MMP350 Class Notes Week 7

This Week's goals

- Presentations – Fane, Juan and Kristine
- Next Week's Presentations
- Homework
- Brief review of CSS issues that were revealed by quizzes
- Review of PHP basics
- Exercises
- Review of WordPress Template Hierarchy
- Creating a child theme

While you are working on the exercises I will meet with students to give midterm reports

Homework

Try to customize the theme you chose for homework this week.
Implement at least one change to your project.
Review notes in anticipation of a quiz.

Next Week's Presentations

Xiaolin and Marlon

Review of Properties of Inline Elements

Please refer to the file **line_height_and_font_size.html** included with today's class materials.

Review of PHP Basics

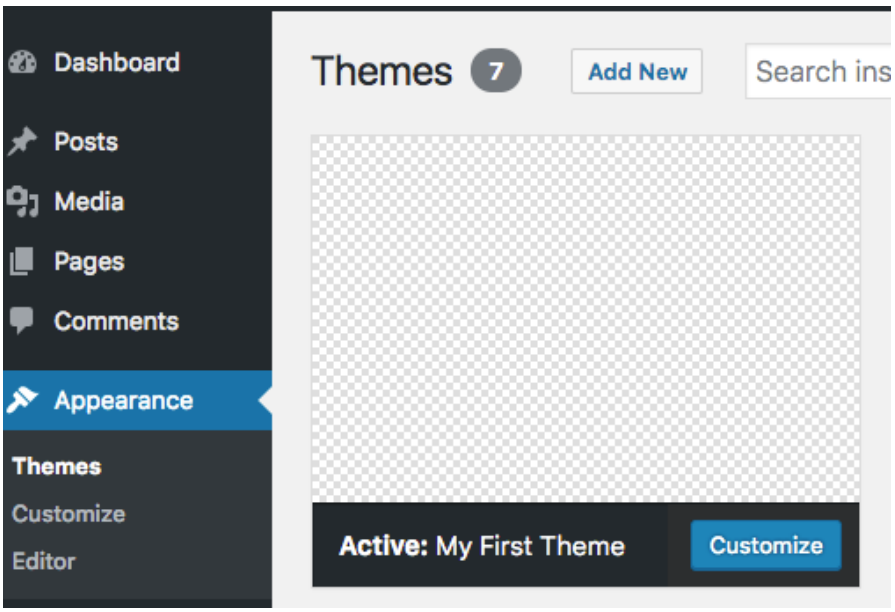
1. PHP requires a server like Apache in order to be executed.
2. PHP can be embedded in html. This embedding is called a server-side-include, because it executes code on the server.
3. Variables begin with $
4. All lines end with a semi-colon
5. PHP files / statements begin with **<?php** and end with **?>**
6. **=** is the assignment operator
7. **==** is the string comparison operator
8. **.** (period) is the string concatenation operator
9. php variables can be concatenated within a strong, for example

   sFullname = "$fname . $lname"

Exercise: Add Featured Image Functionality to Your Theme

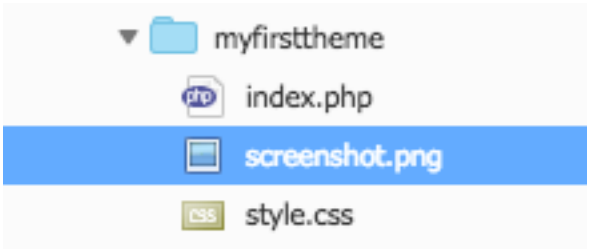
Step 1: Activate My First Theme

Go to Appearance Themes and activate my first theme. If you have not uploaded `myfirsttheme.zip`, you can find it in the materials for today's class.

