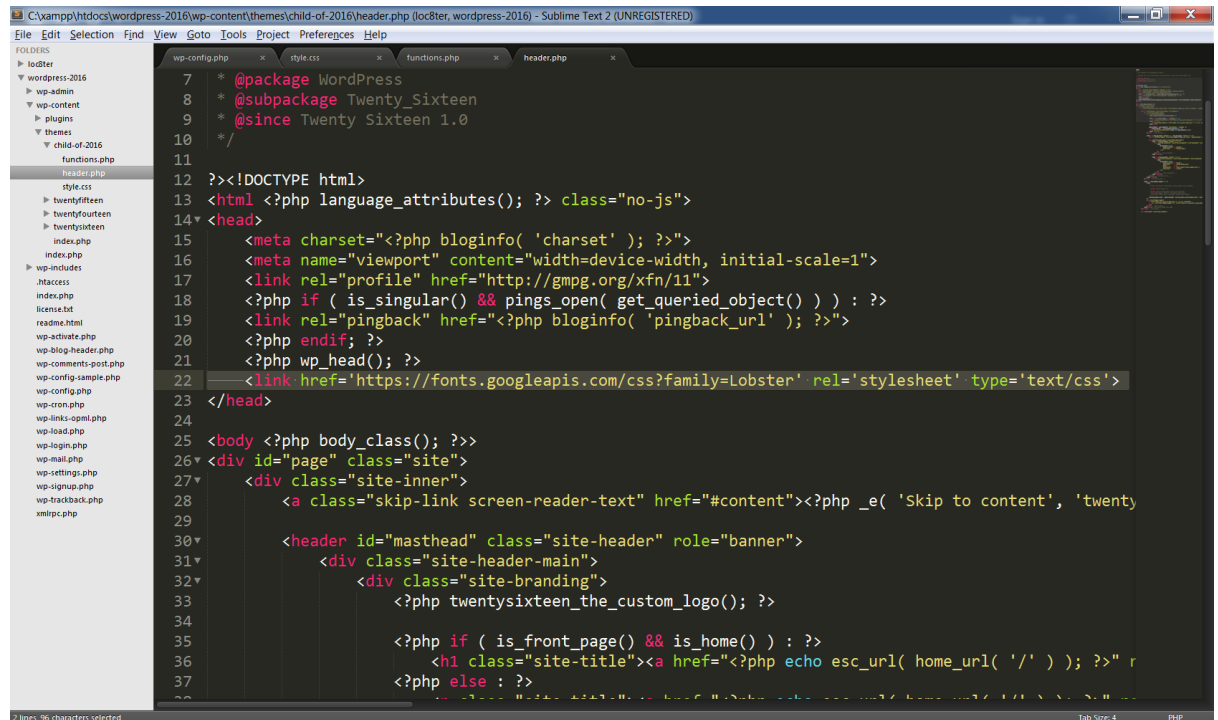
2. After you have pasted the header.php file, open it in sublime:



You can see that this file is a mix of .php and .html. We can now add code to this file as we see fit. This approach of copying files from the **parent theme to the child theme**, can be done with most parent theme files.

The header.php file is ideal as it allows us to add css stylesheet links. So If you want to add a google font, font-awesome or any other kind of css file to your project, you can add it to this copied header.php file.

On the example below I have added a google font at line 22 of the header.php file:

