# WordPress Notes

## Things to know

### PHP specific notes

**\_e(….)** – translates text to common language website is being viewed in. This is to be echo’ed out

\_e(“translate me then echo me”, themeName);

**\_ \_()** – trnalsates a name of parameters, does not echo

\_\_(“translate me”, themeName);

**printr(…)** – similar to console.log() but will output on to the dom.

**echo nl2br(…)** – forces <br> tags to be inserted instead of ignored

<?php echo nl2br(get\_the\_author\_meta( 'description')); ?>

Outpts author bio info and forces <br> tags instead of ignores it and makes a giant paragraph

**var\_dump** – a way of outputting a parameter/variable to see it’s content

this is done in a php file that outputs to the DOM

in the example, it was done in the theme-customizer.php file under the first f()

echo '<pre>';

    var\_dump( $wp\_customize );

    echo '</pre>';

**Functions**

There are **get\_** vs **non-get\_** versions when calling a function

get\_ - returns value – *easier to modify returned value*

non-get\_ - will echo out value

values echoed out cannot be manipulated easily

**the\_post()** vs g**et\_the\_post()**

**add\_theme\_support** – used to add specific features to WP such as featured Images, title tags, custom logos, etc

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

check setup.php file for examples

### RSS Feeds

Add to setup.php or functions.php

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

*add\_theme\_support( 'automatic-feed-links' );*

### HTML 5

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

To have WP generated HTML content converted to be XHTML compatible, add the following line of code

*add\_theme\_support( 'html5' );*

### Startup/Dummy Content

Used when WP installation is initlally created and is used as filler info

Can create posts, pages and menus, etc

*Note: not really sure if this is useful, why not just use fakerPress plugin for dummy data???*

<https://make.wordpress.org/core/2016/11/30/starter-content-for-themes-in-4-7/>

*add\_theme\_support( 'starter-content', [*

*// Create filler content for the following WP items:*

*// Place three core-defined widgeets in the sidebar area*

*'widgets'         => [*

*'ju\_sidebar'    => [ 'text\_business\_info', 'search', 'text\_about']      // adds bus addy/hrs, search bar, about txt*

*],*

*// Create the custom image attachments used as post thumbnails for pages*

*'attachments'     => [*

*'image-about'   => [*

*'post\_title'  => \_\_('About', 'udemy'),*

*'file'        => 'assets/images/about/1.jpg',                          // URL relative to the template director*

*],*

*],*

*// Specify the core-defined pages to create and add custom thumbnails to some of them*

*'posts'               => [*

*'home'              => [ 'thumbnail' => '{{image-about}}',],            // {{ looks for any images w/ ID image-about }}*

*'about'             => [ 'thumbnail' => '{{image-about}}',],            // Note: image-about is reference previously*

*'contact'           => [ 'thumbnail' => '{{image-about}}',],*

*'blog'              => [ 'thumbnail' => '{{image-about}}',],*

*'homepage-section'  => [ 'thumbnail' => '{{image-about}}',],*

*],*

*// Default to a static front page and assign the front and post pages*

*'options'         => [*

*// Note: These reference the 'posts' defined previously*

*'show\_on\_front'     => 'page',                                  // overwirtes homepage: blog posts or single page*

*'page\_on\_front'     => '{{home}}',                              // which actual page goes on homepage*

*'page\_for\_posts'    => '{{blog}}',                              // which actual page will be for post page*

*],*

*// Set the front page section theme mods to the IDs of the core-registered pages*

*'theme\_mods'      => [*

*'ju\_facebook\_handle'      =>  'udemy',                          // overwrites theme settings created*

*'ju\_twitter\_handle'       =>  'udemy',*

*'ju\_instagram\_handle'     =>  'udemy',*

*'ju\_email'                =>  'udemy',*

*'ju\_phone\_number'         =>  'udemy',*

*'ju\_header\_show\_search'   =>  'yes',*

*'ju\_header\_show\_cart'     =>  'yes',*

*],*

*// Set up nav menus for each of the two areas registered in the theme*

*'nav\_menus'       => [*

*// Assign a menu to the 'top' location*

*'primary'       =>  array(*

*'name'        =>  \_\_( 'Primary Menu', 'udemy' ),*

*'items'       => array(*

*'link\_home',              // core 'home' page is a link in case of a static front page is not used*

