COMMENT TAGS

get_avatar();
comment_ID();
comment_text();
comment_type();
comment_time();
comment_date();
comment_form();
comment_class();
comments_link();
comment_author();
comment_excerpt();
comment_text_rss();
wp_list_comments();
comments_rss_link();
comment_id_fields();
comments_number();
comment_author_IP();
comment_form_title();
comment_reply_link();
next_comments_link();
comment_author_url();
comment_author_rss();
comment_author_link();
comments_author_link();
comment_author_email();
previous_comments_rss();
comments_popup_script();
permalink_comments_rss();
comment_author_url_link();
paginate_comments_links();
cancel_comment_reply_link();
comment_author_email_link();

SHORTCUT FUNCTIONS

do_shortcode();
add_shortcode();
shortcode_atts();
strip_shortcodes();
remove_shortcode();
remove_all_shortcodes();

EDIT LINK TAGS

edit_tag_link();
edit_post_link();
edit_comment_link();
edit_bookmark_link();

LIST & DROPDOWN TAGS

wp_page_menu();
wp_list_pages();
wp_get_archives();
wp_list_authors();
wp_list_comments();
wp_dropdown_pages();
wp_list_bookmarks();
wp_dropdown_users();
wp_list_categories();
wp_dropdown_categories();

LOGIN/LOGOUT TAGS

wp_logout();
wp_register();
wp_loginout();
wp_login_url();
wp_logout_url;
wp_login_form();
is_user_logged_in();
wp_lostpassword_url();
wp_registration_url();

LINKS MANAGER TAGS

get_bookmark();
get_bookmarks();
wp_list_bookmarks();
get_bookmark_field();

SHORTLINK TAGS

the_shortlink();
wp_get_shortlink();
wp_shortlink_header();
wp_shortlink_wp_head();

TITLE TAGS

wp_title();
single_cat_title();
single_tag_title();
single_post_title();
the_search_query();
single_month_title();

POST TAGS

the_ID();
the_tags();
the_title();
the_time();
the_date();
the_meta();
post_class();
body_class();
sticky_class();
the_excerpt();
the_content();
the_title_rss();
get_the_title();
the_category();
get_the_date();
the_shortlink();
the_date_xml();z
is_attachment();
wp_link_pages();
posts_nav_link();
next_post_link();
next_posts_link();
the_excerpt_rss();
the_content_rss();
next_image_link();
single_post_title();
the_category_rss();
wp_get_shortlink();
the_search_query();
the_title_attribute();
previous_post_link();
single_month_title();
the_modified_date();
the_modified_time();
previous_posts_link();
get_attachment_link();
the_attachment_link();
previous_image_link();
the_modified_author();
post_password_required();
wp_get_attachment_link();
wp_attachment_is_image();
wp_get_attachment_image();
wp_get_attachment_metadata();
wp_get_attachment_image_src();

CONDITIONAL TAGS

is_tag();
is_tax();
is_day();
is_404();
is_date();
is_time();
is_year();
is_feed();
has_tag();
is_page();
is_home();
is_admin();
is_paged();
is_single();
is_sticky();
is_sticky();
is_mouth();
is_search();
is_author();
is_archive();
is_preview();
is_singular();
is_category();
pings_open();
in_category();
is_trackback();
is_front_page();
is_attachment();
comments_open();
is_page_template();
is_comments_popup();

ATTACHEMENT TAGS

is_attachment();
the_attachment_link();
get_attachment_link();
wp_attachment_is_image();
wp_get_attachment_image();
wp_get_attachment_metadata();
wp_get_attachment_image_src();

CATEGORY TAGS

the_category();
single_cat_title();
the_category_rss();
wp_list_categories();
category_description();
wp_dropdown_categories();

NAVIGATION MENU TAGS

wp_nav_menu();
wp_get_nav_menu();
walk_nav_menu_tree();
wp_get_nav_menu_item();

PERMALINKS TAGS

the_permalink();
get_permalink();
post_permalink();
permalink_anchor();
get_post_permalink();
permalink_single_rss();

AUTHOR TAGS

the_author();
get_the_author();
wp_list_authors();
the_author_link();
the_author_meta();
the_author_posts();
get_the_author_link();
wp_dropdown_users();
get_author_posts_url();
the_author_posts_link();

PLUGIN API FUNCTIONS

has_filter();
add_filter();
apply_filters();
current_filter();
merge_filters();
remove_filter();
remove_all_filters();

do_action();
did_action();
has_action();
add_action();
remove_action();
remove_all_actions();
do_action_ref_array();

POST THUMBNAIL TAGS

has_post_thumbnail();
the_post_thumbnail();
get_post_thumbnail_id;
get_the_post_thumbnail();

QUERY TAGS

get_posts();
query_posts();
rewind_posts();
wp_reset_query();

INCLUDE TAGS

get_footer();
get_header();
get_sidebar();
get_search_form();
get_template_part();
comments_template();

TAG TAGS

the_tags();
wp_tag_cloud();
tag_description();
single_tag_title();
wp_generate_tag_cloud();

DATE & TIME TAGS

the_time();
the_date();
get_calendar();
get_the_date();
the_date_xml();
the_modified_date();
the_modified_time();
single_month_title();
the_modified_author();

BLOG INFO TAGS

bloginfo();
bloginfo_rss();
get_bloginfo();
get_bloginfo_rss();

TEMPLATES: THE BASIC FILES

styles.css	The themes' style sheet
index.php	The main page file
footer.php	The global footer file
header.php	The global header file
sidebar.php	The global sidebar file
page.php	The single page file
single.php	The single post file
archive.php	The archive and category file
searchform.php	The search form file
search.php	The search results page file
404.php	The error page file
comments.php	The comments file
tag.php	Display tags in archive format
front-page.php	Latest post or static page
category.php	Display categories in archive format

TEMPLATES: MISCELLANEOUS

<?php time_stop(1);?>
Display pageload time

<?php include (TEMPLATEPATH.'/***');?>
Include files from theme folder

<?php _e('Message');?>
Prints out message

<!--next page-->
Divides content into pages

<!--more-->
Cuts content and adds link to read the rest

<?php echo get_num_queries();?>
Queries taken to load the page

*** Replace with filename

INFOGRAPHIC BY

make a websitehub.com

TEMPLATES: THE INCLUDE TAGS

```
<?php get_header(); ?>
<?php get_started(); ?>
<?php get_footer(); ?>
<?php comments_template(); ?>
```

TEMPLATES: THEME DEFINITION

```
/"
Theme Name: WordPress Theme
Theme URI: makeawebsitehub.com
Description: makeawebsitehub.com
Version: 1.1
Author: Jamie Spencer
Author URI: makeawebsitehub.com
Tags: clean, white, fresh, blog
*/
```

TEMPLATES: THE LOOP

```
Basic Loop
<?php if(have_posts()) { ?>
<?php while(have_posts()) { ?>
<?php the_post(); ?>
<?php // custom post content code
for title, excerpt and featured image ?>
<?php } // end while ?>
<?php } // end if ?>
```

TEMPLATES: THE HEADER TAGS

<?php bloginfo('name');?>	The blog's title
<?php bloginfo('url');?>	The blog's URL
<?php bloginfo('stylesheet_url');?>	The stylesheet location
<?php bloginfo('template_url');?>	The location of the theme's files
<?php bloginfo('atom_url');?>	The Atom URL of the blog
<?php bloginfo('rss2_url');?>	The RSS2 URL for the blog
<?php bloginfo('charset');?>	Charser of the blog
<?php bloginfo('pingback_url');?>	Pingback URL of the blog
<?php bloginfo('version');?>	WordPress version of the blog
<?php bloginfo('html_type');?>	HTML version of the site
<?php site_url(); ?>	The root url for website
<?php wp_title(); ?>	The title of specific post/page
<?php bloginfo('stylesheet_url');?>	The site description
<?php get_stylesheet_directory(); ?>	The stylesheet folder location

