**CPSC 2030 Course Notes:**

**Creating WordPress Themes**

To create a WordPress theme, all we need are two files: style.css, and index.php. However, we can include many more files to make more interesting and versatile themes. This tutorial will go through the basic procedure of creating a theme from scratch.

This tutorial assumes that you are able to install WordPress, and that you have a basic knowledge of how to use WordPress to create and edit posts and pages.

Import the content from the file **wp-tutorial.xml** into your WordPress; this will serve as the test content for this tutorial.

**Task 1: Basics**

First, we will begin by creating a folder for our new theme.

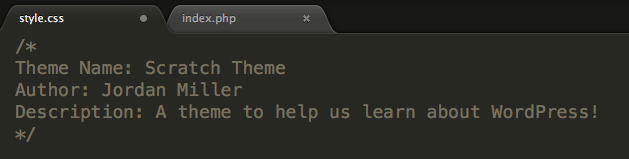
In the folder of your WordPress installation, navigate to **wp-content,** and then to **themes**; this is the folder where all the themes for your installation are stored. You will see a few folders in there for the themes that come pre-installed with WordPress.

**\*Tip: a good way to learn how themes work is to go into the folders for these pre-installed themes and open the files to see how they work.**

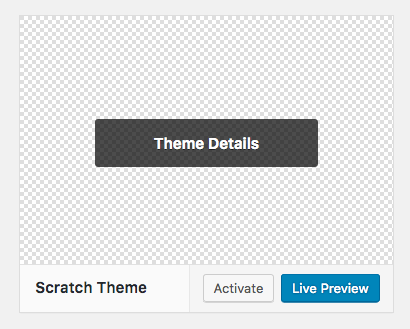
Create a new folder called **scratch**. This folder will contain all the files for the theme we will make from scratch.

Using Sublime or Atom, create two new files in the scratch folder: **style.css**, and **index.php**.

Let’s start with **style.css**. For now, this is just going to contain some basic information about the theme that will show up in the Dashboard themes section. Start by making a commented section, and enter the following information:



Even though this information appears to be commented out, WordPress will actually read it and use it to display information to users. There are other information fields we could enter, but this will be fine for now.

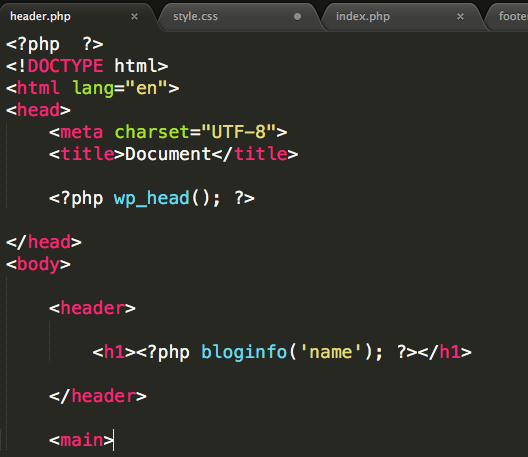
In your dashboard, you should see something like this:

**\*Tip: as you can see, our theme doesn’t have a screenshot to help the user know what it will look like when it has been activated. However, we can add any jpeg or png image to our scratch folder and rename it to *screenshot*, and this image will be used as the screenshot for our theme!**

**Task 2: Header and Footer**

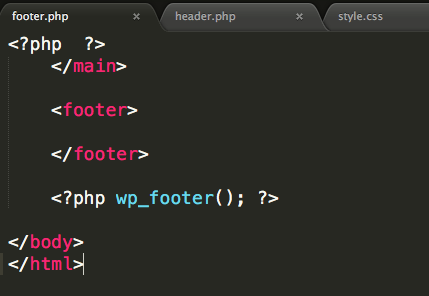
Many websites have the same header and footer on each page. Why not have separate files for the header and footer of each page? This is, in fact, what we will do. The strategy will be to create separate files for the header and footer, and have the **index.php** file add them to each page.

Create a file called **header.php** and enter the following information:



Notice that we have a start tags for the html, body and main elements, but no end tags. This looks like a mistake at first, but it’s fine; those tags will end in another file! Everything will be combined into a single, functional page in the end.

Create a file called **footer.php** and enter the following code:



There’s not much going on here, but we are closing off the HTML tags that we started in **header.php**. We’re also adding the function **wp\_footer(),** which will be necessary for a few things. (don’t worry about the purpose of this function for now!)

Now, we need to put these two files together somehow. We can do that using the **index.php** file that we created earlier. I mentioned before that this file is necessary for a WordPress theme; that’s because it’s the fallback file for generating any kind of page that our website can produce. We’re going to put some code into it that will combine the header and the footer:



As it is, this file is pure PHP, without any HTML. We’re using two WordPress PHP functions to get the header and the footer that we created. We don’t need to enter the names of the header and footer files, because we’ve used standard template file names.