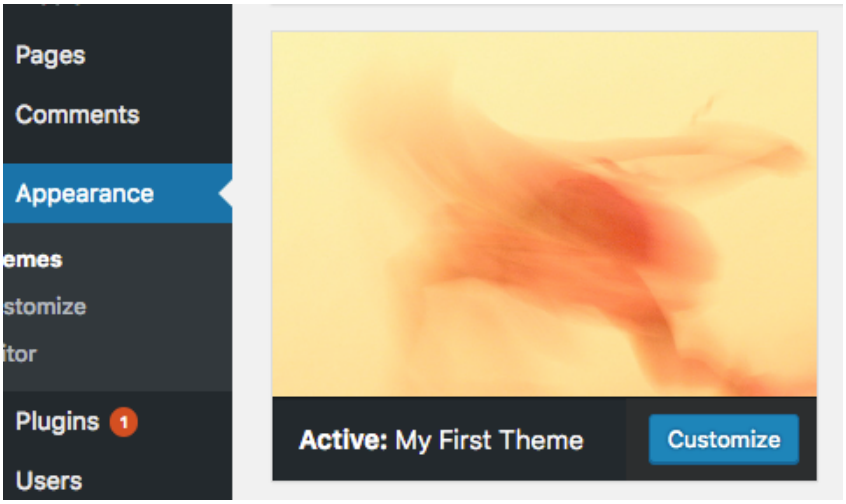


Notice how it doesn't have a theme image. Theme images use the file `screenshot.png` in the root folder of your theme.

To add a Theme Image drag a png file called `screenshot.png` to your main folder:



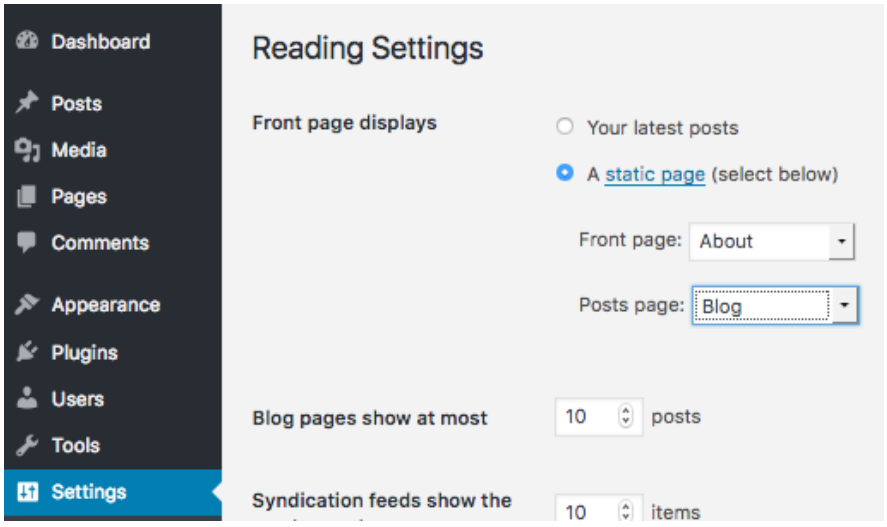
When you return to Appearance Themes your theme will now have a theme image.
Creating an image for your project's theme is a requirement for your final project.



Step 2: Create 3 Pages

If you do not have pages on your site already, create three pages modeled on the following

Page 1: Call your first page **About**. Go to Reading Settings and make it your **Front Page**



Page 2: Call this page **Photos**.

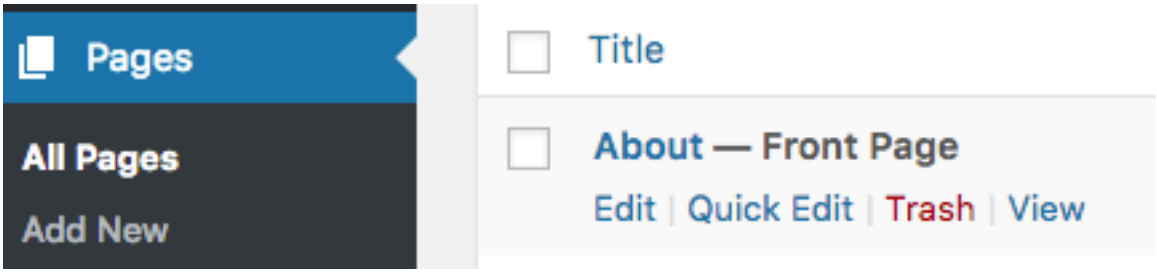
Page 3: Call this page **Contact**.

Remember to publish your pages. We will use these pages in future exercises.

If you do not have any blog posts, create two or three. Give them the category “featured”.

Step: Viewing a Featured Image

Activate the **TwentySeventeen** theme and Edit your Front Page.



