CREATING CUSTOM THEMES

BEFORE WE BEGIN

Creating WordPress themes involves a lot of trial and error.

RESOURCES TO HELP YOU

- WordPress Codex (learn to love it!!!)
- StackOverflow (I do not recommend asking questions here until you've lurked quite a bit. Use it to see if your question has already been answered.)

ACTIVITY: CREATE A NEW THEME

1. Inside `C:/Program Files/XAMPP/htdocs` find the sait-wp folder you created yesterday.
2. Inside `sait-wp` go to wp-content/themes.
3. Create a new directory for your theme. Call it sait.

ACTIVITY: ADD STYLE.CSS

A WordPress theme only **needs** 2 files to exist. `style.css` and `index.php`.

1. In your sait theme folder, create `style.css`. This must be in the root folder of your theme.
2. Add the following code so that WordPress knows a theme exists here.

```
/*
Theme Name: SAIT
Author: Your Name
Description: Minimal theme created for learning purposes
Version: 0.0.1
Tags: minimalist
*/
```

ACTIVITY: TURN HTML INTO PHP

To speed things up, we're going to use Bootstrap!

1. Go to Brightspace and download the sait-theme.zip file found under In-Class Exercises > Day 2. Inside this folder is `index.html` and `css/blog.css`.
2. Rename `index.html` to `index.php`.
3. Take a look through this file and make sure you understand what's going on. This is a static version of a blog.

ACTIVITY: THEME CREATED

Your theme has now been created. Let's check it out.

1. Go back to the WordPress dashboard.
2. Click on Appearance > Themes.
3. You should see SAIT as a theme. Activate it.
4. Now go back to localhost/sait-wp. What do you see?

SOME NOTES

- We've created a custom theme, but it doesn't do much since it's still basically just static HTML.
- blog.css isn't being loaded.

```
<link href="css/blog.css" rel="stylesheet">
```

- This tries to look at `localhost/sait-wp/css/blog.css` but css/blog.css is actually inside `wp-content/themes/sait`.

Note To link to anything on a WordPress page, you'll need some PHP.

ACTIVITY: FIX THE BROKEN CSS LINK

1. Find this code in your index.php file.

```
<link href="css/blog.css" rel="stylesheet">
```

2. Tell it to dynamically link to the themes folder instead.

```
<link href="<?php echo get_template_directory_uri() ?>/css/blog.css"
rel="stylesheet">
```

3. Reload the page. Your blog.css file should now load.

Note: While this works, it's not the proper way to load stylesheets. I'll show you the proper way later.

INDEX.PHP

Right now, everything is in index.php. But we want the header, footer, sidebar, etc. on all the pages to be the same.

To avoid having to create every page with a header, footer, etc., we're going to divide index.php into 4 sections.

- header
- footer
- sidebar - secondary information
- content - main content and articles

ACTIVITY: DIVIDING INDEX.PHP

- Create a `header.php` file. Move everything from `<!DOCTYPE html>` through the main blog header (lines 1-34ish) to header.php.
- Create a `footer.php` file. Move everything from `</div><!-- /.container -->` and down to footer.php.
- Create a `sidebar.php` file. Move all sidebar content to sidebar.php. (lines 101-123)
- Create a `content.php` file. Move all the articles and main content to content.php. (lines 40-97)
- Refresh your page.

INDEX.PHP

The index file should be pretty sparse now. Here's all we have left:

```
<div class="row">
    <div class="col-sm-8 blog-main">
    </div> <!-- /.blog-main -->
</div> <!-- /.row -->
```

Now, we're going to add everything back in.

ADDING THE HEADER INFORMATION

- In `index.php` we pull in the header information by using:

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

- In `header.php` we fire the `wp_head` action which is used to add elements like styles, scripts, and meta tags. It goes directly before the closing `</head>` tag.

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

ACTIVITY: ADD A HEADER

1. In `index.php` add the following code to the top of the page.

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

2. In `header.php` add the following code right before the closing `</head>` tag.

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

3. Refresh your page.

ADDING THE FOOTER INFORMATION

- In `index.php` we pull in the footer information by using:

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

- In `footer.php` we fire the `wp_footer` action which is used to add elements like JavaScript files. It goes directly before the closing `</body>` tag.

