# CPSC 2030 – Web Development II

**Lab10: [60 marks] Creating WordPress Themes**

|  |
| --- |
| **Objectives** |
| * Create a new WordPress theme from scratch |
| **Preparation** |
| 1. Create a new folder named **Lab10** under your CPSC2030\Labs folder. 2. Download **Lab10StarterFiles.zip** and **wp\_tutorial.zip** from D2L and unzip them to the folder Lab10. |
| **What to do** |

The objective of this assignment is for you to create a new WordPress theme from scratch. You will use WordPress template files to fetch and display content from the database, explore a few different ways to organize your theme, style your theme, and add advanced functionality.

Before trying this lab assignment, you are recommended to complete the WordPress Theme tutorial that can be downloaded from D2L. Alternately, you could keep that tutorial close by as you work through this lab assignment and use it as a reference.

**Task 1: [15 marks] Setup**

Install a new Wordpress to htdocs (don’t delete your existing Wordpress installation) and rename the folder **wordpress-6.1** to **lab10**. Your second Wordpress site will be **localhost/lab10/wordpress**. Set the site title to **"Influential Figures in Computing"**.

Import the contents from the XML file included in the starter files. This content includes a set of posts, tags and categories around which you will structure your site.

You already have a theme activated. But in this lab, we're going to create our own theme, which will define the design and functionality of the site.

In the **wp\_content/themes** folder of your WP installation, create a new folder called **firstnamelastname** (with your own first and last name). To start the theme, create files called **index.php, header.php, footer.php,** and **style.css.** Your **style.css** file should contain the following:  
  
**/\***

**Theme Name: Firstname Lastname**

**Author: Firstname Lastname**

**Description: Mathematical science shows what is. It is the language of unseen relations between things. But to use and apply that language, we must be able fully to appreciate, to feel, to seize the unseen, the unconscious. -- Ada Lovelace**

**\*/**

Using **index.php, header.php** and **footer.php**, set up a basic configuration that does the following things: (use the tutorial as reference if you need help with this.)

* In the header, displays the name of the site with a link to the home page;
* Uses a <main> element for the main (non-footer and non-header) content of the site;
* Uses "the loop" to display the following for each post on the index page:
  + The post;
  + The title for the post with a link to the post;
  + The excerpt of the post (use a WP function called **the\_excerpt()** for this.)
  + A list of categories for the post;
  + All of the above should be contained inside a suitable HTML tag. (which one do you think is most appropriate for marking up a post?)

If all goes well, you should have a very basic WordPress site that allows you to see all the posts displayed on the homepage and visit each post individually, with the site title as a link back to the home.

**Hint:** If you're getting a white screen of death and you think some error messages might be helpful, find a file called **wp\_config.php** in the root of your site folder. On line 82, set **WP\_DEBUG** to **true**. However, after debugging, I recommend setting it back to false since it can actually cause some problems when set to true.

**Task 2: [5 marks] Template Hierarchy**

When you click on a category for a post, the WordPress application will take you to a page that lists all posts with that category on the site. This is called an **archive,** and there are several types of archives that can be displayed. (for tags, authors, dates, categories, etc.)

When accessing the category archive, WP will first look for the file **category.php**. If it doesn't find it, it will look for **archive.php**. Right now, since we don't have either of those files, it's falling back to **index.php**. Let's create a file called **archive.php** in the root of the theme folder that will be used to handle the archive page for both tags and categories.

Copy the contents of **index.php** and paste them into **archive.php.** Change the code so that all that is displayed for each post is the title of the post with a link to the full post. (so no excerpt, no categories.)

One last thing: beneath the header of the page but above the posts output by the loop, display the title of the archive by using the function **the\_archive\_title().** Think about which HTML tags should be used to mark up the title of the archive and the titles of the individual posts.

**Task 3: [5 marks] Functions.php**

It would be nice to add some images to our site; particularly, featured images for the posts. In order to do this, we'll have to add support for featured images to our theme. Create **functions.php** in the root of the theme folder, and create a function for setting up the theme. In your setup function, add support for post thumbnails. (See the tutorial for details on how to do this.)

Using the images included in the starter materials, set a featured image for each post. Now modify your **archive.php** file so that with each post, the featured image is also displayed. To do this, use the function **the\_post\_thumbnail().** The featured image should also act as a link to the full post.

