VCD-3600

WEB DESIGN II

TIPS & TRICKS: Wordpress Themes, Part 3

Modular Template Elements

In even a basic custom theme we might have several different template files such as `index.php`, `category.php`, `page.php`, and `single.php`. Since these all have different content, the separate pages are necessary. However, they might all have some common elements in a persistent header and footer. WordPress allows us to separate these elements into standalone files and then link them into the other main template files. Consider this proces:

1. Open a file such as `index.php` that contains your full page structure, including the masthead and footer that is persistent throughout your site.
2. Select and copy the content in this file that represents the persistent header. This can contain the DOCTYPE, opening `<html>` tag and the entire `<head>` as well as content that opens the `<body>` tag.
3. Create a new file called `header.php` inside your theme's root folder. Paste the header content you copied from `index.php` into this file.
4. Still in `header.php` look in the code you just pasted for the closing `</head>` tag and right before it add the following: `<?php wp_head(); ?>`
5. Switch back to index.php and remove the content you had copied into header.php. In its place, add the following: `<?php get_header(); ?>`
6. Open each of your other template files and remove the very same set of header content, replacing it with the call to `get_header()`.

Now we'll repeat this process for the footer content.

7. Back in `index.php` select and copy all the persistent footer content.
8. Create a new file called `footer.php` inside your theme's root folder. Paste the footer content you copied from `index.php` into this file.
9. Still in `footer.php` look in the code you just pasted for the closing `</body>` tag and right before it add the following: `<?php wp_footer(); ?>`
10. Switch back to `index.php` and remove the content you had copied into `header.php`. In its place, add the following: `<?php get_footer(); ?>`
11. Open each of your other template files and remove the very same set of footer content, replacing it with the call to `get_footer()`.

If applicable, you can repeate the same process for persistent sidebar content by creating `sidebar.php` and replacing any such content in template files with a call to `get_sidebar()`.

Further Customizations

A whole slew of additional features are available to further customize your WordPress theme to fit the needs of the site your editing.

Creating A Theme Functions File

Many of the further customizations depend on having a set of functions set up specifically for this theme.

1. So in your theme's root folder create `functions.php`.
2. In it, add an opening `<?php` and you're ready to go.

Working with WordPress Menus

In order to get menus from WordPress to show up in our theme we need to register them using a custom function. This involves setting up a "position" in your theme in which a menu can be placed, and then configuring a particular menu from WordPress to appear in that position.

First let's register a position and assign a menu to it.

1. Open functions.php in your theme folder.
2. Place the following inside the opening PHP tags:

```
function register_menus() {
    register_nav_menus(
        array( 'main-nav' => __( 'Main Nav' ) )
    );
}
add_action( 'init', 'register_menus' );
```

Now we have a position called "header-nav" defined and we've enabled a menu called "Header Nav" to be placed in it by WordPress. Both of these names can be confiured to whatever you'd like, but just be prepared to used the position name again in the next process.

Now we need to indicate where in our actual theme this position should reside.

3. Open the template file in which a menu position should be inserted, such as header.php
4. Identify and remove any static navigation markup you have in place such as a `<ul>`. Feel free to leave any helpful containers such as a `<nav>` or other `<div>` you've already used in your stylesheet.
5. In its place add the following:

```
<?php
    wp_nav_menu( array(
        'theme_location' => 'main-nav',
        'container' => false
    ));
?>
```

Now WordPress will recognize that your theme has a position in which a menu can be placed. So if you use the Appearance > Menus feature from the administration side of WordPress to configure a menu, just assign it to this position and you should see it appear in your site.

Showing Featured Images

A recent addition to the WordPress functionality is the ability to add a featured image to a post. You can enable this feature in your theme as follows:

1. Open `functions.php` and add the following near the bottom:
   `add_theme_support( 'post-thumbnails' );`

2. Now in any template page where you're showing posts that you also want a featured image to appear, add the following at the point in the markup where the image should appear:

   ```
   <?php if ( has_post_thumbnail() ) {
     the_post_thumbnail('large');
   } ?>
   ```

   This is set to show the "large" image but you also have the option to show other sizes. Learn more here.

A Few Miscellaneous Content Tags

Here are just a few others things you might find helpful:

- `body_class($class)` – displays all the classes that WordPress has generated for the page as a whole. This is typically placed inside the opening `<body>` and generates the full class attribute.
- `single_cat_title()` – displays a single category title for use on pages such as a category template page.
- `bloginfo('__value__')` – this function contains a series of helpful options that allow you to retrieve and display different information relative to your site. Consider the following options to use for `'__value__'`:
  - `'url'` – displays the full URL for your site
  - `'name'` – displays the name of your site as is configured from within WordPress
  - `'template_directory'` – displays the path to directory for the current page's template from within your theme. This is helpful for providing paths to images and other assets from within

such a template file, as was covered in WordPress Tips Part 1.

More information on this function can be found here.

Working with Custom or Guest Authors

WordPress allows a lot of flexibility including setting up custom fields on any post. A common such customization you might need is the ability to provide a custom or guest author for a post. First set up a custom field using the desidred post's Custom Field Widget. Be careful to provide a neat and easy to remember slug for the field such as `guest-author`. Then add the following code to your theme's functions.php file:

```
add_filter( 'the_author', 'guest_author_name' );
add_filter( 'get_the_author_display_name', 'guest_author_name' );
function guest_author_name( $name ) {
    global $post;
    $author = get_post_meta( $post->ID, 'guest-author', true );
    if ( $author )
        $name = $author;
    return $name;
}
```

Now any time you call `the_author()`, if the post has a `guest-author` custom field provided that name will appear instead of the user who created the post.