*'page\_about',*

*'page\_blog',*

*'page\_contact',*

*),*

*),*

*// Assign a menu to the 'social' location*

*'secondary'     =>  array(*

*'name'        =>  \_\_( 'Secondary Menu', 'udemy' ),*

*'items'       =>  array(*

*'link\_home',              // core 'home' page is a link in case of a static front page is not used*

*'page\_about',*

*'page\_blog',*

*'page\_contact',*

*),*

*),*

*],*

*]);*

To test, edit value in DB : **wp\_options / fresh\_site**

Default value is 0

Set it to 1 to make WP think it’s a new install

## Template Tags

Various functions and methods to be used in WP display elements on the page

<https://developer.wordpress.org/themes/references/list-of-template-tags>

**get\_header()** – inserts header.php file

**get\_footer()** – inserts footer.php file

**home\_url()** – link to main/landing page

<?php echo home\_url(); ?>

<?php echo home\_url( ‘ / ’ ); ?> - adding a / as an argument specifies you only want root url

**bloginfo()** – different values and properties of the site such as name, url, admin email, etc

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

## Header Titles and Logos

**wp\_titles()** – function to add to insert page title

function is deprecated but is still active for compatibility

pass an add\_theme\_support( ‘title-tag’ ); in the setup.php file

SEO Plugins should be supporting this as well.

add\_theme\_support ( ‘custom-logo’ ); - adds a custom logo to the site

to add a logo, click **Customize** **-> Site Identity** at the top when viewing a page

## Advertisements

Use QPQUADS plugin <https://wpquads.com/> & <https://wordpress.org/plugins/quick-adsense-reloaded/#description>

Plugin is dev friendly and can manage all ads on site

**Documentation** is found in the plugins folder in includes/api.php file

Ccode needs to be added to functions or setup php file to initialize

Additional php code is to be added where the ad is to be placed

Plugin Setup – Used Classic View instead of new interface

Under **Ads** tab, insert html for ad as an img tag

Under **General** tab, check assign checkbox on custom name created (usually at bottom of list)

## Unique WP Variables

$post – a variable used in The Loop that contains additional data (object values) about the current blog post

Also referenced as: WP\_Post *object* and the\_post()

<https://developer.wordpress.org/reference/classes/WP_Post/>

to access object values, use the -> notation

$post->ID;

Used to access post ID number

Can also be $somevar = the\_post(); $somevar->ID;

WP\_Post Object

(

    [ID] =>

    [post\_author] =>

    [post\_date] =>

    [post\_date\_gmt] =>

    [post\_content] =>

    [post\_title] =>

    [post\_excerpt] =>

    [post\_status] =>

    [comment\_status] =>

    [ping\_status] =>

    [post\_password] =>

    [post\_name] =>

    [to\_ping] =>

    [pinged] =>

    [post\_modified] =>

    [post\_modified\_gmt] =>

    [post\_content\_filtered] =>

    [post\_parent] =>

    [guid] =>

    [menu\_order] =>

    [post\_type] =>

    [post\_mime\_type] =>

    [comment\_count] =>

    [filter] =>

)

## The Loop

The loop is a while loop that outputs blog post articles until all articles are posted.

**have\_posts()** – returns true if any blog articles/posts exists

**the\_post()** – returns false when no more posts exist, used to break out of while loop

**get\_template\_part(‘dir/dir/…’)** – inserts .php file located in specified directory

 <?php

if (have\_posts() ) {               *// check for any posts*

while ( have\_posts()) {         *// loop while posts exists*

the\_post();                     *// returns false when no more posts; breaks while loop*

get\_template\_part( 'partials/post/content-excerpt' ); *// content-excerpt.php holds the html code for each blog post*

}

}

?>

The following template tags are used in The Loop to output the data pertaining that blog post article

**the\_permalink()** – blog post link of title being displayed

**the\_title()** – blog post title

<a href="<?php **the\_permalink();** ?>"><?php **the\_title();** ?></a>

**get\_the\_date()** – sets the date of the blog, in multiple blog posts in 1 day, each blog post will show the date

Default will only show the date on the last blog post posted that day

**the\_date()** – similar to default setting

<?php echo get\_the\_date(); ?>

**the\_post\_thumbnail( ‘*size’, [‘class’=>’class names’]* )** – insert thumbnail image of post

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

**get\_author\_posts\_url(get\_the\_author\_meta(‘ID’))** – Links to posts made by author

**the\_author()** – shows the author

<a href="<?php echo get\_author\_posts\_url(get\_the\_author\_meta('ID')); ?>"><?php the\_author(); ?> </a>

**the\_category(' ')** – display categories; parameter is what’s used to separate each category – in this instance it’s a blank space

**comments\_number('0')** - shows the number of comments; parameter is used to show what happens when 0 comments are made

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

in this instance 0 is used vs (default) No Comments

second parameter is when 1 comment is made

third parameter is when multiple comments are made.