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

ACTIVITY: ADD A FOOTER

1. In `index.php` add the following code to the bottom of the page.

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

2. In `footer.php` add the following code right before the closing `</body>` tag.

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

3. Refresh your page.

ADDING CONTENT

- In `index.php` we pull in the content information by using:

```
<?php get_template_part( $slug, $name ); ?>
```

- $slug - the slug name for the generic template
- $name - the name of the specialized template (optional parameter)

ADDING CONTENT

For our site, we're going to use:

```
<?php get_template_part( 'content', get_post_format() ); ?>
```

- This tells WordPress to find a template part called content (`content.php`) and adds it here.
- `get_post_format()` returns the post format of the post (aside, gallery, link, image, etc.)

ACTIVITY: ADD CONTENT

1. In `index.php` add the following code to the main section of the page (right after `<div class="col-sm-8 blog-main">`)

```
<?php get_template_part( 'content', get_post_format() ); ?>
```

2. Refresh your page.

ADDING A SIDEBAR

Adding a sidebar is as simple as adding a header and a footer. It's built in!

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

ACTIVITY: ADD A SIDEBAR

1. In `index.php` add the following code to the sidebar area (right after `</div> <!-- /.blog-main -->`).

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

2. Refresh your page.

WE DID IT

We successfully chopped up our index.php into reusable files that we can use across our site.

If you're logged in to WordPress, you'll also see a top bar now.

ADDING MORE WORDPRESS CODE

Right now, the title and more are all hardcoded in HTML, but we want to be able to change this in WordPress.

Handy WordPress Code

- `get_bloginfo('name');` - site title
- `get_bloginfo('description');` - site tagline
- `get_bloginfo('wpurl');` - WP address

ACTIVITY: USE WORDPRESS TO CHANGE INFO

- In `header.php`, change the contents of the title tag and h1 tag to

```
<?php echo get_bloginfo( 'name' ); ?>
```

- Change the description to

```
<?php echo get_bloginfo( 'description' ); ?>
```

- Wrap the h1 tag in a link. Set the href to

```
<?php echo get_bloginfo( 'wpurl' ); ?>
```

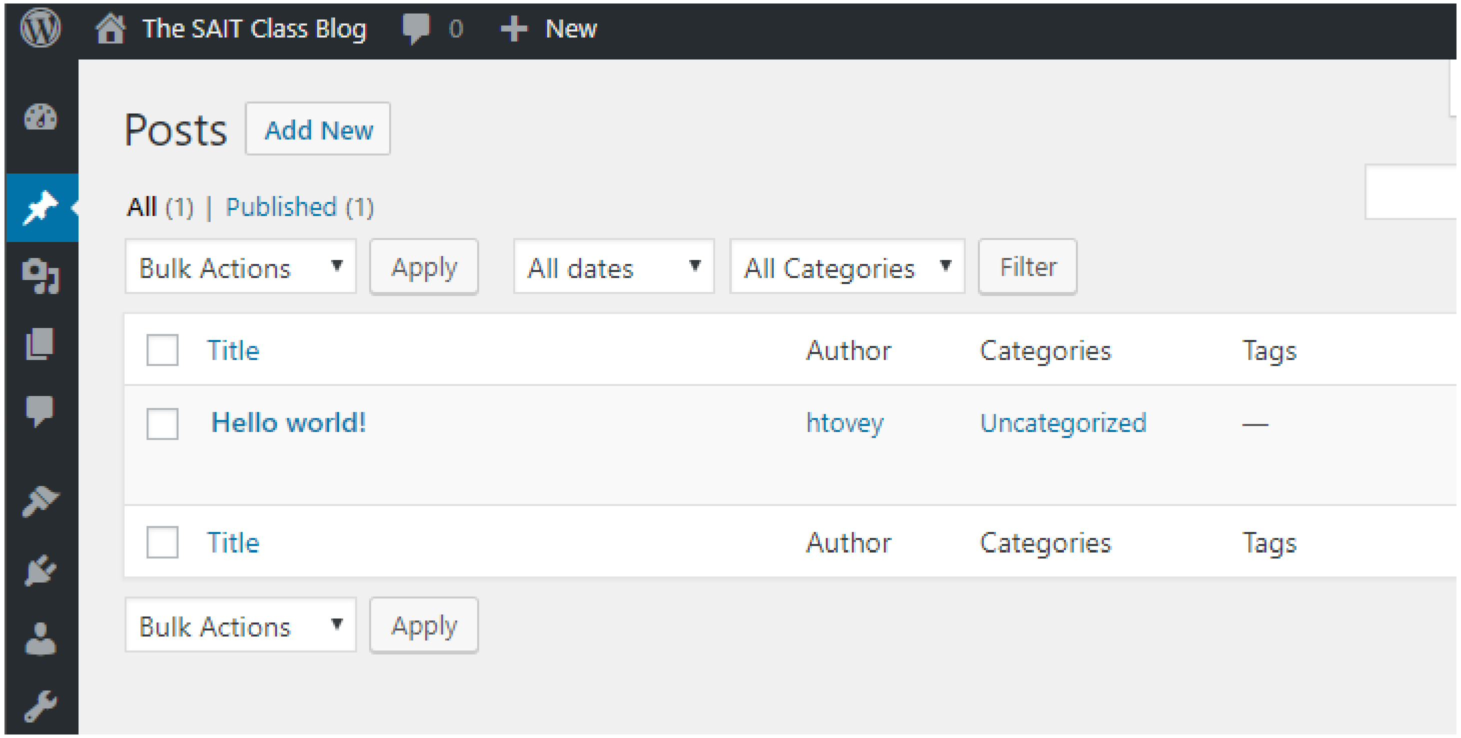
- In your WordPress Dashboard, go to Settings > General to change the site title and tagline.
- Refresh!

