Wp Com Zac

Clean WordPress install, no starter theme. Following the tut from Udemy – Complete WordPress Development Themes and Plugins Course with Zac Gordon.

Bootstrap 4 used.

Thursday, 11 October 2018

[PHP for WordPress 4](#_Toc528417994)

[WordPress Complete with Zac Gordon 4](#_Toc528417995)

[The Loop 5](#_Toc528417996)

[Template Tags 5](#_Toc528417997)

[Conditional Tags 6](#_Toc528417998)

[Learn PHP for WordPress with Rachel McCollin 6](#_Toc528417999)

[Naming Conventions and Cases 6](#_Toc528418000)

[Setup Debugging 7](#_Toc528418001)

[Debugging Plugins 7](#_Toc528418002)

[Setup Test Demo Content & Install ‘Custom Post Type UI’ Plugin 8](#_Toc528418003)

[Wptest Demo Content 8](#_Toc528418004)

[‘Custom Post Type UI’ Plugin 8](#_Toc528418005)

[‘Advanced Custom Fields’ Plugin 8](#_Toc528418006)

[Create the 3 Main WP Files 8](#_Toc528418007)

[style.css 9](#_Toc528418008)

[functions.php 9](#_Toc528418009)

[Enqueuing CSS and JS 9](#_Toc528418010)

[index.php 10](#_Toc528418011)

[Setup Header and Footer Template Files 10](#_Toc528418012)

[header.php 10](#_Toc528418013)

[Custom Headers 10](#_Toc528418014)

[footer.php 10](#_Toc528418015)

[Setup Menus and body\_class(); 10](#_Toc528418016)

[Menus 10](#_Toc528418017)

[1. Register the menu with register\_nav\_menus() 10](#_Toc528418018)

[2. Setup the Nav Menu with wp\_nav\_menu() 11](#_Toc528418019)

[3. Define the Menu in Admin > Menus 12](#_Toc528418020)

[body\_class(); 12](#_Toc528418021)

[Sidebars (‘Widget Areas’ is more of a correct name) 13](#_Toc528418022)

[Pages 14](#_Toc528418023)

[The WP Hierarchy 15](#_Toc528418024)

[Follow the link above in Pages to see how page template files are targeted in the hierachy, but basically this is a watered down version: 15](#_Toc528418025)

[Template-Parts > Content 16](#_Toc528418026)

[Singular.php 17](#_Toc528418027)

[Single.php 17](#_Toc528418028)

[Comments.php 17](#_Toc528418029)

[Home.php for the Blog Homepage 18](#_Toc528418030)

[Create the home.php Template 18](#_Toc528418031)

[Retrieve the Blog Posts Page Title 18](#_Toc528418032)

[Create the content-posts.php Template Part 18](#_Toc528418033)

[Backing Up Sites 21](#_Toc528418034)

[Child Themes 21](#_Toc528418035)

[Deploying to ServerPilot 21](#_Toc528418036)

[Settings 21](#_Toc528418037)

[General 21](#_Toc528418038)

[Still To Do 21](#_Toc528418039)

[What Not To Do 22](#_Toc528418040)

[In General 22](#_Toc528418041)

[There are two files required: 22](#_Toc528418042)

[The function file is used for: 22](#_Toc528418043)

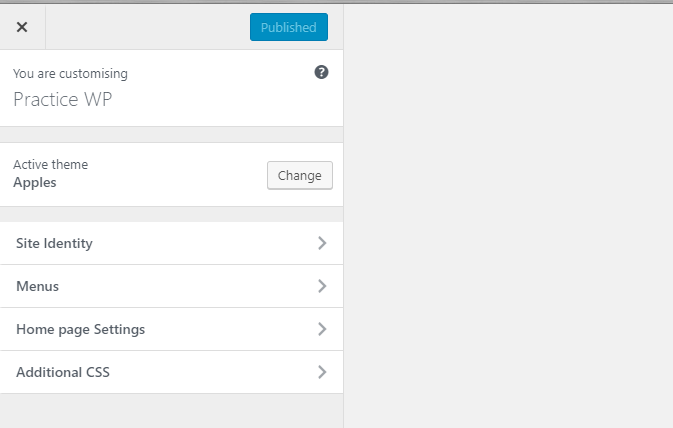
[There are five default post types readily available by the WP installation: 22](#_Toc528418044)

[Suggested theme folder and file structure following TwentySeventeen: 22](#_Toc528418045)

[The blog index page is home.php: 23](#_Toc528418046)