<?php comments\_number('0') ?>

**the\_excerpt()** – output short condensed summary of blog content

<?php the\_excerfpt(); ?>

**the\_content()** – outpt full text content of blog post

<?php the\_content(); ?>

**wp\_link\_pages()** – output additional pages in a blog post

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

blog post must have page breaks

contains multiple arguments to be passed in if wanted

can be added after the\_content() function call

<?php

                the\_content();

                wp\_link\_pages();

              ?>

**the\_tags()** – output tags related to post

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

<?php the\_tags(); ?>

3 parameters – 1) text to appear before tags 2) separator 3) text to appear after

**wp\_reset\_postdata()** – after a secondary loop is created, this must get called after to reset variables in primary/The Loop

## Comments

**wp-comments-post.php** – file found in WP root that processes all comments

**comemnts\_template()** – calls comments.php file

### Comments.php

**post\_password\_required()** – inserted at the top to check if password is needed or if user inputed it in

Is not used in single posts page as WP already handles that

Needed in comments as comments can be displayed anywhere else in site

if (post\_password\_required()) {

    return;

  }

**comment\_form()** – creates a form to insert a comment to blog post

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

contains various amounts of parameters

#### comments section

**have\_comments()** – used to check if comments exist conditional in the beginning of comments section

if (have\_comments()) { … *begin start of comments section* … }

Use a foreach loop to loop through each comment objects

foreach( $comments as $comment ) { … *html code for comments* … }

**comments\_number()** – add total number of comments on blog post

<?php comments\_number() ?>

Outputs: # Comment(s)

**get\_avatar()** – insert the user’s avatar image if blog uses it

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

contains 5 parameters

1. Variable to loop through
2. Size of image
3. Default image if one doesn’t exist
4. Alt tag string for each avatar
5. Array values such as classes to be added

<?php echo get\_avatar( $comment, 60, 'monsterid', 'avatar image', [ 'class' => 'avatar avatar-60 photo avatar-default' ]); ?>

**comment\_author()** – inserts author of comment

**comment\_date()** – date comment was posted

**comment\_text()** – text content by comment author

**the\_comments\_pagination()** – insert pagination for comments

## Author Box

**the\_author()** – outputs author of post

**author\_URL** – outputs page of posts by author or url for author profile

<a href="<?php **echo $author\_URL;** ?>"><?php **the\_author();** ?></a>

**get\_the\_author\_meta()** – outputs the text about author, contains a series of parameters that can be inserted

<?php echo get\_the\_author\_meta( 'description'); ?>

Outputs description if it exists. If not then nothing gets output.

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

## Pagination

**next\_posts\_link( ‘text’ )** – link to the next page if it exists; text is the text to display for the link

**previous\_posts\_link( ‘text’)** – link to previous page if it exists, text is the text to display for the link

**next\_post\_link()** – similar to above but this is used in a single blog post article

**previous\_post\_link()** – similar to above but this is used in a single blog post article

titles are automatically generated for both links

## Search

**get\_search\_form()** – output the search box

**the\_search\_query()** – outputs the text the user searches for

used to show the result of what the user input for search

## Custom Template

### Templates for Pages

<https://developer.wordpress.org/themes/template-files-section/page-template-files/>

Custom pages are for pages only not for blog posts

Create a custom template by creating a comment in php or html at the top of the page.

In comment add string: **Template Name:** blahblahblah

WP will look for Template Name and create that as a template to be used when creating a WP page.