TEMPLATES: THE TEMPLATE TAGS

Pages and Posts

<?php the_title() ?>	Displays the title of the page or post
<?php the_content() ?>	Displays the content of the page or post
<?php the_excerpt() ?>	Displays an excerpt of the page or post
<?php the_date() ?>	Displays the published date of the page or post
<?php the_time() ?>	Displays the published time of the page or post
<?php the category() ?>	Displays the category of the post
<?php the_permalink() ?>	Displays the URL for the page or posts permalink
<?php the_ID(); ?>	Displays the name of the author of the post
<?php the_author(); ?>	Displays previous oage and next page links
<?php posts_nav_link(); ?>	Displays a newer posts link
<?php previous_post_link() ?>	Displays a previous posts link
<?php edit_post_link(__('Edit Post')); ?>	Displays a edit link
<?php the_search_query();?>	Displays the search form value

Lists

<?php wp_list_pages(); ?>	Displays the pages
<?php wp_tag_cloud(); ?>	Displays the tag cloud
<?php wp_list_cats(); ?>	Displays the categories
<?php get_calendar(); ?>	Displays the calendar
<?php wp_get_archives() ?>	Displays the archives by date
<?php get_links_list(); ?>	Displays the blogroll links

Admin and Login

<?php wp_register();?>	Displays a register link
<?php wp_loginout();?>	Displays a log in/log out link
<?php wp_metal();?>	Displays the admin meta

TEMPLATES: NAVIGATION MENU

Default Navigation Menu

```
<?php wp_nav_menu(); ?>
Specific Navigation Menu
<?php wp_nav_menu( array('menu' => 'Project Nav' )); ?>
```

Category Based Navigation

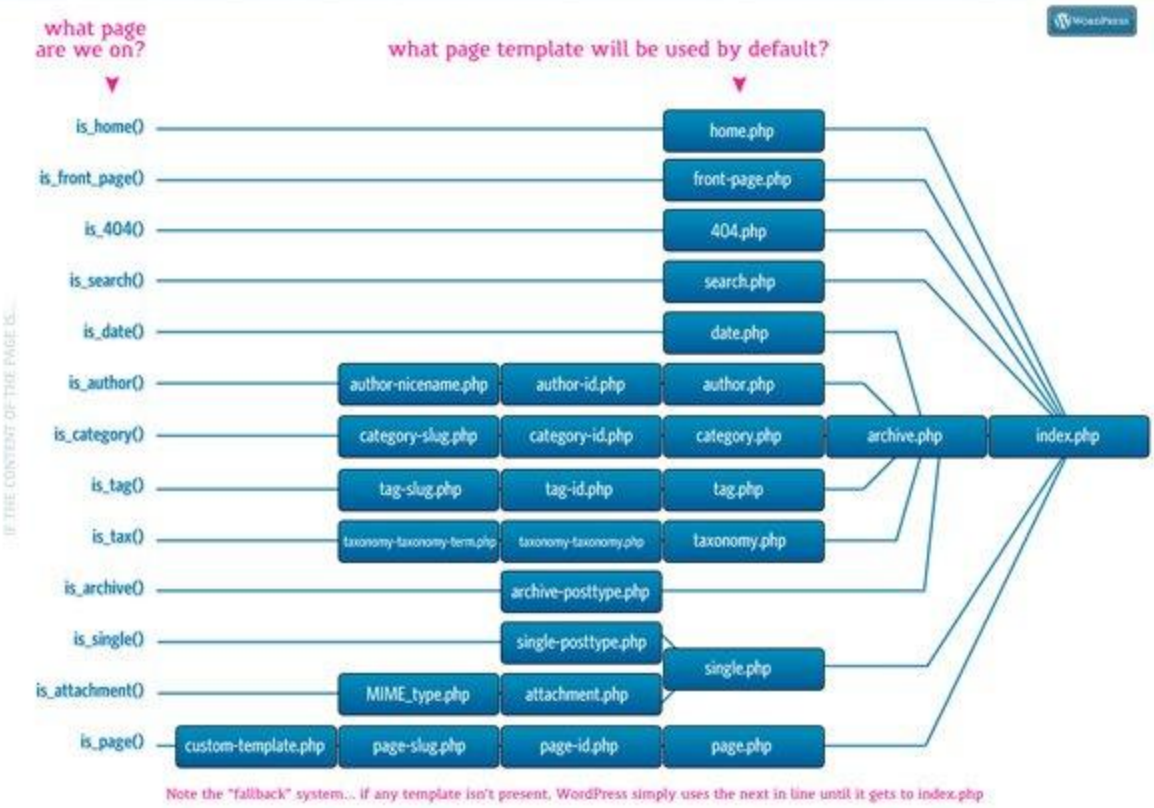
```
<ul id="menu">
<li <?php if(is_home()) { ?> class="current-cat" <?php } ?>>
<a href="<?php bloginfo(&utmx=g_organic'home'); ?>">Home</a></li>
<?php wp_list_categories('title_li=&orderby=id');?>
</ul>
```

Page Based Navigation

```
<ul id="menu">
<li <?php if(is_home()) { ?> class="current-page-item" <?php } ?>>
<a href="<?php bloginfo(&utmx=g_organic'home'); ?>">Home</a></li>
<?php wp_list_pages('sort_column=menu_order&depth=1&title_li=');?>
</ul>
```

TEMPLATES: EXTRA FUNCTIONS

/%postname%/	custom permalinks
<?php include(TEMPLATEPATH . '/x'); ?>	include file from template folder
<?php the_search_query(); ?>	value returned from search from
<?php _e('Message'); ?>	return translated text from translate()
<?php wp_register(); ?>	register link
<?php wp_loginout(); ?>	login/logout link
<!--nextpage-->	divide content into pages
<!--more-->	cut off content and create link to full post
<?php wp_meta(); ?>	admin meta data
<?php timer_start(); ?>	start page timer (header.php)
<?php timer_stop(1); ?>	time to load the page (footer.php)
<?php echo get_num_queries(); ?>	show queries executed to generate page


what page are we on?
what page template will be used by default?
is_home()
is_front_page()
is_404()
is_search()
is_date()
is_author()
is_category()
is_tag()
is_tax()
is_archive()
is_single()
is_attachment()
is_page()
home.php
front-page.php
404.php
search.php
date.php
author-nicename.php
author-id.php
author.php
category-slug.php
category-id.php
category.php
archive.php
index.php
tag-slug.php
tag-id.php
tag.php
taxonomy-taxonomy-term.php
taxonomy-taxonomy.php
taxonomy.php
archive-posttype.php
single-posttype.php
single.php
MIME_type.php
attachment.php
custom-template.php
page-slug.php
page-id.php
page.php
IF THE CONTENT OF THE PAGE IS...
Note the "fallback" system... if any template isn't present, WordPress simply uses the next in line until it gets to index.php

WordPress Security

1. Checking User Capabilities
2. Data Validation
3. Securing (sanitizing) Input
4. Securing (escaping) Output
5. Nonces
6. XML-RPC issue
7. Directory listing issue
8. Internal path disclosure
9. Server signature
10. SQL Injection
11. Clickjacking
12. WordPress version
13. Login URL exposed
14. Robots.txt Disclosure

1.Keep WordPress Updated:

2.Use Strong Passwords:

3.Limit Login Attempts:

4.Implement Two-Factor Authentication (2FA):

5.Secure Hosting Environment:

6.Use SSL/TLS Encryption:

7.Regular Backups:

8.Use Trusted Themes and Plugins:

9.Limit File and Directory Permissions:

10.Monitor for Suspicious Activity:

11.Install Security Plugins:

12.Implement a Web Application Firewall (WAF):

White Screen of Death" (WSOD), is a common issue

- **Disable plugin**
- **Change your default theme**
- **Increase php memory size**
- **Checking Error log**
- **Enable WP Debug mode**
- **Enable debug log**
- **Enable Debug Display**
- **Enable Script debug**

- **Checking WP Debug php error**
- **Check for Plugin or Theme Conflicts:**
- **Increase PHP Memory Limit:**
- **Check for Syntax Errors:**
- **Server Error Logs:**
- **Corrupted Core Files:**
- **Browser Cache:**
- **Disable and Re-enable Theme or Plugin:**
- **WordPress Debugging Tools:**
- **Consult Hosting Support:**

define('WP_DEBUG', true);

define('WP_DEBUG_LOG', true);

define('WP_DEBUG_DISPLAY', false);

This will log errors to a debug.log file in the /wp-content/ directory.