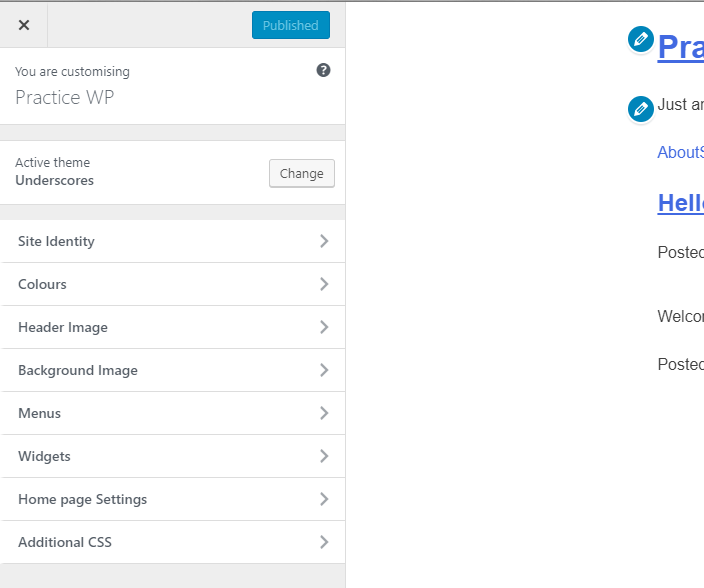
[The Loop 23](#_Toc528418047)

WordPress out of the box with a blank untouched theme, just the required index.php and style.css:



Comes with Site Identity function and Menus function through the customizer

**WordPress out of the box with a blank untouched Underscores theme, just a couple of styles added to move the sidebar to the right:**



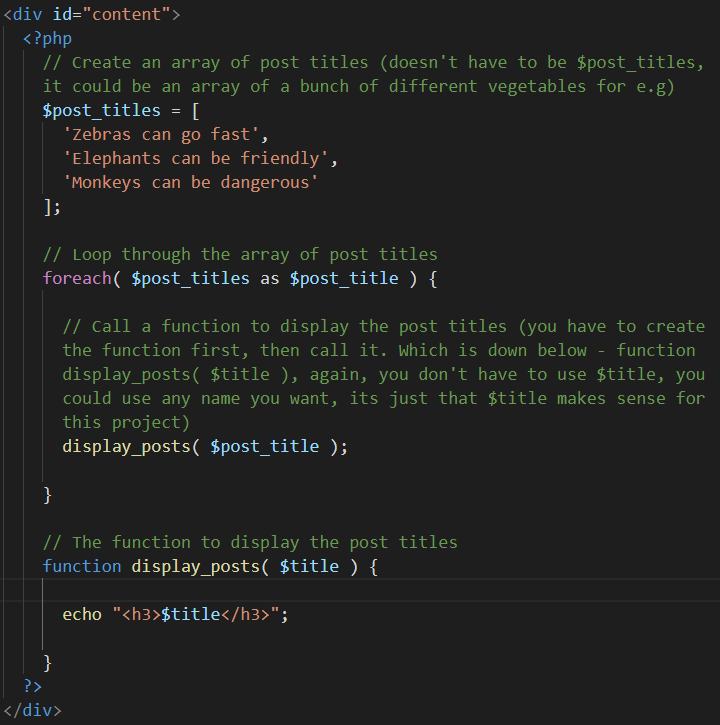
PHP for WordPress

WordPress Complete with Zac Gordon

<https://www.udemy.com/wordpress-theme-and-plugin-development-course/learn/v4/t/lecture/6867538?start=0>

Follow the link above to read PHP for WordPress (small list of WP php how to’s).

A very basic how to display a list of titles using foreach and a function:



The Loop

Starting a loop. The endwhile; stops the loop after it has finished looping

Starting a conditional. If have\_posts() returns true, meaning that there are posts in the database, move on to the while loop with its have\_posts() function. If not, move on to the \_e statement and then endif;

Starting a function. The the\_post() function



the\_post(); is a generic term that covers not only posts, but also pages, custom posts, any type of post really

The main content for a post or a page generally goes Inside of the WP loop (page?)

In PHP, starting a conditional and starting a loop both end in colons, where as the semicolon is ending the line.

\_e is an echo statement. It allows the string to be translated.

Just by creating style.css and index.php, then placing a loop in index.php, if we have a posts and pages created using the WP Admin, they show up in the site. Also, if we use the customizer and switch between the front page choices, it works aswell. When you install the demo test content it shows up too, because they’re stored in posts and pages. Index.php lists all the posts because at the moment the front page is set to display latest posts

Template Tags

* the\_title() - Gets and displays the title
* get\_the\_title() - Just gets the title, then later you can display it
* body\_class() - Entering this template tag into the body tag, WP will list an array of classes that are specific to each page. If you view page source on one page (say the home page), then view page source on another different page (say the blog page) some of the class names will be different. Class names that are specific to that page will change to match that page.

This way, you can then target pages separately from one another. You might have for example; A light grey background on the header of the home page, and then have a dark grey background on the header of the blog page.

The css would look like this: body.page-id-2 #header-bg {  
 background: dark-gray  
 };

post\_class() - Same as body\_class() but for posts.