THE LOOP!

SHOWING BLOG POSTS ON YOUR SITE

If you look in your Dashboard, you'll find a post called "Hello world!".

Our job is to make sure we display this post on our site. Right now, it's missing!



	Title	Author	Categories	Tags
☐	Hello world!	htovey	Uncategorized	—
☐	Title	Author	Categories	Tags

THE LOOP

This is the code to create The Loop in WordPress:

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

<!-- contents of the loop -->

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

- **IF** there are posts, **WHILE** there are posts, **DISPLAY** the post.
- Anything inside the loop will be repeated (like a while loop).
- "contents of the loop" is where we'll put the code we want it to run every time the loop repeats.

ACTIVITY: ADD THE LOOP!

1. In `index.php` replace

```
<?php get_template_part( 'content', get_post_format() ); ?>
```

with

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

    get_template_part( 'content', get_post_format() );

endwhile; endif;
?>
```

INSIDE THE LOOP

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

    get_template_part( 'content', get_post_format() );

endwhile; endif;
?>
```

The only thing inside the loop is content.php, which will contain the contents of one single post.

So let's edit `content.php` so that instead of multiple blog posts, it only stores the template for blog posts.

ACTIVITY: EDIT CONTENT.PHP

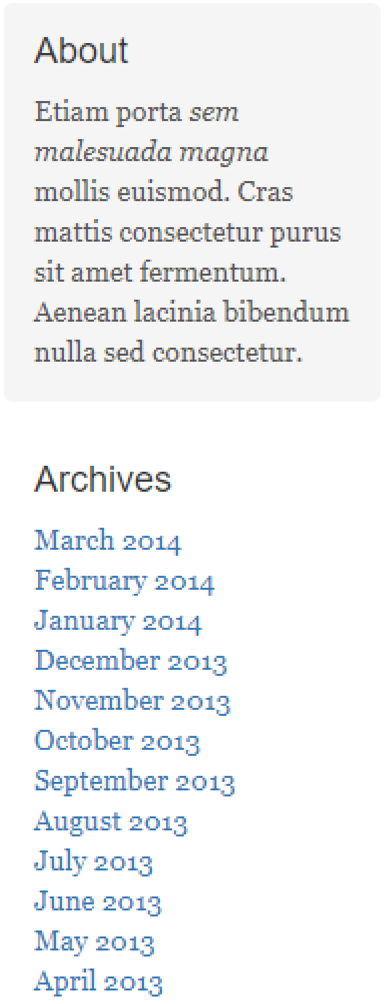
1. Open `content.php` and delete everything except the first blog post.
2. Replace the title with `<?php the_title(); ?>`
3. Replace the blog post date with `<?php the_date(); ?>`
4. Replace the author with `<?php the_author(); ?>`
5. Replace all other content with `<?php the_content(); ?>`
6. Refresh the page. You should see the Hello World post.
7. Create a new blog post to make sure your loop is working.