Check for Plugin or Theme Conflicts:

Deactivate all plugins by renaming the plugins folder inside the /wp-content/ directory to something else (e.g., plugins_old). If this resolves the issue, it means one of your plugins is causing the problem. Reactivate them one by one to identify the problematic plugin.
If the issue persists after disabling plugins, switch to a default WordPress theme (e.g., Twenty Twenty-One) by renaming your current theme's folder inside the /wp-content/themes/ directory.

Increase PHP Memory Limit:

A lack of memory can sometimes cause the white screen. Increase your PHP memory limit by adding the following line to your wp-config.php file:

define('WP_MEMORY_LIMIT', '256M');

Check for Syntax Errors:
Review recent changes to your theme or plugin files. Syntax errors in PHP code can cause a white screen.

Server Error Logs:
Check your server's error logs (typically found in cPanel, Plesk, or through SSH access). Look for any PHP errors or server-related issues that might be causing the white screen.
Database Issues:
Sometimes database problems can lead to a white screen. Ensure your database credentials in wp-config.php are correct, and your database server is running.

Corrupted Core Files:
If none of the above solutions work, it's possible that some of your core WordPress files may have become corrupted. You can try replacing the core files with fresh copies.

Browser Cache:
Clear your browser cache and try accessing the website from a different browser or device to ensure the issue is not browser-related.

Disable and Re-enable Theme or Plugin:
In some cases, temporarily deactivating and reactivating the theme or plugin causing the issue might resolve it.

Consult Hosting Support:
If you're on a shared hosting environment, contact your hosting provider's support. They might be able to identify server-specific issues causing the white screen.

WordPress Debugging Tools:
You can use tools like the Query Monitor plugin or WP_DEBUG to dig deeper into what's causing the issue.

Performance of a WordPress website

Page, Browser, Object, Data Cache,
Use a Lightweight and Optimized Theme:
Limit and Optimize Plugins:
Indexing & Optimize Database:
CDN Setup and Fix Insecure Content
Conditionally enqueue JS/CSS
Cache Preloading
Minifying JS/CSS/HTML
Gzip File Compression
Optimize Core Web Vitals (CLS, LCP and FCP)
Fonts Preloading
Lazy Loading Images and iFrame

Deferring Unused JS/CSS
Inline & Combine JS/CSS
Font Optimization
Image Optimization
Serving Scaled Images
WooCommerce Optimization
Choose a Fast and Reliable Hosting Provider

Advance Performance WP

1. Caching:

- **Object Caching:** Implement an object caching mechanism using tools like Memcached or Redis to reduce the load on the database and speed up queries.
- **Page Caching:** Use a page caching plugin (e.g., W3 Total Cache, WP Super Cache) to serve static HTML versions of pages, reducing server load and improving response times.
- **Browser Caching:** Configure HTTP headers to enable browser caching for static assets (CSS, JavaScript, images) to reduce the need for repeated downloads.

2. Content Delivery Network (CDN):

- Utilize a CDN service like Cloudflare or Amazon CloudFront to distribute content globally, reduce server load, and improve page load times by serving cached content from edge servers.

3. Database Optimization:

- Optimize database queries by using tools like Query Monitor or the Debug Bar to identify and fix slow queries.
- Implement database indexing and use a performance optimization plugin to clean up and optimize database tables.

4. Lazy Loading:

- Implement lazy loading for images and videos to defer the loading of off-screen content, reducing initial page load times.

5. Code Optimization:

- Minimize and concatenate CSS and JavaScript files to reduce the number of HTTP requests.
- Optimize images and use responsive images with the `srcset` attribute.
- Use asynchronous and deferred loading for non-essential JavaScript.
- Remove unnecessary plugins and code.
- Profile and optimize PHP code and functions.
- Implement object-oriented programming (OOP) and code modularization for better code management and performance.

6. Server and Hosting Optimization:

- Choose a high-performance hosting environment with optimized server configurations, such as PHP opcode caching (e.g., OPCache), server-level caching (e.g., Varnish), and HTTP/2 support.
- Consider using a managed WordPress hosting provider that specializes in high-traffic sites and offers scalable solutions.

7. Content Optimization:

- Compress and optimize images and other static assets to reduce file sizes.
- Minimize the use of external scripts and third-party resources.
- Optimize and reduce the number of external HTTP requests (e.g., for social media widgets or tracking scripts).

8. Content Management:

- Implement content management strategies to reduce the number of posts on the homepage and archive pages.
- Use excerpts and pagination to split long content into smaller, more manageable sections.

9. Load Testing and Performance Monitoring:

- Conduct load testing using tools like Apache JMeter or LoadImpact to simulate high traffic and identify potential bottlenecks.
- Set up performance monitoring using tools like New Relic, Pingdom, or Google PageSpeed Insights to continuously track website performance and receive alerts about issues.

10. High Availability and Redundancy: - Implement redundancy and failover mechanisms to ensure high availability and minimize downtime, such as using load balancers and multiple web servers.

11. Content Scaling: - Consider offloading resource-intensive tasks, such as media storage, to third-party services like Amazon S3 or a dedicated media server.

12. Content Delivery: - Use a Content Delivery Network (CDN) to serve static assets and cached content from edge locations closer to your users.

13. Content Caching: - Employ full-page caching and object caching to reduce the load on your web server and database.

14. Content Distribution: - Implement a Content Delivery Network (CDN) to distribute content globally and reduce server load.

15. Database Scaling: - If your site relies heavily on the database, consider database scaling techniques like master-slave replication or database sharding to distribute the database workload.

16. Monitoring and Scaling: - Continuously monitor server performance and traffic patterns. Set up automated scaling solutions to handle traffic spikes.

17. Content Preloading: - Use techniques like preloading popular pages or objects to reduce load times for frequently accessed content.

Htaccess file importance

- **Setting Custom 404 page**
- **Password Protect a Directory**
- **restricted ip lohin**
- **Increasing the speed of website**
- **Rewrite URLs**
- **Enable Browser Caching**
- **File compression**
- **Redirect http URLs to https**
- **Redirect non www URLS to www & vice versa**
- **Removing trailing slash from URLs**
- **Redirecting one URL to another URL**
- **Creating SEO friendly URL**

Redirect http URLs to https
RewriteEngine On
RewriteRule ^(.*)$ https://example.com/$1 [L,R=301]

www to non-www
RewriteCond %{HTTPS} on
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ https://%1/$1 [R=301,L]

non-WWW to WWW
RewriteCond %{HTTPS} on
RewriteCond %{HTTP_HOST} !^www\.(.*)$ [NC]
RewriteRule ^(.*)$ https://www.%{HTTP_HOST}/$1 [R=301,L]

Add trailing slash
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_URI} !(.*)/$
RewriteRule ^(.*)$ $1/ [L,R=301]

Remove trailing slash
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_URI} (.*)/$
RewriteRule ^(.*)/$ $1 [L,R=301]

Disallowing directory listing
Options -Indexes

Redirect one page to another
Redirect 301/old-page https://example.com/new-page
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^article/([^/]+)$ /article.php?id=$1 [L]
ErrorDocument 404 https://example.com/custom-404

For password protected directory or protected site
AuthType Basic
AuthName "Protected Area"
AuthUserFile /path/to/password/file
Require valid-user

Enable browser caching
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/jpg "access plus 1 year"
ExpiresByType image/jpeg "access plus 1 year"
ExpiresByType image/gif "access plus 1 year"
ExpiresByType image/png "access plus 1 year"
ExpiresByType text/css "access plus 1 month"
ExpiresByType text/html "access plus 1 month"
ExpiresByType application/pdf "access plus 1 month"
ExpiresByType text/x-javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType image/x-icon "access plus 1 year"
ExpiresDefault "access plus 2 days"
</IfModule>

Core Web Vitals

Core Web Vitals are a set of specific website performance metrics that Google considers important for delivering a positive user experience. These metrics measure aspects of web page loading speed, interactivity, and visual stability. Google uses Core Web Vitals as part of its ranking algorithm to assess and prioritize websites that provide a better user experience. The three core metrics that make up Core Web Vitals are:

1.Largest Contentful Paint (LCP):

LCP measures the loading speed of the largest element visible within the viewport of a web page. It reflects the time it takes for the main content of a page to become visible to the user. To provide a good user experience, LCP should occur within the first 2.5 seconds of the page starting to load.