Conditional Tags

&& - Checks to see if two conditions are met

|| - Checks to see if one or the other of two conditions are met

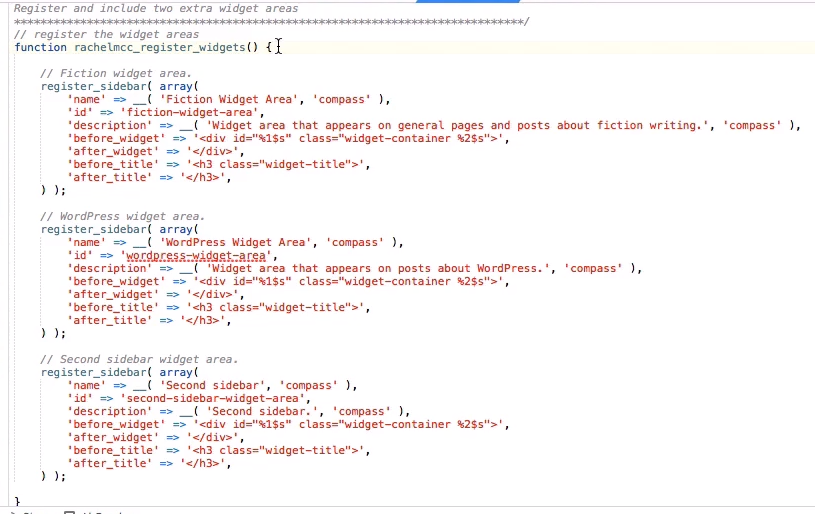
! – Checks to see if not the conditions are met

Learn PHP for WordPress with Rachel McCollin

Naming Conventions and Cases

There are four things to think about. Looking at the functions file, there are:

* Functions.
* Files, and the way that you name them.
* Classes.
* Variables that you create.



Rachel has a function, that she’s defined as **rachelmcc\_register\_widgets().** This is a WP function naming convention that requires a custom prefix, which is in this case, **rachelmcc.** This prevents any clashes with other sidebars that might be added.

Functions:

* Are written in lowercase and use underscores, not hyphens.
* Always have parentheses after the name. Paremeters can go inside them. After the parentheses functions have curly brackets, what the function does goes inside them.
* Are added into the functions.php file. They can’t be added directly into the theme template files. They can then be used anywhere in the theme simply by calling it.
* Start with its name - function

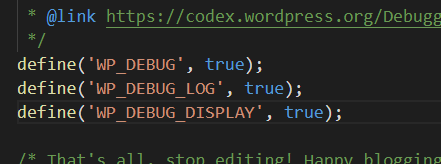
Themes that are designed with the option of implementing a child theme, the function in the parent theme must be made pluggable. This way it can be overriden without throwing an error. The child theme can use the same function and it will be run first, the parent theme function won’t run because of its if statement – if function does not exist (It already exists in the child theme):



When defining a php class, use capitols for the first letter: **WP\_Query**

Setup Debugging

In wp-config.php set WP\_DEBUG to true. Underneath that add the log and display constants (had to enter them one by one, otherwise it would throw an error):



The log constant when set to true, and when picks up an error, creates a file named debug.log in the wp-content folder.

The display constant when set to true, will show error messages in the html pages.

Debugging Plugins

Install:

* Debug Bar
* Log Deprecated Notices (To check log, go to Tools)
* Theme Check (To test the theme, go to Appearance > Theme Check)

Setup Test Demo Content & Install ‘Custom Post Type UI’ Plugin

Wptest Demo Content

Go to <http://wptest.io/> and download the test demo content. Then, as we go along building the site we can check to see that we’re supporting all the different scenarios of a theme that normally we wouldn’t think of. Extract the folder so we can select just the wptest.xml file in the next step.

In WP Admin > Tools > Import, install the WP Importer, run it and import the wptest.xml file. Next, select ‘Download and import file attachments’ and press the Submit button, then click on ‘Have fun’

‘Custom Post Type UI’ Plugin

There are different types of posts, for example there are blog posts, there are product posts. WP posts are blog posts, if you want a different type of posts, you use the ‘Custom Post Type Ui’ plugin. Taxonomies are tags and categories etc.

In WP Admin > Plugins install the Custom Post Type UI plugin. Add a new post type named Portfolio (see the plugin in WP Admin for settings). From the WP Admin sidebar click on Portfolio, then Add New, give it the heading ‘Template Tags’ and some dummy text, click on Publish. Create 3 more different named ones, all of them to live in Portfolio.

‘Advanced Custom Fields’ Plugin

Install the Advanced Custom Fields plugin. Click on it, then click on Add New and enter Project Fields into the name field. Click on Add New Field. Label = URL, Name = url, Type = Text, Instructions = Enter in the URL for the project, Required = Yes, Placeholder Text = http://, Formatting = No formatting, Style = Seamless (no metabox), Order No = 0, Post Type is equal to project.

In Portfolio add the url address and an excerpt and press Publish.