On the bottom right of your screen you will see **Set featured image**

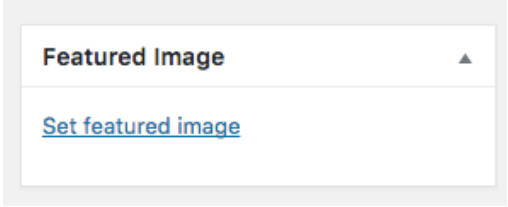


Figure: Images for blog posts are set through Set featured image.

Re-activate My First Theme and return to your **Front Page** page.

Q: The featured image option is missing? Why?

A: Featured image functionality needs to be enabled in your functions.php before it can be used.

Activate **My First Theme** and add a new file called **functions.php**, described below.

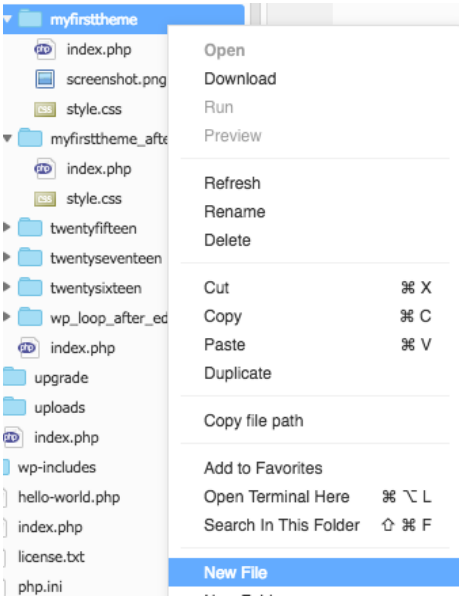


Figure: right click with your mouse on the theme name and select New File to add a file.

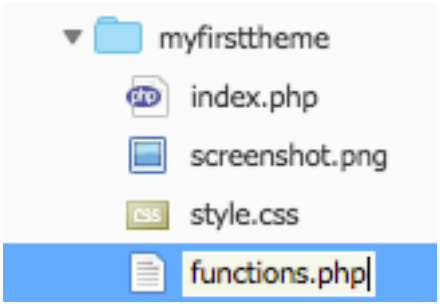


Figure: Give your new file the name functions.php

Add the following code to `functions.php`

```
<?php
add_theme_support( 'post-thumbnails' );
?>
```

After you refresh, featured images can now be added to your posts and pages.

WordPress Loop

Let's use php to make our site dynamic. We will begin with our main page, **index.php**. Note that index.php is the base file for the folder, as opposed to index.html.

Let's start with the `<head>` section of **index.php**; it is at the beginning of the document.

```
index.html
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Theme Starter Files</title>
7      <link rel="stylesheet" href="style.css">
8  </head>
```

Figure: the head section of index.php

Most of this will stay the same, but we need to change two lines.

First we'll change the `<title>`. WordPress themes should be customizable, so we will begin replacing default or hard coded content with php variables that are provided by WordPress.

```
6      <title><?php bloginfo('name');?> | <?php bloginfo('description')?></title>
```

Figure: Inserting WordPress functions into the <title> element.

This is the code you will be adding:

```
<?php bloginfo('name'); ?> | <?php bloginfo('description')?>
```

The first php tag uses the bloginfo() function to retrieve the blog name. The second line uses the same function to retrieve the blog description, which is a reference to the tagline from the Dashboard settings

By using php variables, we can get those pieces of information, filled in by the user or by us, at any part of our website.

Next change the CSS link tag:

```
<link rel="stylesheet" type="text/css" href="<?php bloginfo('stylesheet_url'); ?>">
```

Figure: Adding a php function to a <link> element

This is the code you will be adding:

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>">
```

Again we're using the `bloginfo()` function, this time to get the URL location of the stylesheet. Because WP uses a more complex file structure than a static website, it is a good idea to use this references instead of a direct URL to the style sheet.

Note that php can be used inside of the quotes of html attributes, like href and rel.

Finally, add this tag inside the head section:

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

Figure: Adding a function in the head section.

This is the code you will be adding:

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

This is going to load some WordPress functionality into the head of the website. We're going to skip the `<header>` and `<nav>` for now, we'll cover those in depth in upcoming lessons.

Skip ahead to this section:

```
25          <div class="posts">
26              <article class="post">
27                  <header>
28                      <h2 class="entry-title"><a href="permalink_to_title">The Title of
                        the Post</a></h2>
29
30
31                  </header>
32
33                  <div class="content">
34                      <p>
35                          the_content() of the post goes here
36                      </p>
```

We're going to add a few lines and make some changes here.

First add this line, called the WP Loop. We'll cover this in depth soon. The important thing to know now is that WP Loop does most of the work of going through all of your posts or pages and grabbing the content from the database.