2.First Input Delay (FID):

FID measures the time delay between when a user interacts with a web page (such as clicking a button or link) and when the browser is able to respond to that interaction. FID measures the interactivity and responsiveness of a website. To deliver a good user experience, FID should be less than 100 milliseconds.

3.Cumulative Layout Shift (CLS):

CLS measures the visual stability of a web page during its loading phase. It quantifies how much the content of a page moves or shifts unexpectedly, causing a poor user experience. CLS is calculated by multiplying the impact fraction (the portion of the viewport affected by the layout shift) with the distance fraction (the distance the elements moved). A good user experience is indicated by a CLS score less than 0.1.

These metrics are measured and reported by tools such as Google PageSpeed Insights, Lighthouse, and the Chrome User Experience Report (CrUX). It's important for website owners and developers to optimize their web pages to improve these Core Web Vitals metrics. Optimizations may include optimizing server response times, optimizing and compressing images, minifying CSS and JavaScript, deferring unnecessary scripts, leveraging browser caching, and improving hosting and server performance.

By focusing on improving Core Web Vitals, website owners can enhance the overall user experience, potentially leading to better rankings in Google search results and increased user engagement and satisfaction.

What are the steps you will take if your WordPress file is hacked?

This hacking situation is common nowadays. The more your website grows, the more chance it will have to be hacked. On starting a new website you need to know one sure thing which is, how to create a backup of your data and WordPress core files. Without this knowledge you can get rid of the hackers but with the cost of your data which is a nightmare for every admin. So always have a backup of your website's core files and the data. Now if you have that, the rest of the tasks are easy.

- Change the password immediately.
- Remove all the suspicious users and change all of their permissions to view content only. Talk to them and Change their passwords also.
- Run a good antivirus plugin and eliminate all the malicious and infected content.
- Update all your plugins, themes, or even WordPress.
- Clean your sitemaps.
- Reinstall your plugins, themes, and WordPress core.
- Remove the files which your antivirus can't remove and restore them with your backup file.
- Activate your firewall and antivirus for future attacks.
- Ask for third-party antivirus agencies if the damage is huge and you are unable to solve it on your own.

What to do when your WordPress website hacked?

If you want to secure the WordPress data, you need to implement the security policies in WordPress, also analyze the loopholes.

1. 1. Firstly you should stay calm,
2. 2. Search for an optimal solution,
3. 3. Locate the hack,
4. 4. Contact your hosting company,
5. 5. Hire IT professional (Expertise of InfoSec),
6. 6. Restore WordPress Versions,
7. 7. Scan, Identify and reduce the malware,
8. 8. Check access permission,
9. 9. Change password and secret keys.

- Change the password immediately.
- Remove all the suspicious users and change all of their permissions to view content only. Talk to them and Change their passwords also.
- Run a good antivirus plugin and eliminate all the malicious and infected content.

- Update all your plugins, themes, or even WordPress.
- Clean your sitemaps.
- Reinstall your plugins, themes, and WordPress core.
- Remove the files which your antivirus can't remove and restore them with your backup file.
- Activate your firewall and antivirus for future attacks.
- Ask for third-party antivirus agencies if the damage is huge and you are unable to solve it on your own.

How to prevent from hacking to wordpress site?

- Install security plugins like WP security
- Re-install the latest version of WordPress
- Change password and user-ids for all your users
- Check if all your themes and plug-ins are up-to-date
- Uninstall all plugins that are downloaded from untrusted places

How to Secure WordPress?

Security in WordPress is taken very seriously, but as with any other system there are potential security issues that may arise if some basic security precautions aren't taken. You need to read "http://codex.wordpress.org/Hardening_WordPress" carefully to understand WordPress security.

How can you improve the security of your WordPress site?

- **Keep WordPress updated**: WordPress regularly releases updates that address security vulnerabilities, so keeping your WordPress installation up-to-date is crucial to ensure the security of your site.
- **Use strong passwords**: Strong passwords include letters, numbers, and special characters. You can ignore easily guessable passwords such as "Your Name" or "123456".
- **Limit Login attempts**: This can help prevent brute force attacks. You can use a plugin like Limit Login Attempts Reloaded to limit the number of login attempts.
- **Use a secure hosting provider**: Choose a hosting provider that has a good reputation for security and provides SSL encryption.

- **Use two-factor authentication (2FA)**: Two-factor authentication adds an extra layer of security to your site. You can use a plugin like Google Authenticator or Authy to enable 2FA.
- **Use security plugins**: Security plugins like Wordfence, Sucuri, or iThemes Security can help to identify and block potential security threats.
- **Remove unnecessary plugins and themes**: Unnecessary plugins and themes can pose a security risk. You should delete any plugins and themes you don't use.
- **Regularly back up your site**: Regularly backing up your site can help you restore your site in case of a security breach or a website crash.

How do I update my wordpress website?

1. One-click update

All versions of WordPress from 2.7 onwards have this feature. You can update by going to the Dashboard and clicking "Updates" screen. On "Update WordPress," click "Update Now" for initiating the process.

2. Manual Update

- Replace your WordPress files
- Download and unpack the latest zip file.
- Deactivate plugins and delete old wp-includes and wp-admin directories on your host.
- Using FTP, upload new wp-includes and wp-admin directories.
- Upload individual files from a new wp-content folder, and overwrite existing files. Do NOT delete existing wp-content folder or files and folders in the current wp-content directory.
- Next, upload new files from the root directory of the latest version to your current directory.

List some features of WordPress.

WordPress powers more than 28% of the web and this figure is not limited it rises every day.

- It's Simplicity
- Easier publishing tools
- Search Engine Optimized
- User Management
- Media Management

- Easy Theme System
- Easy Extendability with Plugins
- Multilingual Support
- Easy Installation and Upgrades
- Multisite Support
- Built-in Comments System
- Custom Content Types

List the positive aspects of WordPress?

- It is a very easy installation process and up gradation of it.
- It has in-built SEO engine.
- It has many easy theme system
- Multilingual- available in more than 70 languages, which helps to provide easy understanding for all website users.
- By using WordPress, a user can own data- no unwanted advert on your website
- Flexibility and Easy publishing option.

What are disadvantages of WordPress?

- Use of multiple plugins can make website heavy to load and slow
- Sometimes updates can lead to loss of data, so you always need a backup copy
- Modifying images and tables are difficult.
- Only utilizes PHP
- Using many plugins make website heavy to load and run, which makes performance issue in certain conditions.
- PHP knowledge is a must and required to do any changes on the WordPress website.
- To avoid from hacking, WordPress's software needs to be updated to keep it up to date with all its current browsers and mobile devices. And updating of WordPress versions may lead to loss of data and hence it requires the backup copy of website every update.
- Changing and formatting the graphic images and tables is a little difficult.

What are the difference between action hook and filter in WordPress?

Actions Hook

Actions Hook are triggered by particular events that take place in WordPress such as changing themes, publishing a post, or displaying an administration screen. It is a custom PHP function defined in your plugin and hooked, i.e., set to respond, to some of these events.

Actions offently do one or more of the following things

- Modify database data
- Send an email message
- Modify the generated administration screen or front end page sent to a user web browser.

Below are list of some **Action hooks** functions

- has_action()
- add_action()
- do_action()
- do_action_ref_array()
- did_action()
- remove_action()
- remove_all_actions()
- doing_action()

Filters Hook

Filters Hook are functions that WordPress passes data through, at certain points in execution, just before taking some action with the data. It sits between the database and the browser and between the browser and the database; all most all input and output in WordPress pass through at least one filter hook.

The necessary steps to add your filters to WordPress are listed:

- Create the PHP function that filters the data
- Hook to the screen in WordPress, by calling add_filter()
- Put your PHP function in a plugin file and activate it.