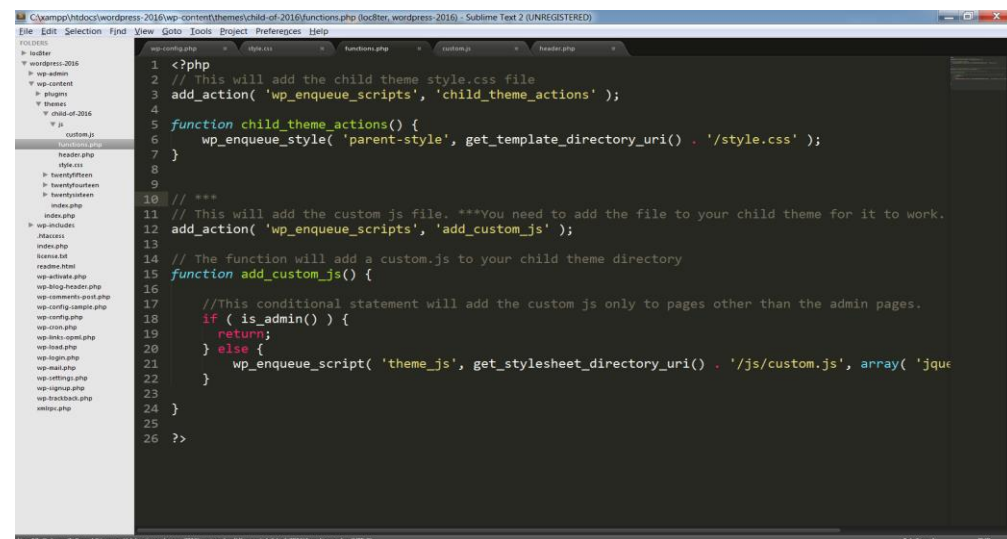


```
7  * @package WordPress
8  * @subpackage Twenty_Sixteen
9  * @since Twenty_Sixteen 1.0
10 */
11
12 <?><!DOCTYPE html>
13 <html <?php language_attributes(); ?> class="no-js">
14 <head>
15     <meta charset="<?php bloginfo( 'charset' ); ?>">
16     <meta name="viewport" content="width=device-width, initial-scale=1">
17     <link rel="profile" href="http://gmpg.org/xfn/11">
18     <?php if ( is_singular() && pings_open( get_queried_object() ) ) : ?>
19     <link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">
20     <?php endif; ?>
21     <?php wp_head(); ?>
22     <link href='https://fonts.googleapis.com/css?family=Lobster' rel='stylesheet' type='text/css'>
23 </head>
24
25 <body <?php body_class(); ?>>
26 <div id="page" class="site">
27     <div class="site-inner">
28         <a class="skip-link screen-reader-text" href="#content"><?php _e( 'Skip to content', 'twenty
29
30     <header id="masthead" class="site-header" role="banner">
31         <div class="site-header-main">
32             <div class="site-branding">
33                 <?php twenty_sixteen_the_custom_logo(); ?>
34
35                 <?php if ( is_front_page() && is_home() ) : ?>
36                 <h1 class="site-title"><a href="<?php echo esc_url( home_url( '/' ) ); ?>" r
37                 <?php else : ?>
38                 <h2 class="site-title"><a href="<?php echo esc_url( home_url( '/' ) ); ?>" r
39                 <?php endif; ?>
40             </div>
41             <div class="site-description"><?php bloginfo( 'description' ); ?></div>
42         </div>
43     </div>
44 </div>
45 </body>
46 </html>
```

You also have the option of adding you own custom classes and id's to child theme override files (like header.php). This will give you greater control of targeting elements with css.

Adding a custom.js File to the Project:

We could do this by adding a .js link to the header.php or footer.php child theme override file (like what we just did with the google font file above). You are welcome to do so, but I have added some php code to functions.php. You can see and download it via the github repo:



```
1 <?php
2 // This will add the child theme style.css file
3 add_action( 'wp_enqueue_scripts', 'child_theme_actions' );
4
5 function child_theme_actions() {
6     wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
7 }
8
9
10 // ***
11 // This will add the custom js file. ***You need to add the file to your child theme for it to work.
12 add_action( 'wp_enqueue_scripts', 'add_custom_js' );
13
14 // The function will add a custom.js to your child theme directory
15 function add_custom_js() {
16
17     //This conditional statement will add the custom js only to pages other than the admin pages.
18     if ( is_admin() ) {
19         return;
20     } else {
21         wp_enqueue_script( 'theme_js', get_stylesheet_directory_uri() . '/js/custom.js', array( 'jquery'
22     }
23 }
24
25
26 >
```

The updated functions.php file will allow you to have a custom.js in your child theme folder. Please uncomment the code if you want to take this approach. Remember to make a folder in your child theme called “js” and inside that folder place a custom.js file.

Good luck and please email if you have any questions.