```
25          <div class="posts">
26
27              <?php if(have_posts()) : while(have_posts()) : the_post(); ?>
28
29              <article class="post">
```

This is the code you will be adding:

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

Next, replace all of `<a href="permalink_to_title">The Title of the Post</a>` with the following:

```
<h2 class="entry-title">
    <a href="<?php the_permalink(); ?>">
        <?php the_title(); ?>
    </a>
</h2>
```

This is the code you will be adding:

```
<h2 class="entry-title">
      <a href="<?php the_permalink();?> ">
            <? the_title(); ?>
      </a>
</h2>
```

These lines are using functions to get the link to each post and the title of the post. You should start to see a pattern emerging.

Next, replace the entire section `<div id="content"> ... </div>` with this line:

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

Next replace the line `<p class="entry-meta">Posted on February 21, 2005 by <span class="author">author</span></p>` with this:

```
<p class="entry-meta">Posted on <?php the_date(); ?> by
<span class="author"><?php the_author(); ?></span>
</p>
```

Figure: adding the author of the website and date of each post.

This is the code you will be adding:

```
<p class="entry-meta">Posted on <?php the_date(); ?> by

      <span class="author"><?php the_author(); ?></span>

</p>
```

Finally, delete the entire second `<article>`. With WordPress working, we only have to make a single template for each post, which will be populated with new content.

Then, after the `<article>` section, before the end of the `<div id="content">` section, add the following:

```
<?php endwhile; else: ?>
    <p><?php _e('Sorry, no posts matched your criteria.'); ?></p>
<?php endif; ?> <!--End of posts -->
```

Figure: Exception handling while loading blog posts.

This tells WordPress what to do once it's loaded all of the blog posts or there are no blog posts to load.

Let's change a couple of things in the `<footer>`:

```
<footer>
    <p class="copyright">&copy; 2014</p>
    <p class="webdesigner">website by Your Name</p>
</footer>
```

Add the current date using the function `the_date()` and author, using `the_author()` functions to the footer:

```
<footer>
    <p class="copyright">&copy; <?php the_date(Y);?></p>
    <p class="webdesigner">website by <?php the_author(); ?></p>
</footer>
```

Figure: the_date() and the_author () functions

That's it for **index.php** for now.

Let's edit **style.css** and then test our theme.

```
/*
Theme Name: My First Theme
Author: Brian MacMillan
Description: A basic starter theme for MMP350 portfolio project.
*/
```

Right now this is all you will see in **style.css**. In order for WordPress to recognize the theme, it needs to have this comment with this basic information. Replace the info for **Theme Name**, **Author** and **Description** with your own information.

Once you're done, view the results.

Debugging

Things are never as smooth in real life as they are in class examples. More likely than not the edits you just did created a bug.

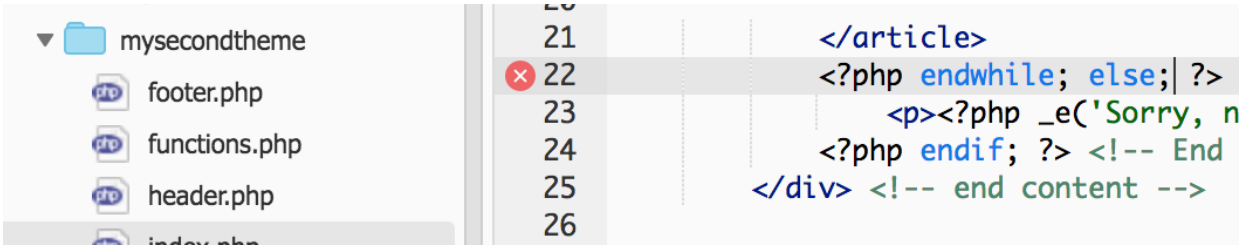
Let's take a moment off to fix a common bug.

Load and activate **mysecond_theme.zip.** This theme contains the edits you just made, but I have intentionally left a bug in the theme.

View the results. You will see the following error message:

Parse error: syntax error, unexpected ';', expecting ':' in /home/ubuntu/workspace/wp-content/themes/myfirsttheme_after_edits/index.php on line 86 Call Stack: 0.0003 236256 1. {main}() /home/ubuntu/workspace/index.php:0 0.0003 236704 2. require('/home/ubuntu/workspace/wp-blog-header.php') /home/ubuntu/workspace/index.php:17 0.0275 2908416 3. require_once('/home/ubuntu/workspace/wp-includes/template-loader.php') /home/ubuntu/workspace/wp-blog-header.php:19

Now go to the Cloud9 menu and select `index.php` for this theme:



The main error message is **"unexpected ';', expecting ':'**

Can you figure out what the error is and correct it?

Review: The WordPress Loop

Here's the WordPress explanation of the Wordpress Loop.