Select Template drop down menu in WP page setup

### Templates for Post Types

<https://developer.wordpress.org/themes/template-files-section/page-template-files/#creating-page-templates-for-specific-post-types>

<https://developer.wordpress.org/themes/template-files-section/custom-post-type-template-files/>

Similar as custom templats for pages but header comment has to have:

**Template Name:** Whatever Name

**Template Post Type:** *post type*

## Customizer API

Create a menu selection when customizing a page

This can be found when viewing a page and selecting the “Customize” button at the top WP admin menu

Add additional selections to the Customize Menu on the left when selected

These options can contain color selections, input boxes, radio boxes, etc

In functions.php add the add\_action hook:

add\_action ( ‘customize\_register’, ‘whaterver\_name’);

When creating the function, add the $wp\_customize parameter

function ( $wp\_customize){…}

This parameter has to be used when calling this function

<https://developer.wordpress.org/reference/classes/wp_customize_manager/>

In the function, you need to

add\_setting - <https://developer.wordpress.org/reference/classes/wp_customize_manager/add_setting/>

this is the value stored in the db

add\_section – Creates the title and priority of menu (lower number, higher placement in menu

add\_control - <https://developer.wordpress.org/reference/classes/wp_customize_manager/add_control>

assigned under the section and controls the setting

creates the input field in the menu

*see theme-customizer.php* *file for how it’s used*

in this example, created an input box to input the user’s fb address (ex joeyparagas, this will make it so the fb icon will link to fb.com/joeyparagas)

Note that you need to go to the header.php file where the fb link exists.

 if( get\_theme\_mod('ju\_facebook\_handle') ) { *// Check if the customizer link exists using* ***get\_theme\_mod()*** *function*

to output the user input to the a href tag, you need to echo out the data

 <a href="https://facebook.com**/<?php echo get\_theme\_mod('ju\_facebook\_handle'); ?>**">Facebook</a>

Note that if the user **DOES NOT** fill in the data, the icon will disappear due to the *if* condition not being met

### Output Page ID Using Permalinks

When using permalinks to output a href link, you can use get\_theme\_mod (‘function\_name’) to output the ID of the page

<a href="<?php the\_permalink(get\_theme\_mod ( 'ju\_footer\_privacy\_page')); ?>">Terms of Service</a>

*See footer.php example*

### Organizing Customizer

3 methods/hierarchy of organizing theme customizer – Panels, Sections, Controllers

#### Panels

Panels group the theme customizers together under one submenu

This creates a *Udemy* submenu which then opens up into *Social Settings* and *Misc Settings*

Add an *add\_panel* section in the theme\_customizer.php

This creates the actual panel itself

Under the *add\_section* for each theme customizer call, add *panel* in array under the priority

This moves each *section* under the panel

Movies *Social Settings* and *Misc Settings* under the Udemy submenu

#### Changing Existing Controllers, Panels & Settings

Find the ID of the existing panel. In this example, var\_dump was used to output the contents of wp\_customze object

EX: Change *Site Identity* Panel settings

Search wp\_customzie object for *Site Identity* and look for ID name which is *title\_tagline*

Use *get\_section* method to edit the panel settings

*get\_section()* to edit the existing section

*get\_control()* to edit existing controllers

### Using Javascript in settings

In this example, we will use JS to refresh the setting on the fly vs using page load to refresh the page.

This isn’t necessary but just makes it slightly faster

The example is to edit the *Social Settings* add search & cart selection

When the user makes a selection, it would remove/add the graphic on the fly vs using a page refresh

Add a ‘transport’ key into the *add\_settings* method and set it equal to *‘postMessage’*

WP defaults to using *Refresh* which causes the page to refresh on any changes

postMessage changes to use JS instead

You can use an object ***wp*** which comes packaged with all the Wordpress methods and functions needed

**wp**.customize – this can access the customize object

console.log(wp) to see what else you can use

### Using Color Picker & Upload Controllers

Use add\_setting and create another id and set the default to the default hex color

*$wp\_customize->add\_setting('ju\_read\_more\_color', [*

*'default'        => '#1ABC9C'*

*]);*

Add a controller <https://developer.wordpress.org/reference/classes/wp_customize_color_control/>

*$wp\_customize->add\_control(*

*new WP\_Customize\_Color\_Control(*

*$wp\_customize,*

*'ju\_read\_more\_color\_input',*

*array(*

*'label'      => \_\_( 'Read more link color', 'udemy' ),*

*'section'    => 'ju\_misc\_section',*

*'settings'   => 'ju\_read\_more\_color'*

*)*

*)*

*);*

Wordpress will create the interface in the Customizer due to the controller.

In order for the color to take effect, the style color change needs to be inserted via php

In this example, we use inline css into the header and use a variable that equals the selection of the color from the controller

Enqueue code below:

#### Enqueue Inline CSS

Use **wp\_add\_inline\_style( ‘id\_name’, ‘css code’);**

*This example comes from front/enqueue.php*

 wp\_add\_inline\_style(

    'ju\_custom',                                                // load if this exists; I this example, ju\_customs points to the custom.css file

    'a.more-link { color:' . $read\_more\_color . '; border-color: ' . $read\_more\_color . '; }'

  );

### File Upload & Link Capabilities

Create a file upload dialog box in customizer.