**Task 4: [15 marks] Single Posts and Refactoring**

So far, we have two template files that display content in two different formats; **archive.php** is used to display the archive pages for categories and tags, and **index.php** is used for everything else. Neither of these templates display the full content of a post, so there's currently no way to read a full post.   
  
We could consult the WP template hierarchy to see which template file should be implemented for single posts. (Spoiler alert: it's **single.php.**) However, before we rush into doing that, compare the files **index.php** and **archive.php** side-by-side. You may notice that they actually look similar and contain a lot of the same code, in particular, the code for implementing the loop. Since this isn't very *DRY*, we might want to consider just using one file to implement the code for the loop, and use **template partials** to implement the stuff that is actually different. (that is, the stuff inside the loop.)

So, instead of making yet another copy of the loop within a new template file called single.php, we're going to do things a bit differently. However, let's first refactor our current site to implement a more efficient architecture:

* Create a new subfolder in your theme called **template-parts**. This is where the **template partials** will be stored.
* Inside this new folder, create two new files called **content-post.php** and **content-home.php**.
* In **index.php**, cut the code from the interior of the loop (everything except the call to **the\_post()** ) and paste it into **content-home.php.**
* Replace the code that you cut out of **index.php** with a call to the function **get\_template\_part().** This function will be used to fetch the code from the file **content-home.php** and insert it into the parent template. (It kind of works like the **include** function in PHP.) Figure out which parameters **get\_template\_part()** should take to make this work: <https://developer.wordpress.org/reference/functions/get_template_part/>
  + Hint: check the user-contributed notes in the above article.

If all goes well, your site should be working exactly the same as it was before, but using a slightly less repetitive architecture. Now I'm going to show you how to do the following:

* Set up the file **content-post.php** to display the content of the single post;
* Have the above file fetched from within **index.php** and inserted into the template when viewing single posts.

Start by adding this tester content to **content-post.php:**

**<?php ?>  
<p>Single posts are working!</p>**

Now, in **index.php**, write some code that does the following:

* If we're on the home page, conditionally include the **content-home.php** partial. **Hint:** the function **is\_home()** will help with this.
* If we're in a single post, conditionally include the **content-post.php** partial.
  + hint: the functions **is\_single()** will be helpful for this.
* **Optional challenge:** refactor your code so that the archive pages are generated without using **archive.php.** (you'll have to delete or rename archive.php to take it out of the template hierarchy.)

Once you have this working, remove the tester content in **content-post.php** and have the template display the following:

* The title of the post;
* The featured image of the post;
* The full content of the post;
* The tags associated with the post;
* A link to the previous and next posts in the blog with the prefixes "Previous" and "Next", respectively. (Figure out how to do this.)

**Task 5: [5 marks] Navigation Menus**

Using the tutorial as a guide, register two menus, and have them displayed in the header and footer. Set up a header menu with links to all four categories. The footer menu should have a link to the Langara home page, Wikipedia, and some other website of your choosing.

**Task 6: [15 marks] Styling**

Using the tutorial as a guide, enqueue the **style.css** file to be linked to your pages.

One thing that isn't mentioned in the tutorial is that we can sometimes have problems with the server caching old version of the stylesheet, which makes it impossible to see recent changes. To get around this, add a new version number automatically to the stylesheet every time the file is updated by replacing the **wp\_enqueue\_style** function calling with the following line:

**wp\_enqueue\_style('main-stylesheet', get\_template\_directory\_uri() . '/style.css',**

**array(), filemtime(get\_template\_directory() . '/style.css'), false);**

Using the tutorial, figure out how to embed classes in the <body> tag that reflect the current page being viewed. Use these classes to target different areas of the site in your CSS styles.  
  
Now, implement the following styles, using the screenshots as a guide:

* **Mobile:**
  + Each post should have a border;
  + The height of the images should be set to a fixed value, and the object-fit property should be used to properly scale the image inside its frame;
  + Feel free to be creative in choosing your own colors and fonts.
* **Desktop:**
  + The index and archive pages should display the posts in a three-column responsive layout.
  + On the index page, the categories displayed within each post should be arranged to the top right corner of the post, with a vertical line separating them.
  + The navigation menu items should be arranged horizontally;
  + In individual post pages, the featured image should be floated to the right;

|  |
| --- |
| **What to hand in** |
| Zip your **theme folder** and hand it in to D2L. (Double check to make sure you've followed all the instructions for naming your theme folder.) |
| **When to hand in** |
| By 11:59pm, Sunday, November 27, 2022 |