```
<div class="posts">
  <!-- Starting posts-->
    <?php if(have_posts()) : while(have_posts()) : the_post(); ?>

    <article class="blogPost">

    <!-- Title of the post -->
      <h2><a href="<?php the_permalink(); ?>">
        <?php the_title(); ?>
        </a></h2>

    <div class="content">
        <!-- Post's content -->
      <?php the_content(); ?>

    </div>
  </article>

  <?php endwhile; else: ?>
  <p>
    <?php _e('Sorry, no posts matched your criteria.'); ?>
  </p>
  <?php endif; ?> <!--End of posts -->
</div> <!-- end content -->
```

beginning the loop

html tags & css classes

get title from post

& content

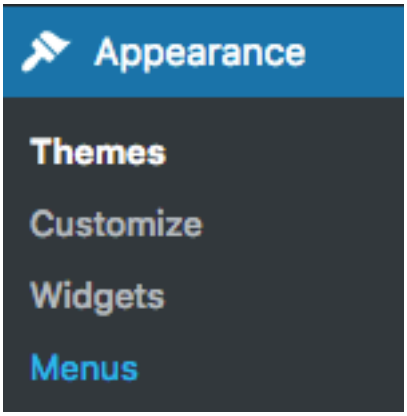
end loop

Menus, Headers and Footers

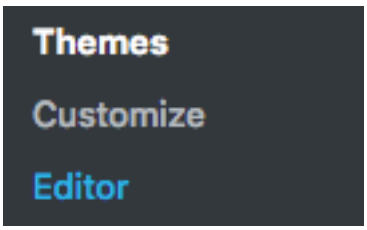
Now lets create standard menus, headers, footers for our theme. Let's begin with menus.

Step: Where is the Menu?

Activate the **TwentySeventeen** theme and go to **Appearance**. You will see the following:



Now activate **My First Theme** and select **Appearance**.



Q: What is the difference?

A: **Widgets** and **Menus** options are missing (among others). Why? Because menus and widgets - and other WordPress functions - need to be registered in functions.php order to be activated. This is done to minimize the overhead of your site. Only functionality that is used by your site is activated by WordPress.

Step: Create and Register a Menu

Activate **My Second Theme** and then add the following code to the file `functions.php`. just below your thumbnail code and save.

```
register_nav_menu('main-menu', 'Main Menu');
```

Now when you click on **Appearance** the menu option will be visible. Create a menu called primary.

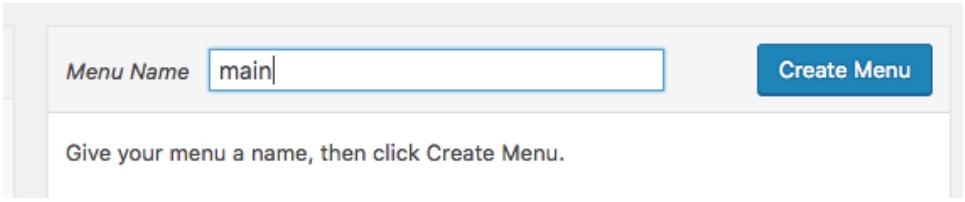
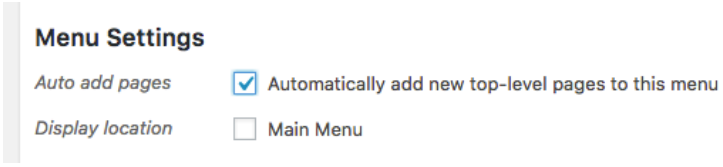


Figure: Add a menu name and then select Create Menu



Q: What happens when you change the functions.php code as follows:

```
register_nav_menu('main-menu', 'Main Menu');
```

A: The display location label changes on the Menu page.

Note that the menu isn't active yet. We'll see how that's done in our next exercise.

Step: header.php

Create a file called header.php and give it the following code. Copy it to `mysecondtheme`.