Below are list of some **Filter hooks** functions

- has_filter()

- add_filter()
- apply_filters()
- apply_filters_ref_array()
- current_filter()
- remove_filter()
- remove_all_filters()
- doing_filter()

Define what are rules to follow in WordPress plugin development?

- Find a unique name
- Setup a prefix (related to your brand)
- Create the plugin's folder
- Create sub-folders for PHP files, assets, and translations
- Create the main plugin file and fill in obligatory header information
- Create a readme.txt file
- Use proper constants and functions to detect paths to plugin files
- Create additional PHP files and include them inside the main one
- Create activation and deactivation functions
- Create an uninstall script

How to display menu in WordPress?

```
wp_nav_menu( array( 'theme_location' => 'header-menu' ) );
wp_nav_menu( array $args = array(
                                'menu' => "",
                                'menu_class' => "",
                                'menu_id' => "",
                                'container' => "",
                                'container_class' => "",
                                'container_id' => "",
                                'fallback_cb' => "",
                                'before' => "",
                                'after' => "",
                                'link_before' => "",
                                'link_after' => "",
                                'echo' => "",
                                'depth' => "",
                                'walker' => "",
                                'theme_location' => "",
                                'items_wrap' => "",
                                'item_spacing' => "",
                                )
            );
```

How many default tables are in WordPress?

- wp_options
- wp_users
- wp_links
- wp_commentmeta
- wp_term_relationships
- wp_postmeta
- wp_posts
- wp_termmeta
- wp_term_taxonomy
- wp_usermeta
- wp_terms
- wp_comments

How to get a website URL in WordPress?

get_site_url(int $blog_id = null, string $path = '', string $scheme = null)

All parameters are optional.

This command is used to retrieve URLs for a given website with WP application files accessible.

echo get_site_url();

How we can change table prefix (wp_) in WordPress?

- **Before installation** : Go to `wp_config.php` file and rename `$table_prefix = 'wp_';` whatever you want.
- **After installation** : Go to phpmyadmin -> select your databse -> check all tables -> Add/Update table prefix.

What are the importers in WordPress?

In WordPress, an importer is a tool, which is used to migrate content from an existing WP website to another one. This tool can also be used to move a website from the localhost, i.e. local server to an online hosted server.

Importers are plugins that give usefulness to import a mass XML document with any number of records. It empowers to import Posts, Page, Custom Posts and Users information in an XML record.

Using WordPress importer, you can migrate different data like:

- New Posts, pages, or other custom created post types
- Comments in posts
- Custom fields and post metadata
- Categories, tags, and other terms from custom taxonomies section
- Authors on the website

You can call shortcode function like this

```
function display_related_posts($array = array())

    {
    return 'BestInterviewQuestion.com';
    }

add_shortcode('display_related_posts', 'display_related_posts');


// echo do_shortcode('[display_related_posts]');
```

What is wp_footer() in WordPress?

It is a type of action hook where the code is dynamically added to a theme in between footer tag. With the help of these functions, we can insert plugins additional code into footer tag.

How to add custom dynamic sidebars in WordPress?

```
$args = array(
            'name' => __( 'Sidebar name', 'theme_text_domain' ),
            'id' => 'unique-sidebar-id',
            'description' => '',
            'class' => '', 'before_widget' => '<li id="%1$s" class="widget %2$s">',
            'after_widget' => '</li>',
            'before_title' => '<h2 class="widgettitle">',
            'after_title' => '</h2>'
            );
```

How does WordPress interact with databases?

There are many ways when WordPress can interact with databases like using functions like **get_posts(), wp_get_post_terms(), wp_query()**, and **$wpdb()**.

From all these functions the most effective function to get a database table out of WordPress is the **$wpdb()** function. $wpdb is a global function and hence is declared with global keyword.

global $wpdb;

Database query on WordPress

For progressively fundamental questions, for example, choosing data from a table, see the other WordPress database functions such as

- **get_results()**
- **get_var()**
- **get_row()**
- **get_col()**

When Should You Not Recommend WordPress to a Client?

WordPress has many advantages but sometimes it can get costly for small businesses. This happens when you have a big website and less knowledge to even edit content.

WordPress, CMS requires regular maintenance every 2-3 months for which you need to hire a WordPress developer. These developers will make your website more of a code reliable than plugins reliable. When this happens you will not be able to customize and

edit pages and content by yourself and hire more developers instead which will increase the cost of the project.

The second thing is when your website deals with a piece of sensitive information or you are worried more about your website's security, you should not use WordPress because it is prone to attacks by hackers. 1/3rd of all websites are created on WordPress and hence it becomes popular among hackers too since they have too many targets.

What are the template tags in WordPress?

A template tag <template> is a container that is used to store the hidden HTML syntax from the users while loading a page. You use it in JavaScript.

How to take backup of our WordPress website?

There are three ways to back up a WordPress website thoroughly.

1. Backup Through Hosting

Hosting providers offer service that completes daily, complete backups to your WordPress website.

2. Backup Files Manually

We have to download the entire WordPress directory via cPanel or SFTP program. We can also use WordPress database to backup website data manually.

3. Automated Backup with Plugins

There are many secure and easy to use WordPress backup plugins available for your use. Just install, configure few settings and you are ready to go. Few of them are: BackupBuddy Premium, BackWPup Free WordPress Plugin, Duplicator Free WordPress Plugin, UpdraftPlus Backup and Restoration Free WordPress Plugin, WordPress Backup to Dropbox Free WordPress Plugin, etc.

What is a permalink in WordPress?

Permalinks are the permanent Full URLs to your website blog posts and pages, as well as your tag archives and category. It is the web address used to link to your content.

WordPress allows us to create a custom URL structure for your permalinks and archives. It can improve the usability and forward-compatibility of your website's links.

How do you create a page template?

We can create templates for various purpose like if we want to make a different layout rather than a home page or if we want to display custom posts types like news than we can create a template and merely display all news there.

Steps to create template in wordpress themes.

- Create a news.php file on root in theme folder.
- Insert `get_header();` and `get_footer();` in news.php page
- Add in top of this created page. Now this template name is "News Template".

How can you disable comments in WordPress?

Goto admin panel -> setting -> click on the Discussion tab

- Under **Default article settings** uncheck the box next to **Allow people to post comments on new articles**
- Now click on **Save Setting**

Explain Avatar in WordPress?

Avatar or Gravatar is an icon or representation of a user in a shared virtual reality, such as chat, forum, website or any other form of online community in which the user wants to have something to distinguish themselves from other users.

It usually an 80px by 80px image that the user will create by ownself.

What is Category in WordPress?

A category is one of the predefined taxonomies in WordPress, and it is used to sort and group content into different sections. In new WordPress installation, "Uncategorized" is

the default category. We can change the default category from Settings -> Writing screen. In WordPress post can be filed under many categories and can have both tags and categories.

What is Tag in WordPress? Explain

A tag is one of the pre-defined taxonomy in WordPress. You can add tags to posts along with categories in WordPress. Tags are smaller in scope and focused on specific topics.

How to change the default length of the excerpt in WordPress?

```
add_filter( 'excerpt_length', function($length) {
    return X;
    } );
```

Here above X is the number of words to be displayed.

What is the difference between Tag & Category in WordPress?

In WordPress categories are hierarchical which means terms can have a relationship with each other like child and parent.

Example: You could have a Category called India, and under India, you could have categories called Delhi, Haryana, and U.P. Categories means for broad grouping of posts.

But Tags are NOT hierarchical which means no relationship between them. Tags means to describe specific details posts.

What is custom fields in WordPress?

Custom fields are metadata used to include additional information about a post or page, like author name, title, date and time, etc. By default, the custom fields option in WordPress is hidden on the "**edit**" screen. If you want to see it, click '**Screen Options**' at the top and enable "**custom fields.**"

What is config file in WordPress?

wp-config.php is the most important file in WordPress installation. It links the database and files on your WordPress site together. Located in the root of WordPress file directory, **wp-config.php** contains configuration details, including database connection information.