**Task 3: Displaying Content**

So far, we have an HTML header and footer on our page, but no actual content between them. We’re going to need to somehow get the posts that have been created for the blog and output them on the page. To do this, we’re going to use something called **“The Loop”.**

More information about The Loop can be found here:

<https://codex.wordpress.org/The_Loop>

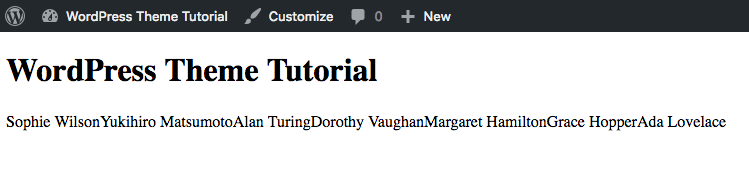
(Note: the codex uses a slightly different PHP syntax for The Loop than our syntax. Either syntax is fine to use in this class.)

Let’s make The Loop for the site in the **index.php** file, between the header and footer:



What’s happening here is the code is checking to see if there are any posts, and then looping through all the posts in the blog (using a while loop) and displaying the title of each one, by using **the\_title()** function.

Your page should now look something like this:



As you can see, the titles the posts are just being spit out one after another onto the page. We’d like to wrap them in some HTML to make them more semantically meaningful.



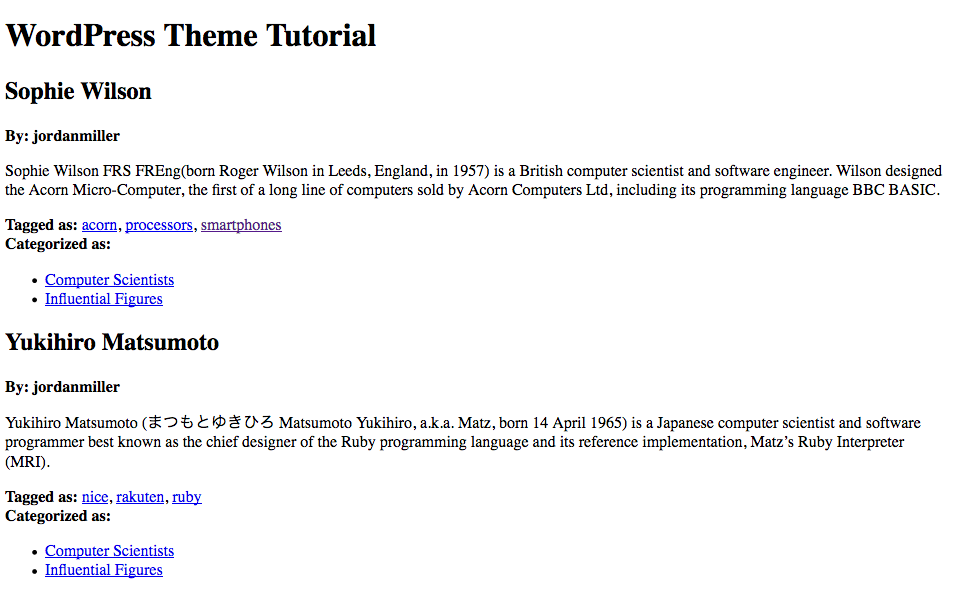
What we’re doing here is using some optional parameters for **the\_title()** function; we can actually specify some material to be inserted **before** and **after** the title of each post. In this case, we’ve decided to insert **h2** start and end tags, so that each title is in an **h2** element.

Now is a good time to experiment with adding different types of content to the page. Try this out:

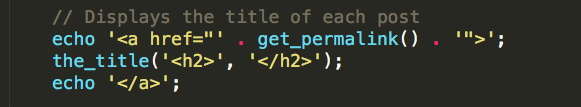


This code will add a few other things to each post that displays on the page, such as the author, an excerpt of the article, tags, and categories

Now, our website looks something like this:



This looks okay, but we’d like to be able to click on the title of each post to link to the full post. Let’s add **<a>** tags to our titles:



Note that for the **href**, we need to include the quotation marks that are necessary for the value of an attribute.

**\*Tip: the above method works because the function *get\_permalink()* does *not* automatically echo its value onto the page. Some functions do: for example, *the\_author().* If you include a function like *the\_author()* in an echo command, you will actually end up with the author on your page *twice.***

*Challenge: try to find a way to insert the* ***<a>*** *element* ***inside*** *the* ***<h2>*** *element by including it in the parameters passed into* ***the\_title()*** *function. You should be able to do this on one line of code.*