```
<!doctype html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
	<title><?php bloginfo('name'); ?> | <?php bloginfo('description');
?></title>
	<link href="<?php bloginfo('stylesheet_url'); ?>" rel="stylesheet"
type="text/css">
	<?php wp_head(); ?>
</head>
<body>
	<div class="wrapper">
		<header>
			<h1><a href="<?php echo home_url('/'); ?>"><?php
bloginfo('name'); ?></a></h1>
			<h2><?php bloginfo('description'); ?></h2>

			<?php $main_menu_top = array(
					'theme_location' => 'main-menu',
					'container' => 'nav'
				);
			?>
			<?php wp_nav_menu($main_menu_top); ?>

		</header>
```

Step: Create a file called footer.php and give it the following code:

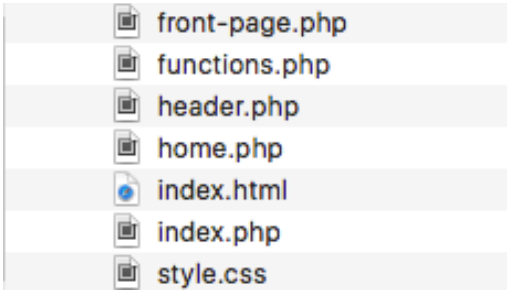
```
	<footer>
		<p class="copyright">&copy; <?php the_date(Y); ?></p>
		<p class="webdesigner">website by <?php the_author(); ?></p>
		</footer>
	</div> <!-- end wrapper -->
</body>
</html>
```

Step: Edit index.php

Edit index.php by replacing the top part of the file with the php function get_header() and the bottom part with the php function get_footer().

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

        <div class="posts">

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

            <article class="post">
                <header>
                    <h2 class="entry-title">
                        <a href="<?php the_permalink();?> ">
                            <? the_title(); ?>
                        </a>
                    </h2>
                </header>
                <?php the_content() ?>

                <p class="entry-meta">Posted on <?php the_date(); ?> by
                    <span class="author"><?php the_author(); ?></span>
                </p>

            </article>
            <?php endwhile; else: ?>
                <p><?php _e('Sorry, no posts matched your criteria.'); ?></p>
            <?php endif; ?> <!-- End of posts -->
        </div> <!-- end content -->

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

Go to your local WP files. Notice that there are a number of new files, header.php, functions.php, front-page.php and home.php.



Let's talk about `header.php` first. `Header.php` is a generic header that can be reused by all of your pages. It is called by the function `get_header()` which you will see that the top of index.php, front-page.php and home.php.

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

```php
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title><?php bloginfo('name'); ?> | <?php bloginfo('description'); ?></title>
    <link href="<?php bloginfo('stylesheet_url'); ?>" rel="stylesheet" type="text/css">
    <?php wp_head(); ?>
</head>
<body>
    <div class="wrapper">
        <header>
            <h1><a href="<?php echo home_url('/'); ?>"><?php bloginfo('name'); ?></a></h1>
            <h2><?php bloginfo('description'); ?></h2>

            <?php $main_menu_top = array(
                    'theme_location' => 'main-menu',
                    'container' => 'nav'
                );
            ?>
            <?php wp_nav_menu($main_menu_top); ?>

        </header>
```

Figure: the contents of header.php

Q: What do you notice about header.php?
A: Incomplete html, body, div.wrapper and html elements are not closed.

The Template Hierarchy

WordPress has a complicated set of context sensitive rules which determine how content is displayed. Index.php is the default display template. Three of the most important other templates are `front-page.php`, which is used to format the landing page for a website, and `home.php` which is used to format blogs, and `single.php` which is used to display posts.

Step 1

Load **mythirdtemplate.zip** into WordPress. You will notice two new files, `front-page.php` and `home.php`

`Front-page.php` and `home.php` are edited versions of `index.php` Both files will use the loop to display different content. They are activated in different contexts.

Let's edit **front-page.php** first.

For the front page we're not going to use the title for the post, just the content.