What is taxonomy in WordPress? Explain

Taxonomies in WordPress are used to group posts and post types together. Custom taxonomies can help developers create groups and bring them under one head.

There are 4 in-built taxonomies in WordPress- **Category, Tag, Link Category, and Post Formats**.

Is WordPress dynamic or static?

WordPress is dynamic. This is because nearly everything in WordPress, including Pages, is generated dynamically.

How to check if any plugin is active in WordPress?

With the method `is_plugin_active()` we can check any particular plugin is active or not.

```
include_once( ABSPATH . 'wp-admin/includes/plugin.php' );
if ( is_plugin_active( 'plugin-directory/plugin-file.php' ) )
    {
    // Activated
    }
```

How is it possible to rename the WordPress folder?

Yes, we can rename the WordPress folder. If WordPress is already installed in our device, we have to log in to the weblog as the administrator and the change the required settings mentioned below.

- WordPress address (URI)
- Blog address(URI)

After the changes, we can rename the folder or directory with the use of the WordPress file in it.

What may be the reason that widgets won't display in the sidebar?

Users have to first ensure that whether the themes they are using support the widget they are wishing to add. In some cases, the problem may occur when the `function.php` or file similar to that is missing. There are chances also this could happen if the user forgets to save the changes made in the widget or to refresh the older page display.

What are the new features in WordPress 5.3?

Here is the list of new features of WordPress 5.3

- Improved features of current WordPress blocks like Block appender, changes in block
- Visual differences in the admin panel.
- Brand new blocks
- Uploading and automatic rotation of high-resolution images
- A new way of notifications via Snackbar notices.
- New default theme Twenty Twenty
- Typewriter experience
- Auto-save function

What is the difference between wp_reset_query() and wp_reset_postdata() in WordPress?

wp_reset_query()	wp_reset_postdata()
This ensures that the query has been reset to the originally created main query.	This ensures that the global $post has been restored in the current post inside the main query.
Should be used immediately after every loop using query_posts()	Should be used after every custom WP_Query()

What is wp_head() in WordPress?

It is a type of action hook where the code is dynamically added to a theme in between head tag.

<head>
<?php wp_head(); ?>
</head>

How to display an image URL from thumbnail in WordPress?

You we use `get_the_post_thumbnail_url();`.

You can use this inside loop or outside loop.

If you are using this outside of loop then you have to pass post ID otherwise its optional in case inside loop.

How to check a featured image exists or not in WordPress?

```
if ( has_post_thumbnail() )
    {
       the_post_thumbnail();
    }
```

How can you display a list of child pages in WordPress?

We can use simplay **WP_Query()** with post_parent and post_type, then we can get all child pages of that particular parent page.

```
$my_query = new WP_Query(array(
    'order' => 'ASC',
    'orderby' => 'menu_order',
    'post_parent' => 13,
    'post_type' => 'page',
    ));
if($my_query->have_posts())
{
    while($my_query->have_posts())
    {
    // display your required things
    }
}
```

How to add your custom image size for the featured image in WordPress?

Edit your existing theme's funcation.php file and add the following code

```
add_image_size( 'image_size_name', 1000, 590 );
```

How to display custom field value in page?

```
get_post_meta(get_the_ID(), 'custom_field_name', TRUE);
```

// custom_field_name is field name.

How do you create a custom taxonomy in WordPress?

```
function custom_taxonomy() {
    $labels = array(
      'name'                       => 'Taxonomy Name',
      'singular_name'              => 'Taxonomy Singular Name',
      'menu_name'                  => 'Menu Name',
      'all_items'                  => 'All Items',
      'parent_item'                => 'Parent Item',
      'parent_item_colon'          => 'Parent Item:',
      'new_item_name'              => 'New Item Name',
      'add_new_item'               => 'Add New Item',
      'edit_item'                  => 'Edit Item',
      'update_item'                => 'Update Item',
      'separate_items_with_commas' => 'Separate items with commas',
      'search_items'               => 'Search Items',
      'add_or_remove_items'        => 'Add or remove items',
      'choose_from_most_used'      => 'Choose from the considerable used items',
      'not_found'                  => 'Not Found',
      'no_terms'                   => 'No items',
      'items_list_navigation'      => 'Items list navigation',
      'items_list'                 => 'Items list',
      'back_to_items'              => 'Back to items'
    );
    $args = array(
      'labels'                     => $labels,
      'hierarchical'               => true,
      'public'                     => true,
      'show_ui'                    => true,
      'show_admin_column'          => true,
      'show_in_nav_menus'          => true,
      'show_tagcloud'              => true,
      'rewrite'                    => array( 'slug' => 'taxonomy' )
    );
    register_taxonomy( 'taxonomy', array( 'post' ), $args );
  }
  add_action( 'init', 'custom_taxonomy', 0 );
```

- Save the functions.php file.
- Go to your WordPress admin panel and click on ‘Posts’ or ‘Pages’, depending on the post type you assigned your taxonomy to.
- Click on ‘Add New’ or edit an existing post/page.
- In the right sidebar, you should now see a box labeled with the name of your taxonomy. You can add new terms by clicking the ‘+ Add New Taxonomy’ link.

How to change the homepage URL in WordPress?

Here are the steps to successfully change the Homepage URL in WordPress. Go to the function.php file in your WP theme and use this code:

```
function redirect_homepage() {
    if( ! is_home() && ! is_front_page() )
    return;
    wp_redirect( 'http://siteurl.com/news', 301 );
    exit;
 }
 add_action( 'template_redirect', 'redirect_homepage' );
```

How to display custom Post in WordPress?

```
$args = array( 'post_type' => 'blog', 'posts_per_page' => 10 );
$loop = new WP_Query( $args );
while ( $loop->have_posts() ) : $loop->the_post();
    the_title();
    echo '<div class="entry-content">';
    the_content();
    echo '</div>';
endwhile;
```

What ways to use WordPress?

WordPress can be used in many different ways. It is open to possibilities.
You can use WordPress as the following:

1. Arcade
2. Blog
3. Content Management System (CMS)
4. Gallery

5. Portfolio
6. Rating Website
7. Shopping Store
8. Video Collection Site
9. Membership Site

Template Files List for development WordPress Theme?

Here is the list of the Theme files recognized by WordPress. Of course, your Theme can contain any other stylesheets, images, or files.

style.css: The main stylesheet. This must be included with your Theme, and it must contain the information header for your Theme.

rtl.css: The rtl stylesheet. This will be included automatically if the website's text direction is right-to-left. This can be generated using the RTLer plugin.

index.php: The main template. If your Theme provides its own templates, index.php must be present.

comments.php: The comments template.

front-page.php: The front page template.

home.php: The home page template, which is the front page by default. If you use a static front page this is the template for the page with the latest posts.

single.php: The single post template. Used when a single post is queried. For this and all other query templates, index.php is used if the query template is not present.

single-{post-type}.php

The single post template used when a single post from a custom post type is queried. For example, single-book.php would be used for displaying single posts from the custom post type named "book". index.php is used if the query template for the custom post type is not present.

page.php: The page template. Used when an individual Page is queried.

category.php: The category template. Used when a category is queried.

tag.php: The tag template. Used when a tag is queried.

taxonomy.php: The term template. Used when a term in a custom taxonomy is queried.

author.php: The author template. Used when an author is queried.

date.php: The date/time template. Used when a date or time is queried. Year, month, day, hour, minute, second.

archive.php: The archive template. Used when a category, author, or date is queried. Note that this template will be overridden by category.php,author.php, and date.php for their respective query types.

search.php: The search results template. Used when a search is performed.

attachment.php: Attachment template. Used when viewing a single attachment.

image.php: Image attachment template. Used when viewing a single image attachment. If not present, attachment.php will be used.
404.php: The 404 Not Found template. Used when WordPress cannot find a post or page that matches the query.

Does de-activated plugins slow down a WordPress site?

No, de-activated plugins cannot slow down the WordPress site. WordPress only loads the active plugins and ignores everything else.

In Define what case we cannot recommend WordPress to our client?

We cannot recommend WordPress on following situation:

- If client is working on non-CMS base project
- If site wants complex or innovative e-commerce
- In case of enterprise intranet solution
- Sites requiring custom scripting solutions.