MAKE THE SIDEBAR DYNAMIC

Right now, we have hardcoded archives and user description. But let's make it dynamic instead!!

```
<!-- gets a list of the archives (monthly, yearly, daily,
weekly, etc.)-->
<?php wp_get_archives( 'type=monthly' ); ?>
```

```
<!-- gets description about the author -->
<?php the_author_meta( 'description' ); ?>
```


About

Etiam porta *sem malesuada magna* mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.

Archives

March 2014
February 2014
January 2014
December 2013
November 2013
October 2013
September 2013
August 2013
July 2013
June 2013
May 2013
April 2013

ACTIVITY: MAKE THE SIDEBAR DYNAMIC

1. In `sidebar.php`, replace the latin placeholder text under About with

```
<?php the_author_meta( 'description' ); ?>
```

2. Replace the list items `<li>` of dates with

```
<?php wp_get_archives( 'type=monthly' ); ?>
```

3. In WordPress, go to Users, click on yourself, and edit your Biographical Info.
4. Refresh the front page.

MENUS AND PAGES

ACTIVITY: ADD A PAGE

1. In the WordPress Dashboard, add an About Page. You can add any content you want.

EDIT THE NAVBAR

We can use the following code to add the navbar where we want it. This allows us to use WordPress to decide what pages show up on the menu.

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

ACTIVITY: ADD A MENU

1. In `header.php`, remove everything between your nav tags.
2. Replace the list with this code:

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

3. Refresh your page. You should now see a menu with About and Sample Page on it.

SOME NOTES

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

This code will grab the first menu it sees, or create a default menu, which isn't always what you want. Later, we'll cover how to add support for specific menus to your custom theme.

PAGES

PAGE.PHP

While `index.php` is used as the blog index page, we can create a `page.php` to use for pages instead.

ACTIVITY: CREATE PAGE.PHP

1. Duplicate your `index.php` file and name the new file `page.php`.
2. In `page.php`, remove this code:

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

3. Change `col-sm-8` to `col-sm-12` since we're moving from 8 columns wide to the full 12 columns wide after getting rid of the sidebar.
4. Click on About or Sample Page on your WordPress site. You should now see a difference!

INDIVIDUAL POST PAGES

CONTENT.PHP

Before we do anything else, we need to be able to access each post individually. Right now, we can only see a list of the posts but there are no links to take us to the single page.

We can add permalinks using

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

This will give us the permalink for each post.

ACTIVITY: ADD PERMALINKS TO CONTENT.PHP

1. In `content.php`, add a link `<a>` around the title.
2. Set the href value to

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

3. Refresh the homepage.

SINGLE.PHP

We can now click on the blog post links to take us to the single blog post, but we want to style this page a bit differently.

We need to:

- create `single.php` so that we can remove the sidebar.
- create `content-single.php` so that we can remove the permalink we just added to `content.php`.

ACTIVITY: CREATE AND EDIT SINGLE.PHP

1. Duplicate `page.php` since we already removed the sidebar there.
2. Name this new file `single.php`.
3. Refresh your single post. You should now see no sidebar.
4. Duplicate `content.php`. Name this new file `content-single.php`.
5. Remove the permalink around the title.
6. In `single.php`, replace

   ```
   get_template_part( 'content', get_post_format() );
   ```

   with

   ```
   get_template_part( 'content-single', get_post_format() );
   ```

7. Refresh your single post again.

WHAT JUST HAPPENED

If we look at our files, we'll see that `index.php` (the homepage blog list) pulls in `content.php`.

`single.php` pulls in `content-single.php` instead.

This allows us to style individual posts separately from the list of posts.

ANOTHER CONTENT.PHP CHANGE

Often, you want to display an exerpt of the posts instead of the full blog post on the main blog page. To do this, we use

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

instead of pulling in the entire content.

This will show only the first 55 words of the post.