If file exists, then show on page

Use add\_setting and set default to empty string:

*$wp\_customize->add\_setting('****ju\_report\_file'****, [*

*'default'        => ''*

*]);*

Add a controller: <https://developer.wordpress.org/reference/classes/wp_customize_upload_control/>

*$wp\_customize->add\_control(*

*new WP\_Customize\_Upload\_Control(*

*$wp\_customize,*

*'ju\_report\_file\_input',*

*array(*

*'label'      => \_\_( 'File Report', 'udemy' ),*

*'section'    => 'ju\_misc\_section',*

*'settings'   => '****ju\_report\_file****'*

*)*

*)*

*);*

Create HTML link where you want the downloadable link to appear

*<?php*

*if ( get\_theme\_mod('****ju\_report\_file****')) { ?>*

*<a href="<?php  echo get\_theme\_mod( 'ju\_report\_file' ); ?>">Download Report</a><br>*

*<?php*

*}*

*?>*

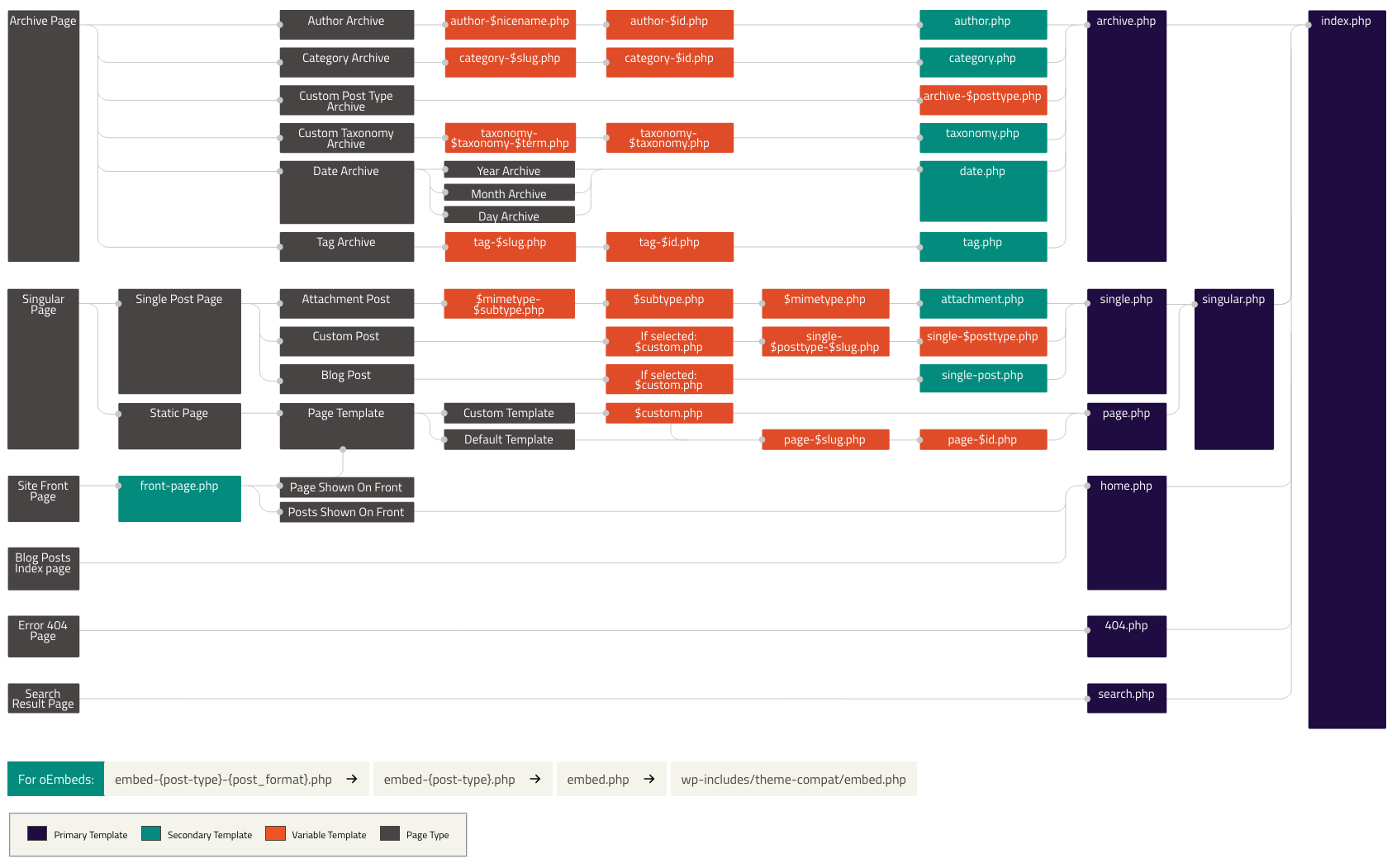
## Page Templates

<https://developer.wordpress.org/themes/basics/template-hierarchy>

Heirarchy diagram - <https://developer.wordpress.org/files/2014/10/Screenshot-2019-01-23-00.20.04.png>

Diagram showing what template file is used when loading a specific page

<https://wphierarchy.com/>



### Page Template Order

WP will be looking for pages in the following order

* page-*slugName*.php (page-contact.php)
* page-*idNum*.php (page-10.php)
* page.php
* singular.php
* index.php

### Category Template Order

WP will be looking for categories in the following order

* similar to index.php
* category-*slugName*.php (category-movies.php)
* category-*idNum*.php (category-10.php)
* category.php
* archive.php
* index.php

### Date Template Order

WP will be looking for dates in the following order

* similar to category.php
* not used often
* **dates that don’t have pages associated to them will return a 404 page**. (example.com/2030 will go to 404 page since doesn’t exist)
* date.php
* archive.php
* index.php

### Attachment Templates

Attachments are any files that are uploaded into WP

Attachment page is the page that opens up when you want to view the file via WP/Media folder/click on file/View attachment page

WP will be looking for mime types in the following order:

* *mime-type*.php (image.php)
* *attachment*.php
* *single*-attachment-*slug*.php
* single.php
* singular.php
* index.php

Mime type – a way for program/browser to identify the kind of file that’s being loaded