Create the 3 Main WP Files

In the themes folder create a folder named wp-com-zac, inside that create 3 files, one named style.css, one named index.php and one named functions.php. Go to WP Admin > Themes to see the new theme listed.

style.css

Go to <https://developer.wordpress.org/themes/basics/main-stylesheet-style-css/> and copy the header comment section for style.css. Paste it into style.css and update the details with your own.

functions.php

WP offers a bunch of different options that can be used in a theme, by default you have to tell WP which ones you want to use. Tell WP that your theme can support these features <https://developer.wordpress.org/themes/basics/theme-functions/>.

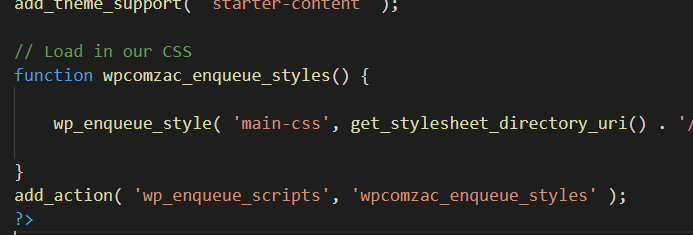
In functions.php add\_theme\_support() for the following elements:



Enqueuing CSS and JS

<https://developer.wordpress.org/themes/basics/including-css-javascript/>

Underneath the add\_theme\_support() list, add the wp\_enqueue\_style() for connecting the css:



Namespace the enqueue functions so that if someone else has an enqueue function it doesn’t conflict with ours, like if it was in a plugin that was added.

Hook the enqueue function into wp\_enqueue\_scripts, then when WP starts loading scripts it will add ours also

wp\_enqueue\_style tells WP to add it to the head of the current file



The time() stamp tricks WP into not caching the file while developing, it will think we have a new css file every second so don’t cache this and load me the newest version.

index.php

Enter an html5 base doctype boilerplate and an h1 with index.php written in it, see if it shows in browser.

Setup Header and Footer Template Files

header.php

In index.php cut the content out from the top at the <!DOCTYPE html> tag down to the <body> tag. In themes > wpcomzac create a file named header.php and paste it in there.

Underneath the meta tags add the wp\_head template tag. This enables other features such as enqueue\_scripts (css) to hook in to the head. The others are like, seo tags, the title tag

Custom Headers

To have separate headers for separate pages, just give them the name header with a hyphen and a name that can be whatever you want, for example; header-splash.php or header-front.php or header-blog.php etc. Then when you call it make sure to use the corresponding name for e.g; get\_header(‘splash’ );. You still have a default header.php for fallback.

footer.php

Same as header.php

Setup Menus and body\_class();

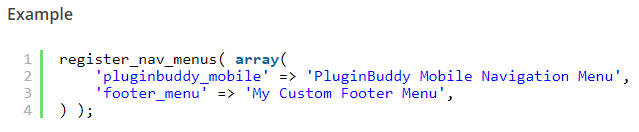
Menus

Setup dynamic menus, if the user starts to wreck the design by having too many links in the menu, get them to place them inside child links (dropdowns).

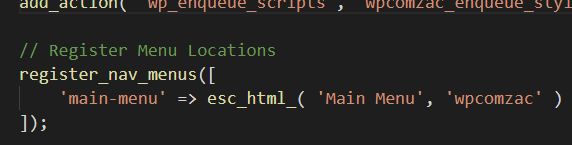
1. Register the menu with register\_nav\_menus()

<https://developer.wordpress.org/reference/functions/register_nav_menus/>

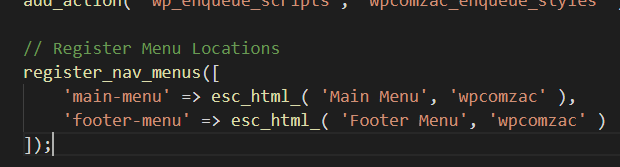
Register the menus and their locations using the register\_nav\_menus() function. Each menu is passed 2 parameters, the code that hooks into the menu location in header.php and the name that displays in the WP Admin menu:



Underneath the wp\_enqueue\_style() function add the register\_nav\_menus() function to register the nav and display it in the Admin menu and the customizer:



Multiple navs:



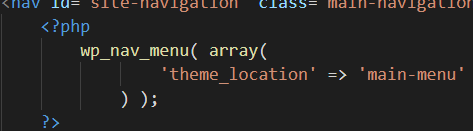
1. Setup the Nav Menu with wp\_nav\_menu()

<https://developer.wordpress.org/reference/functions/wp_nav_menu/>

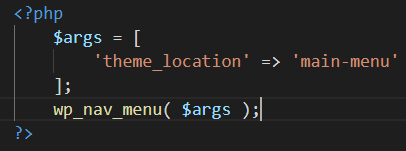
After registering the menus, we setup the menu using the wp\_nav\_menu() function. There are many different arguments we can use, listed in this link:

<https://code.tutsplus.com/tutorials/function-examination-wp_nav_menu--wp-25525>

In header.php enter the nav\_menu(); function into its location (header.php, just under the opening body tag):



Another way to enter it is this:



1. Define the Menu in Admin > Menus

Even though we’ve created a menu, and its been entered into the Admin menu, we still have to define it (create it using the Admin ui).