ACTIVITY: EXCERPTS

1. In `content.php`, replace

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

   with

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

2. Make sure at least one of your blog posts has more than 55 words.
3. Refresh your homepage. You should now see [...] at the end of one of the posts.

PAGINATION

ACTIVITY: PAGINATION

By default, 10 posts will show up on a page before you can navigate to more posts.

- For testing purposes, go to Settings > Reading and change **Blog pages show at most** to 1.
- Save your changes and refresh the homepage.

PAGINATION

Now we have only 1 post on the homepage, but we have no way to navigate to previous posts!

We can add pagination with

```
<?php next_posts_link( 'Older posts' ); ?>
<?php previous_posts_link( 'Newer posts' ); ?>
```

BACK TO THE LOOP

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

    get_template_part( 'content', get_post_format() );

endwhile; endif;
?>
```

IF there are posts, **WHILE** there are posts, **DISPLAY** the post.

- If we want to add pagination, we don't want to add links while it's looping because we don't want pagination links for every blog post on the page. We can't add pagination links before `endwhile;`.
- Instead, we'll add pagination just before the `endif;` statement. This adds pagination links **IF** there are posts.

ACTIVITY: ADD PAGINATION

- In `index.php`, add the following code between `endwhile;` and `endif;`:

```
<nav>
    <ul class="pager">
        <li><?php next_posts_link( 'Previous' ); ?></li>
        <li><?php previous_posts_link( 'Next' ); ?></li>
    </ul>
</nav>
```

- Make sure to close the php tag after `endwhile;` and open again before `endif;` or you'll get a syntax error.
- Refresh your page!

COMMENTS

ADDING COMMENTS

First, decide where you want comments to show up. Typically, you see comments on single blog post pages. So we're going to work with `single.php` to add comments.

Then, we can paste the following code where we want comments to show up.

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

This checks to see if comments are open OR if there are already comments that have been made.

ACTIVITY: ADD COMMENTS TO SINGLE.PHP

1. In `single.php`, add the following code under `get_template_part();`

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

You should now be able to see comments on your posts.

COMMENTS

You can do a lot more with comments by creating `comments.php`.

For now, we're going to leave comments alone since you have all you need for basic comments.

But if you'd like to customize them further, I recommend using this comments.php template as a starting point.

FUNCTIONS.PHP

FUNCTIONS.PHP

- Add functionality and change defaults throughout WordPress.
- Plugins and custom functions are basically the same. Any code you create can be made into a plugin, and vice versa.
- The main difference is that anything you place in your theme's functions is only applied while that theme is actively selected.

BASICS OF FUNCTIONS.PHP

`functions.php` can seem a little complicated at first, but it's mostly made up of a bunch of code blocks that look similar to this:

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

1. Create the function.
2. Add the action. 1st parameter is the name of the action you're hooking your function to. 2nd parameter is the name of your function.

Here's a list of actions which are used in WordPress.

ACTIVITY: CREATE FUNCTIONS.PHP

1. Create a functions.php file and place it in your theme directory.
2. Add a php opening tag to the top of the file. You don't need a closing tag since this is a pure PHP file.

```
<?php
```

ENQUEUE SCRIPTS AND STYLESHEETS

Earlier, we incorrectly linked to CSS and JavaScript in the header.php and footer.php files.

In WordPress, you should add styles and scripts in `functions.php`.

ENQUEUE FUNCTION

```php
// Add scripts and stylesheets
function sait_scripts() {
    wp_enqueue_style( 'bootstrap.css', 'bootstrap-link', array(), '3.3.5' );
    wp_enqueue_style( 'blog', get_template_directory_uri() . '/css/blog.css' );
    wp_enqueue_script( 'bootstrap', 'bootstrap.js', array( 'jquery' ), '3.3.5', true );
}

add_action( 'wp_enqueue_scripts', 'sait_scripts' );
```

1. Create a function named after our theme.
2.
```
wp_enqueue_style(stylesheetName, src, dependenciesArray, version, media);
```
3.
```
wp_enqueue_script(scriptName, src, dependenciesArray, version, inFooter);
```

For these to be properly inserted, `wp_head()` must be in your head tag in `header.php` and `wp_footer()` must be in your `footer.php`.

Note: Don't copy this code directly. It's been shortened so it all fits on the page. Replace bootstrap.css and bootstrap.js with the actual links found in your `header.php` and `footer.php` files.

ACTIVITY: ENQUEUE STYLES AND SCRIPTS

- Add the following code to `functions.php`