Explain what are the essential features you look for a theme?

Theme selection differs according to the requirement, but an ideal theme would be something that would not restrict to use the number of pages, plugins or static homepage.

How Custom theme is different than Normal theme?

Custom theme allows for SEO search, but with a SEO plugin available it would not make much difference to normal theme. One benefit using the Custom theme is that it allows to make the changes without going much into the coding part.

How you can create a static page with WordPress?

To create a static page in wordpress, in the page section you have to upload a php files to the server in the theme folder, and then select that as your template.

This allows you to add any page and look that you wanted for your blog and it will remain static.

Define what are meta-tags?
Meta-tags are keywords and description used to display website or page information.

What should one use for plugin development — custom post types or custom database tables?
There is no specific preference for plugin development; it depends on Define what type of plugin's one has to develop. Though few recommend custom post type, as it has few benefits comparison to custom database table.

can use organize series plugin to write series in wordpress.

Define what are the reasons why one should not hack WordPress core file?
The best reason not to hack the core files is that Define whatever you might be doing has to be reworked as a patch.

In which cases you don't see plugin menu?
You can't see your plugin menu when the blog is hosted on free wordpress.com as you cannot add plugin there. Also, if you do not have an account of an administrator level on your WordPress dashboard, it is not possible to see plugin.

Define what is the difference between the wp_title and the_title tags?
wp_title(). function is for use outside "The Loop" to display the title of a Page.

the_title(). on the other hand is used within "The Loop".

Define what do next_posts_link(. and previous_posts_link(. do?
Because post queries are usually sorted in reverse chronological order, next_posts_link(. usually points to older entries (toward the end of the set. and previous_posts_link(. usually points to newer entries (toward the beginning of the set).

How to create mailchimp or vertical response campaign for newsletter subscribers & link with WordPress ?

First Create List & Campaign on mailchimp/ WordPress account . Then subscribe users from WordPress in mailchimp list by plugin or manual hard code webform.

Define what options are there to implement a multi language site ?

WordPress has no bi/multi-language feature built in by default. The other road would be to extend the site with Add-ons like plugins and themes that have WPML multi-language features built-in.

How can I stop WordPress from prompting me to enter FTP information when doing updates?

If you edit your wp-config.php file you can preload these FTP settings as constants read by WordPress. Keep in mind, on a shared host, you should be mindful of possible security implications.

Define what's the difference between site_url(. and home_url()?

The site_url(. will always be the location where you can reach the site by tacking on /wp-admin on the end, while home_url(. would not reliably be this location.

The home_url(. would be where you have set your homepage by setting General > Settings "Site Address (URL)" field.

The WordPress Address is where to look for WordPress files, while the Site Address is what will be used as the base for creating URLs for web pages.

Difference Between Posts vs. Pages

1. Posts can be categorized vs. Pages are hierarchical.
2. Pages have custom template feature vs. Posts do not.
3. Posts are timely vs. Pages are timeless.
4. Posts are included in RSS feed vs. Pages are not.
5. Posts are social vs. Pages are NOT.

How to store the post title in variable?

$page_title = get_the_title($post->post_parent);

Can wordPress use cookies?

Yes, wordpress use cookies.WordPress uses cookies, or tiny pieces of information stored on your computer, to verify who you are. There are cookies for logged in users.

Define what is the special meaning of _sleep and _wakeup?

_sleeps return the array of all variables than need to be saved,

while _wakeup retrives them.

Which 'meta box' is not hidden by default on Post and Page screens?

Featured Image is the **meta box** that is not hidden by default on Post and Page screens.

In WordPress, objects are passed by value or by reference.

In WordPress, all objects are passed by value.

Why wordpress is better than blogger for blogging?

1. Customization and Flexibility
2. Hosting Opportunities
3. To customize the tiny image that people see in their address bar when they visit your blog, upload a blavatar.
4. The more tag allows you to display excerpts of your posts on your main posts page, instead of revealing the entire post.
5. To expand the text editor and hide the modules on the publishing screen, enable Distraction Free Writing.
6. Reproduce fully-functioning tweets — not just static screenshots of tweets — in posts, pages, and even comments with Twitter Embeds.
7. You can customize your post and page slugs, which can be handy when you want to create a URL that's easier to remember (it's also a good way to make posts search-friendly). Note that if you change a post slug, the old link will still work.

8. Reproduce fully-functioning tweets — not just static screenshots of tweets — in posts, pages, and even comments with Twitter Embeds.

How to get our WordPress site on Google?

If you want your WordPress site on Google then these following steps help you:

- 1. Firstly go to the settings.
- 2. Scroll down to the Search Engine Visibility option to the Reading page.
- 3. Check your Settings.
- 4. Install the Yoast SEO Plugin.
- 5. Link your website to the Google Search Console.
- 6. For this, you have to click Add Property in your Google Search Console account.
- 7. Add your Website URL.
- 8. Set up XML sitemaps.
- 9. Submit XML Sitemaps on Google Search Console.

What is usermeta function in WordPress?

The user metafunction is used to retrieve the metadata of users. It can return a single value or an array of metadata.

Syntax is: get_user_meta(int $user_id, string $key = ”, bool $single = false)

User id is the required user id parameter

Key is the optional parameter which is the meta key to retrieve. By default, it returns data for all key values.

Single is an optional parameter that tells whether the single value will return. By default, it is false.

Creating a child theme in WordPress

1.Create a New Child Theme Directory:

Create a new directory for your child theme in the wp-content/themes/ directory. Name the directory something unique, preferably related to your child theme.

For example, if your parent theme is named "MyTheme," you can name your child theme directory as "MyTheme-child."

2.Create the style.css File:

•Inside the child theme directory, create a new file named style.css.

•Open the style.css file and add the following header information:

```
/*
Theme Name: MyTheme Child
Theme URI: http://example.com/mytheme-child/
Description: Child theme for MyTheme
Author: Your Name
Author URI: http://example.com
Template: mytheme
Version: 1.0
Text Domain: mytheme-child
*/
```

•Make sure to replace "MyTheme" with the name of the parent theme.

3.Enqueue the Parent Theme Stylesheet:

•Still in the style.css file of your child theme, add the following code to enqueue the parent theme's stylesheet:

```
/* Theme customization starts here */

/* Enqueue parent theme stylesheet */

@import url('../mytheme/style.css');
```

•Make sure to adjust the path to the parent theme's stylesheet (in this example, it assumes the parent theme is in a directory named "mytheme").

4.Optional: Add Additional Customization:

•If you only want to modify the CSS of your child theme, you can continue adding your custom CSS rules to the style.css file.

•To modify template files or add new functionality, create files with the same names as the files you want to override from the parent theme.

•For example, if you want to modify the header.php file of the parent theme, create a header.php file in your child theme directory, and WordPress will use that file instead of the parent theme's file.

Another Way

As I mentioned, importing style sheets is the best way to enqueue them through the **functions.php** file.code.

```
1. <?php
2. add_action( 'wp_enqueue_scripts', 'enqueue_parent_styles' );
3. function enqueue_parent_styles() {
4. wp_enqueue_style( 'parent-style', get_template_directory_uri().'/style.css' );
5. }
```

Edit a Template File of single.php

We can easily edit individual template files in the child themes. For example, if we want to edit single.php, we must copy that file from the parent theme folder and paste it into the WordPress child theme's folder.

Let's say we want to remove the comments section from single post pages. After copying the file into our child theme folder, we can edit it and remove the following comments section.

```
1.  // If comments are open or we have at least one comment, load up the comment template.
2.  if ( comments_open() || get_comments_number() ) :
3.  comments_template();
4.  endif;
```

Activation and deactivation processes Plugin

```
<?php
// Security check: Exit if accessed directly
if (!defined('ABSPATH')) {
   exit;
}
   // Plugin activation tasks
register_activation_hook(__FILE__, 'your_plugin_activation');


// Deactivation hook
function your_plugin_deactivation() {


 if (!current_user_can('activate_plugins')) {
      wp_die(__('You do not have permission to deactivate this plugin.', 'your-plugin-text-domain'));
   }
   // Additional security checks or cleanup tasks
   // Plugin deactivation tasks
}
register_deactivation_hook(__FILE__, 'your_plugin_deactivation');
// Plugin installation tasks
register_installation_hook(__FILE__, 'your_plugin_ installation);
```

Steps of creating theme from scratch

1. Planning and Design:

- Define the purpose and requirements of your theme.
- Create wireframes or design mockups for your theme's layout and appearance.
- Consider user experience (UX) and responsive design for various devices.