1. In WP Admin > Menus > Edit Menus, click on ‘create a new menu’, then enter a name for the new menu.
2. In the Pages section, select the pages you want to add to the menu, click on Add to Menu.
3. In Menu Settings, check the location (Main Menu) check box.

WordPress adds a bold class to the links using .current-menu-item.

body\_class();

Entering this template tag into the body tag, WP will list an array of classes that are specific to each page. If you view page source on one page (say the home page), then view page source on another different page (say the blog page) some of the class names will be different. Class names that are specific to that page will change to match that page.

This way, you can then target pages separately from one another. You might have for example; A light grey background on the header of the home page, and then have a dark grey background on the header of the blog page.

The css would look like this: body.page-id-2 #header-bg {

background: dark-gray

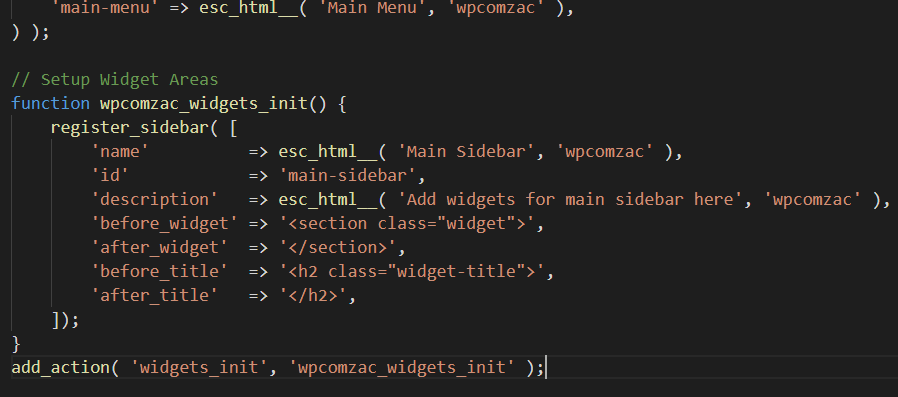
};

In index.php remove the custom ‘splash’ code from get\_header and get\_footer.

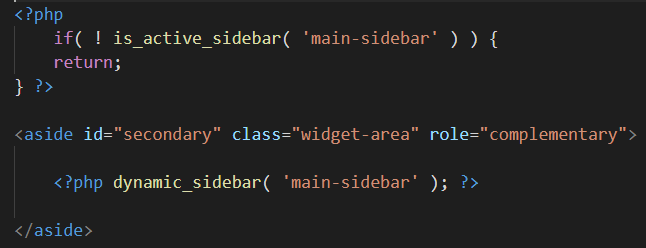
In header.php enter the body\_class(); into the body tag.

Sidebars (‘Widget Areas’ is more of a correct name)

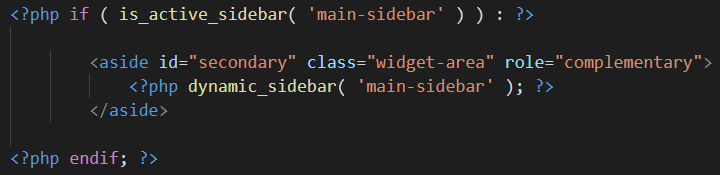
In functions.php register the sidebar(s):



In sidebar.php enter the sidebar to pull in your widgets:



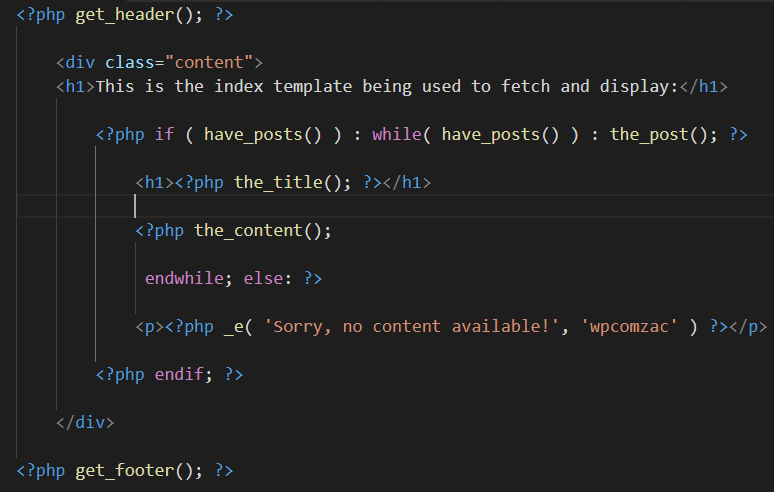
The if statement means that If it hasn’t been populated, return so it stops right there and doesn’t try and load it which will throw an error. Alternatively use if it has been populated keep going. They both stop the error, doesn’t matter which way you go. The conditional prevents an error and makes things more efficient.