```
// Add scripts and stylesheets
function sait_scripts() {
    wp_enqueue_style( 'bootstrap',
'https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css', array(),
'3.3.5' );
    wp_enqueue_style( 'blog', get_template_directory_uri() . '/css/blog.css' );
    wp_enqueue_script( 'bootstrap',
'https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js', array(
'jquery' ), '3.3.5', true );
}

add_action( 'wp_enqueue_scripts', 'sait_scripts' );
```

- Remove the links to bootstrap.min.css, blog.css, jquery, and bootstrap.min.js in `header.php` and `footer.php`.
- Refresh your page.

WORDPRESS TITLE

Before, we used the following code to add a title tag in `header.php`.

```
<?php echo get_bloginfo( 'name' ); ?>
```

But this means that the title tag will be the same for every page. Boooooo!

WordPress will take care of changing the title for us if we let it using:

```
add_theme_support( 'title-tag' );
```

ACTIVITY: CHANGING THE TITLE

1. In `header.php`, remove the title tag entirely.
2. In `functions.php`, add this code block:

```
add_theme_support( 'title-tag' );
```

3. Refresh the page and visit other pages.

BACK TO MENUS

MENUS

Earlier, we created a very basic menu, but often, themes have:

- primary menu (often found in the header)
- secondary menu (often found in the footer)
- social media links

WP_NAV_MENU();

To specifically add support for specific menus and put them where you want in your theme, you need to register them first!

Single Menu

```
function register_my_menu() {
    register_nav_menu('new-menu',__( 'New Menu' ));
}
add_action( 'init', 'register_my_menu' );
```

Multiple Menus

```
function register_my_menus() {
    register_nav_menus(
      array(
        'new-menu' => __( 'New Menu' ),
        'another-menu' => __( 'Another Menu' ),
        'an-extra-menu' => __( 'An Extra Menu' )
      )
    );
}
add_action( 'init', 'register_my_menus' );
```

ACTIVITY: ADD MENUS

1. Register 2 menus for your WordPress theme.
   - Primary Menu (primary-menu)
   - Social Media Menu (social-media-menu)
2. In your WordPress dashboard, go to Appearance > Menus. This is a new tab that has appeared because you registered menus.
3. Create a primary menu with Home, About, and Sample Page.
4. Create a social media menu with links to Twitter, Facebook, and Github.
5. Make sure to specify the Display Location for each menu.

MENUS, CONT.

Now that you have your menus built, you need to do something with them!

```php
<?php wp_nav_menu( array( 'theme_location' => 'new-menu' ) ); ?>
```

This tells WordPress to add a menu like before, but now we're specifying which menu we want to appear in a specific location.

ACTIVITY: ADD MENUS TO THEME

1. In 'header.php', replace

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

with

```
<?php wp_nav_menu( array( 'theme_location' => 'primary-menu' ) ); ?>
```

2. In 'sidebar.php', replace

```
<ol class="list-unstyled">
 <li><a href="#">GitHub</a></li>
 <li><a href="#">Twitter</a></li>
 <li><a href="#">Facebook</a></li>
</ol>
```

with

```
<?php wp_nav_menu( array( 'theme_location' => 'social-media-menu' ) ); ?>
```

WORDPRESS, CONT.

There's far far far more that you can do with WordPress. Today, you learned how to take a simple HTML page and turn it into a theme.

Here are some resources to help you on your WordPress learning path:

- 10 Useful Code Snippets for WordPress Users
- Shun the Plugin: 100 WordPress Code Snippets from Across the Net
- wp-snippet.com

STARTER THEMES

To save time, WordPress developers often create or use starter themes. These themes will have a lot of their most often used code already set up and ready to go.

In this class, we'll be using the Naked Theme.

This is a heavily documented theme that helps new WordPress develoeprs understand the files a bit better.

ACTIVITY: USE NAKED THEME

1. Download the Naked Theme
2. Install it on your local WordPress installation.
3. Activate the theme in WordPress and start reading through the files.
4. Familiarize yourself with this theme and try customizing it. You will be using this theme in later assignments.