```
<div class="front-page">

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

    <div class="intro">
        <?php the_content(); ?>
    </div>

    <?php endwhile; else: ?>

    <p>
        <?php _e('Sorry, no posts matched your criteria.'); ?>
    </p>
    <?php endif; ?> <!--End of posts -->
</div> <!-- end content -->
```

I want to make sure only the Featured posts appear on the front page, so I'm going to make a custom query using **WP_Query()**. I'm going to create an argument for the query as a variable and save the result in another variable:

Step 2

```
<?php
    $args = array('category_name' => 'featured');
    $featured = new WP_Query($args);
?>
```

and feed that variable into my content loop:

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

Finally, I'm going to take out the **else** statement. If there are no Featured posts, well, I'm going to have to make some to fulfill my design. I also need to add the function **wp_reset_postdata()** to reset my query for later loops.

```
    <?php endwhile;  ?>
    <?php endif; wp_reset_postdata(); ?> <!--End of posts -->
</div> <!-- end content -->
```

The final result should look like the following:

```php
<?php get_header(); ?>
        <div class="front-page">
            $args = array(
                "category_name" => "featured"
            );
            $featured = new WP_Query($args);
            <?php if(have_posts()) : while($featured->have_posts()) :
                the_post(); ?>
            <div class="intro">
                <?php the_content(); ?>
            </div>
            <?php endwhile; ?>
            <?php endif; wp_reset_postdata(); ?> <!-- End of posts -->
        </div> <!-- end content -->
        <div><span>This page uses front-page.php</span></div>

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

View your site. Your main page should now use `front-page.php` not `index.php`.

Step 4: Modifying home.php for blog posts

Now modify `home.php` so that it looks like the following:

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

        <div class="posts">

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

            <article class="post">
                <header>
                    <h2 class="entry-title">
                        <a href="<?php the_permalink();?> ">
                            <? the_title(); ?>
                        </a>
                    </h2>
                </header>
                <?php the_content() ?>

                <p class="entry-meta">Posted on <?php the_date(); ?> by
                    <span class="author"><?php the_author(); ?></span>
                </p>

            </article>
            <?php endwhile; else: ?>
                <p><?php _e('Sorry, no posts matched your criteria.'); ?></p>
            <?php endif; ?> <!-- End of posts -->
        </div> <!-- end content -->
        <div>This page uses home.php</div>

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

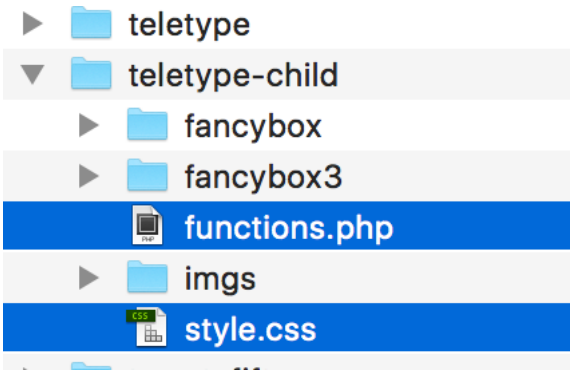
Create a Child Theme

"A child theme is a theme that inherits the functionality and styling of another theme, called the parent theme. Child themes are the recommended way of modifying an existing theme."

https://codex.wordpress.org/Child_Themes

Steps

1. Create a new folder in the themes folder
2. Give the folder a representative name, for example [parent-theme-name]_child



3. Create a file called style.css and place it in the child theme folder. Style.css for child themes is similar to style.css for main themes, with the following exception: it must have a Template: line, which contains the parent theme name.

```
/*
Theme Name: Teletype-child
Theme URI: http://dinevthemes.com/themes/teletype/
Author: Brian MacMillan
Author URI: http://brianmacmillan.com/
Template: teletype
Description: Teletype is a minimalist theme with masonry
Version: 1.1.7
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Text Domain: teletype
Tags: blog, photography, portfolio, two-columns, left-si
*/
```

Figure: A child theme of Teletype must contain a Template row, pointing to its parent theme

4. Create an empty file called functions.php, which will contain functions for your child theme, and a file called screenshot.png, which will be the image displayed on the activate theme menu in the WordPress dashboard.

Note that when you create functions for the child theme you need to use the function `get_stylesheet_directory()` to determine paths. The function `get_template_directory()` returns the path to the parent theme's folder.

Outtakes

```
1     register_nav_menus( array (
2                         "main-menu" => "Main Menu",
3                         "footer-menu" => "Footer Menu"
4               )
5     );
```