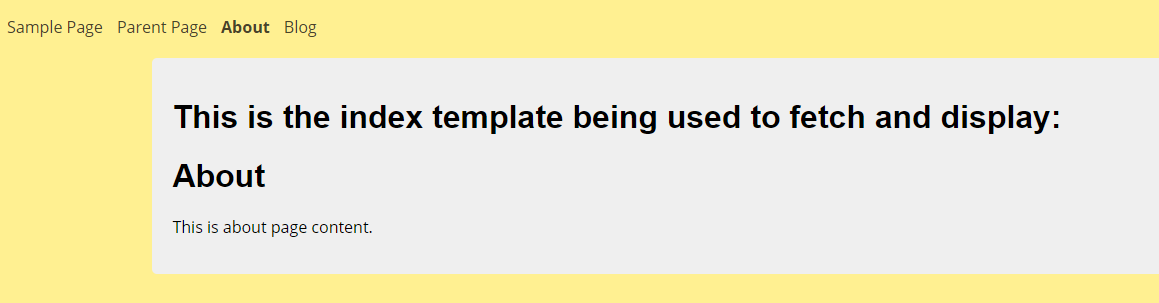
You can add a whole big list of widgets now, to that one area, that one widget area. To add more widget areas just duplicate register\_sidebar(), then in another file add the if statement and dynamic\_sidebar().

Pages

You can create virtual pages from the WP Admin > Pages, by clicking on Create New Page, but they won’t load unless there’s the WP loop there to load them. If there’s no templates made up but just the index.php template (index.php is not a page, it’s a template) with a loop inside it, WP will use index.php template to load the pages, and whatever layout is in the index.php template for positioning the page content. In index.php enter the WP loop:



Because we haven’t created any templates yet besides index.php, the index.php template is used to fetch and display the about page in the browser.



You have a static page for the front page (one page scroll). It doesn’t require a page template, because its static, there is only one of them, and the client wont be making another one, it’s the home page, there’s only one home page on any website. Then you have a page template (page.php) <https://developer.wordpress.org/themes/template-files-section/page-template-files/> for pages such as policy, legal, etc, and for any extra pages that the client might want to add to the site.

So to have a one page scroll front home page, you create a new file in the root named front-page.php then in WP Admin > Settings > Reading switch the switch for having a static front page and select front-page.php.

If we just use index.php for our template, every page is going to look the same, so we create other templates such as page.php for pages, single.php for single blog posts, archive.php for blog archives, etc. Then index.php is set up with the loop to display the blog post listing page as a fallback. Index.php is just a fallback template used to fetch and display the blog post listing page.

Instead of repeating the loop code over and over again in each individual file, it can be kept in its own include file and be called on to each template that requires it. You can also give it a name to specify if the loop is only for a certain type of template. The loop file would be named loop-archive.php:



Anyway, following along with the tutorial:

The WP Hierarchy

Follow the link above in Pages to see how page template files are targeted in the hierachy, but basically this is a watered down version:

* *page.php* – Controls how single pages are displayed.
* *single.php* – Controls how single blog posts are displayed.
* *category.php* – Controls category archive pages.
* *tag.php* – Controls tag archive pages.

Fallbacks:

* *page.php* > *singular.php* > *index.php*
* *single.php* > *singular.php* > *index.php*
* *attachment.php > singular.php > index.php*

You can leave the singular.php one out if you want?

Here’s the hierarchy a bit more specific:

So, for example, if a visitor is reading a single post from your blog, WordPress will work through this list of files and use the first one it finds from this list to display content:

1. *single-post.php* (for single posts, not pages or custom post types)
2. *single.php* (for single posts or custom post types, not pages)
3. *singular.php* (for single posts, custom post types or pages)
4. *index.php* (for anything that doesn’t have a more specific template part)

And if your user navigates to an archive page, for example the archive for the “Featured” category, WordPress will work through these template files:

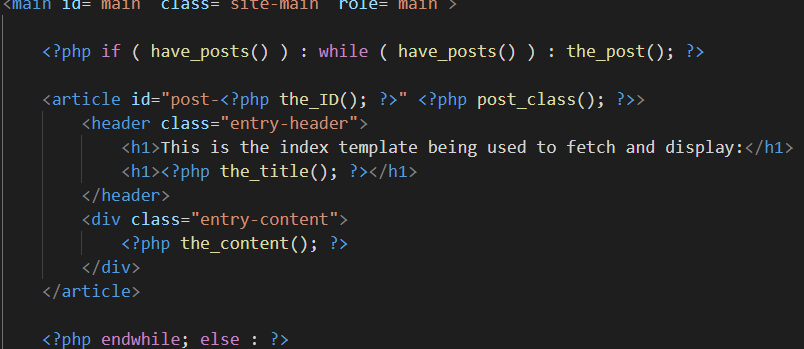
1. *category-featured.php* (just for that category)
2. *category-XX.php* (where XX is the ID of the ‘featured’ category)
3. *category.php* (for all category archives)
4. *archive.php* (for all archives including categories, taxonomy terms, authors and tags)
5. *index.php*

When we imported the wptest.io dummy data, it created a home and a menu page, both with instructions on how to set them up to display one as home (front) page and one as blog page. In Admin > Settings > Reading, set the home page to display as front page and the blog page as blog page, then the content of the blog page should have been deleted and now showing the list of blog posts.