* <https://developer.wordpress.org/reference/functions/get_allowed_mime_types/> (different file/mime types)
* Similar to a file type extension
* Stored internally

# Javascript

### Enqueue Scripts

As always, you need to enqueue your scripts in functions.php

# Database

## WP Query

<https://developer.wordpress.org/reference/classes/wp_query/>

**WP\_Query()** is one of most used class to access db

Used to query the database for data

An instance of the class is needed to use it

# Plugins

A list of plugins that are used in this lesson

**WP Quads** – used to create ads

**Faker Press** – used for filler info

## Wordpress Popular Posts

used to see which posts have been viewed and can display which posts are popular on a page

Parameters can be found here: <https://github.com/cabrerahector/wordpress-popular-posts/wiki/2.-Template-tags#wpp_get_mostpopular>

The following code is used to output the most read/opened article using this plugin.

Customizer options were added to enable/disable the popular posts being displayed along with changing the title of the section *“Breaking News”*

This can also be found in index.php

*<!-- Using Get Most Popular Post plug-in -->*

*<?php*

*// Check to see if index is not a single page and is the home page & if function exists & if customizer setting is on*

*if( !is\_single() && is\_home()*

*&& function\_exists( 'wpp\_get\_mostpopular')*

*&& get\_theme\_mod( 'ju\_show\_header\_popular\_posts' )  ){*

*wpp\_get\_mostpopular([                         // run plugin function*

*// Starting tags of html parameter*

*'wpp\_start'     => '*

*<div class="section header-stick bottommargin-lg clearfix" style="padding: 20px 0;">*

*<div>*

*<div class="container clearfix">*

*<span class="badge badge-danger bnews-title">' . get\_theme\_mod( 'ju\_popular\_post\_widget\_title' ) . '</span> (This is where variable title is inserted)*

*<div class="fslider bnews-slider nobottommargin" data-speed="800" data-pause="6000" data-arrows="false"*

*data-pagi="false">*

*<div class="flexslider">*

*<div class="slider-wrap">*

*',*

*// Closing tags of html parameter*

*'wpp\_end'       => '*

*</div>*

*</div>*

*</div>*

*</div>*

*</div>*

*</div>*

*',*

*// Inside portion where post data will be displayed parameter*

*'post\_html'     => '*

*<div class="slide">*

*<a href="{url}">*

*<strong>{text\_title}</strong>*

*</a>*

*</div>*

*'*

*]);*

*?>*

*<?php*

*} ?>*