Template-Parts > Content

In wpcomzac root add a folder named template-parts, inside that add a file named content.php and a file named content-none.php

In index.php, cut this content to the clipboard:



Paste it into content.php

In index.php, cut this content to the clipboard:



Paste it into content-none.php and in the browser everything should still work the same.

Singular.php

Save index.php as singular.php. Inside index.php enter a h1 with index.php, inside singular.php enter an h1 with Template: singular.php. Test in the browser the hierarchy fallback with both of them.

Single.php

Save singular.php as single.php, inside it enter an h1 with Template: single.php. Test in the browser to see now if single.php is being used to load the single posts.

To make the single post different from the pages and home blog page, we can add content that’s specific for the single post. So save template-parts/content as content-page, in singular.php (the fallback for pages) add in ‘page’ as a second argument to the get\_template\_part() tag.

The WP Cycle

Human readable URL – <https://codecrayons.com/single-post>

Actual WP URL to the post – <https://codecrayons.com/?p=2726> , 2726 is the id to the post object we want to retrieve from the database and display on the front end.

p=2726 🡪 Database Server says it’s a post, and goes ahead and grabs it 🡪 The info is sent to WP, and WP grabs the correct template: single.php. With WP out of the box, single.php has references to header.php, footer.php, content.php and sidebar.php

You might have different headers, one type for posts and another type for pages. In that case you just add to the end of header.php, e.g; header-custom.php

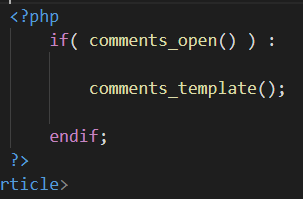
Comments.php

Copy the Underscores comment.php file into wpcomzac, change all the handles from underscores to wpcomzac, then in content.php just before the closing article tag add **<?php comments\_template(); ?>**

See it in the browser in a single post, then go to Edit Post > Screen Options and turn on Discussion and Comments. Deselect ‘Allow Comments’, now the comments are closed.

To not show any comments at all, in content.php add this conditional to the **comments\_template();**

It checks to see if comments are open (switched on in the admin), and only if they’re open, then display comments. If they’re not open, don’t display any comments at all:



Home.php for the Blog Homepage

Home.php falls back to index.php if its not available, if home.php is there it is worked as the blog homepage.

When a virtual home page and blog page are made, you can choose to have a static front page for the home page and use the blog page for the posts. Or you can choose to not have a static front page and have the posts for the home page. But because there’s no home.php template and no blog.php template, WP will use the hierarchy fallback system. If you choose to have a static front page WP will fallback to the next template singular.php if there is one, if theres not then index.php. The posts page will look for home.php template, if there isnt one then singular.php, then after that index.php.

Anyway, the best thing to do is to create the front-page.php template and the home.php template so the pages can have their own templates and we can style them differently.

Create the home.php Template

Save index.php as home.php, inside it enter an h1 with Template: home.php. Test in the browser to see now if home.php is being used to load the posts page.

Retrieve the Blog Posts Page Title

By entering wp\_title(); (different to the\_title(); which gets and echos out the post title, wp\_title(); gets and echos out the page title) into the home.php template, the function retrieves the page title. The title will by default have a couple of arrows before it because its expecting two names, the name of the page and the name of the site (as in the page tab). To fix this, just add an empty parameter between the parentheses.

Create the content-posts.php Template Part

We need to make the posts home page so that it just retrieves and displays the excerpts and links, which is different to the single post page where it displays the whole post and a link that is different to the posts page link. So one way of doing it is to create a whole new template part named content-posts.php. So in home.php add posts to the content template tag. Then save content.php as content-posts.php, change the h1 to an h2, add a link to its single post, delete the comments section, and change the content tag to an excerpt tag:



Use **esc\_url()** so we can attach the link to it?

Add paginate\_links(); to the home.php template.

Backing Up Sites

Github has a copy of all the sites created and hosted at ServerPilot, so we can clone them to local for swapping the demo content with clients content. These sites should also be backed up on one of the cloud platforms (Keep them stored in the Documents folder and they’ll be backed up to Microsoft’s OneDrive cloud)

Using AutoMySqlBackup will run a mysqldump every day and store the files in a folder of the websites /var/lib/automysqlbackup folder. Once a week, copy the automysqlbackup folder to the cloud platform where the sites are also. Plus keep a copy on pc.

Backup the .env file and the /etc file if that has customized configs. The remote .env file is different to the local .env file.

Child Themes

<https://www.udemy.com/wordpress-theme-and-plugin-development-course/learn/v4/t/lecture/7407140?start=0>

<https://developer.wordpress.org/themes/advanced-topics/child-themes/>

Most themes come out with updates. Some are for security, some are for new functionality, and some are to keep up with WP core and theme best practices. When you run a theme update, it will pull down the new theme and override all the code for your current theme. So if you made any changes to the style sheets or templates, they’ll all be overridden. This is why the child theme was invented.

Deploying to ServerPilot

Just use the same WP Admin Sign In that you use for signing in to the Admin. Or Install a WP plugin that creates a password for using on sites before going public.

Settings

General

Set Timezone to Sydney

Still To Do

* Add more tags to the theme intro.
* Use Understrap starter theme (maybe use Sage one day, but they build things differently like adding the add\_action hook before enqueuing)
* To get to know how to use the WP editors to customise themes, using the Flatsome theme to copy off, customise (change the whole design into a one page scroll maybe) the Twentyseventeen theme with the new WP Gutenburg editor. Then customise another Twentyseventeen theme using the Elementor drag and drop editor plugin.

What Not To Do

* Don’t use backup or caching plugins, they slow the site down, store data in it etc. ServerPilot has a good caching system setup

<https://getflywheel.com/wordpress-support/what-plugins-are-not-allowed/>

In General

There are two files required:

* + index.php – Fallback file for when there’s no file in the hierarchy.
  + style.css – For description of the theme.

The function file is used for:

* + Enqueuing styles and scripts.
  + Registering theme support for functions such as menus, html5, dynamic title tag, widgets, etc.
  + And more…

There are five default post types readily available by the WP installation:

* + Post
  + Page
  + Attachment
  + Revision
  + Navigation menu

Suggested theme folder and file structure following TwentySeventeen:

* + assets
    - css
    - images
    - js
  + inc
    - custom-header.php
    - customizer.php
    - jetpack.php
  + template-parts
    - footer
    - header
    - navigation
    - page
    - post
  + 404.php
  + archive.php

The blog index page is home.php:

* By default WP sets home.php or page.php as the front page (depending on what is set in Settings > Reading). If front-page.php exists it will override it.

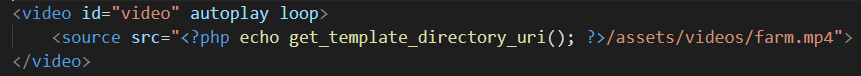
The Loop

<https://developer.wordpress.org/themes/basics/the-loop/>

* Default mechanism for outputting posts through the themes template files. In Reading settings it is determined how many posts are retrieved per page.
* The post data is collected from the database and inserts it in place of each template tag. Any html or php code in the loop will be processed for each post.
* The loop should be placed in index.php and any other template that requires post information. It must always begin and end with the same if and while statements.
* List of common template tags and conditional tags that can be used in a loop:  
  <https://developer.wordpress.org/themes/basics/the-loop/#what-the-loop-can-display>
* Use rewind\_posts() to display the posts in a table of content at the top of the page, then display the posts below.

Video

Normally, in a static site you would add the video with a link, style it and that would be it. In WordPress, with a front-page.php, a blog on the back with home.php, and a video header on the front page, a static setup would be something like this:



Custom Header

This way does not set it up with a default image

* Add theme support for it in functions or include
* Enter it into the header.php file
* Upload the image through the WP library thingo
* To have the image on front page only add this conditional:



**Use echo when a function begins with get, get just gets the item, it doesn’t display it to the browser**

**The codex is different to Underscores and the Teens, just use them, stuff the codex.**

**Front-page.php**

**Walker allows you to customize how menus appear. By default menus have a certain structure of unordered lists, lists, anchor tags etc. Sometimes when using Bootstrap for e.g, you wanna override it. Although you can do a lot with css and not have to use it?**

**In the Bookmarks Bar > WordPress 2 > wpmudev folder, do the tutorials. Start at Beginners: Building Themes and continue through the series. That leads to Intermediate: Theme Development in Detail, continue through the series. That leads to the WP Customizer tuts.**

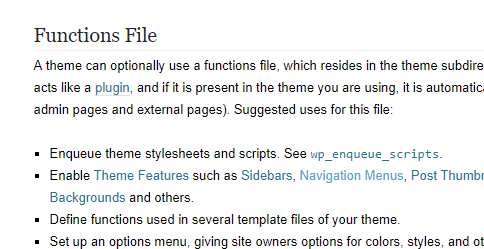
**How to use the Codex**

**Start at Codex Main Page:**

[**https://codex.wordpress.org/Theme\_Development**](https://codex.wordpress.org/Theme_Development)



**Scroll down to the element, function or Hook etc (In this case, the Navigation Menus link and click on it ):**



**Which takes you to here:**