2. Set Up Your Development Environment:

- Install WordPress locally or on a development server.
- Create a child theme (recommended) or start with a blank slate if you're creating a completely custom theme.

3. Create Theme Files and Structure:

- Create the necessary theme files, including `style.css`, `index.php`, `header.php`, `footer.php`, `single.php`, `page.php`, and any custom template files as needed.
- Organize your theme files into directories for templates, assets (CSS, JavaScript, images), and other resources.

4. Write HTML and PHP Templates:

- Use WordPress template tags and functions to insert dynamic content and functionality.
- Follow best practices for naming conventions and separation of concerns.
- Ensure that your theme is accessibility-friendly by adding appropriate ARIA roles and attributes.

5. Enqueue Styles and Scripts:

- Properly enqueue CSS and JavaScript files using the `wp_enqueue_style` and `wp_enqueue_script` functions.
- Minimize and concatenate CSS and JavaScript files for performance optimization.
- Use conditionals to load scripts only where needed (e.g., use `is_page()` or `is_single()`).

6. Implement WordPress Features:

- Add support for post formats, custom headers, and custom backgrounds as required.

- Register custom menus and sidebars using `register_nav_menus()` and `register_sidebar()`.
- Implement widget areas and customize widgetized areas.

7. Add Theme Support:

- Use the `add_theme_support()` function to enable features like post thumbnails, custom logos, and HTML5 markup support.
- Define theme support for automatic feed links and title tag.

8. Implement Customizer Settings (Optional):

- Create customizer sections and settings for theme options, allowing users to customize colors, fonts, and other design elements.
- Sanitize and validate user input to ensure security.

9. Handle Widgets and Widgets Areas:

- Register custom widgets if needed and place them in widget areas using `register_widget()` and `dynamic_sidebar()` functions.

10. Implement Custom Post Types and Taxonomies (Optional): - Create custom post types and taxonomies for content organization, if required. - Use `register_post_type()` and `register_taxonomy()` functions.

11. Implement Security Best Practices: - Validate and sanitize user inputs to prevent SQL injection and cross-site scripting (XSS) attacks. - Escape output data using functions like `esc_html()`, `esc_attr()`, and `esc_url()`. - Keep your theme and WordPress core up to date to patch security vulnerabilities.

12. Optimize Performance: - Optimize images and use responsive images for various screen sizes. - Minimize HTTP requests by reducing the number of scripts and stylesheets. - Enable browser caching and set cache-control headers. - Use a content delivery network (CDN) for faster content delivery. - Profile and optimize database queries. - Implement lazy loading for images and scripts.

13. Test and Debug: - Test your theme on different browsers and devices to ensure cross-browser compatibility and responsiveness. - Use debugging tools like the WordPress Debug Bar and browser developer tools to find and fix issues. - Validate your HTML and CSS for compliance with web standards.

14. Document Your Theme: - Provide clear documentation for theme users, including instructions for installation, configuration, and customization. - Include details about theme features, custom templates, and recommended plugins.

15. Prepare for Deployment: - Backup your website, including the database and theme files. - Perform a final security scan to ensure no vulnerabilities exist. - Export and import any necessary content and settings to the production site.

16. Deploy Your Theme: - Upload your theme to the production server. - Activate the theme on the production website. - Test all functionality and troubleshoot any issues.

17. Monitor and Maintain: - Regularly update your theme to fix bugs and security vulnerabilities. - Stay informed about WordPress core updates and adjust your theme accordingly.

Woocommerce customization

1. **Customizing WooCommerce Templates:**
    - WooCommerce provides templates that control the layout of various pages (e.g., product pages, cart, checkout).
    - You can create a child theme for your WordPress site and copy the WooCommerce templates you want to customize into your child theme directory. Make changes to these templates to match your design requirements.
2. **Adding Custom CSS:**
    - Use custom CSS to style your WooCommerce store to match your branding or design preferences.
    - You can add custom CSS in the WordPress Customizer or in your theme's stylesheet.
3. **Using Hooks and Filters:**
    - WooCommerce provides hooks and filters that allow you to add or modify functionality.
    - You can create custom functions in your theme's `functions.php` file or in a custom plugin to hook into these points and customize various aspects of your store.
4. **Adding Custom Product Data:**
    - WooCommerce supports custom product attributes and product meta data.
    - You can add custom fields to products to store additional information, such as product specifications or custom pricing.

5. **Creating Custom Checkout Fields:**
    - WooCommerce allows you to add custom fields to the checkout page to collect additional information from customers.
    - You can use plugins like "WooCommerce Checkout Field Editor" to add custom fields without coding.
6. **Customizing Product Display:**
    - You can change the number of products displayed per page and the order in which they appear.
    - Plugins and custom code can be used to create custom product layouts or grids.
7. **Integrating Payment Gateways:**
    - WooCommerce supports a variety of payment gateways by default, but you can integrate additional gateways using plugins or custom development.
8. **Extending with Plugins:**
    - WooCommerce has a rich ecosystem of plugins that can extend its functionality.
    - You can find plugins for features like subscriptions, memberships, product bundles, and more.
9. **Customizing Email Templates:**
    - WooCommerce sends various email notifications to customers and administrators.
    - You can customize these email templates to match your branding and messaging.
10. **SEO and Performance Optimization:**
    - Optimize your store for search engines by using SEO plugins.
    - Improve performance through caching, image optimization, and lazy loading.
11. **Localization and Currency:**
    - Customize your store for different regions by configuring languages and currencies.
    - WooCommerce allows you to set prices and shipping methods based on customer locations.
12. **Adding Custom Widgets and Shortcodes:**
    - Create custom widgets or shortcodes to display specific products or promotional content in your store's sidebar, footer, or content.
13. **Security and Updates:**
    - Regularly update WooCommerce and its plugins to ensure security and access to new features.
    - Implement security measures to protect customer data and transactions.

Plugin Creation Visshal programming

a.php | form_data.sql | contact_us.php

```php
register_activation_hook(__FILE__,'form_data_activate');
register_deactivation_hook(__FILE__,'form_data_dactivate');

function form_data_activate(){
    global $wpdb;
    global $table_prefix;
    $table=$table_prefix.'form_data';
    $sql="CREATE TABLE $table (
  `id` int(11) NOT NULL,
  `name` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE $table
  ADD PRIMARY KEY (`id`);
  ALTER TABLE $table
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;";
  $wpdb->query($sql);
}

function form_data_dactivate(){
    global $wpdb;
    global $table_prefix;
    $table=$table_prefix.'form_data';
    $sql="DROP TABLE $table";
    $wpdb->query($sql);
}
?>
```

```php
36
37  add_action('admin_menu','form_data_menu');
38
39  function form_data_menu(){
40      add_menu_page('Form Data','Form Data',8,__FILE__,'form_data_list');
41  }
42
43  function form_data_list(){
44      include('form_data_list.php');
45  }
46  ?>
```

```php
add_shortcode('form_data_list_shortcode','form_data_list');
function form_data_list(){
    include('form_data_list.php');
}
```

form_data.php | form_data.sql | form_data_list.php

```php
<?php
global $wpdb;
global $table_prefix;
$table=$table_prefix.'form_data';
$sql="select * from $table";
$result=$wpdb->get_results($sql);
?>
<table>
    <tr>
        <td>ID</td>
        <td>Name</td>
    </tr>
    <?php
    foreach($result as $list){
    ?>
    <tr>
        <td><?php echo $list->id?></td>
        <td><?php echo $list->name?></td>
    </tr>
    <?php
    }
    ?>
</table>
```

Display Posts from Specific Category On A WordPress Page

https://artisansweb.net/display-posts-specific-category-wordpress-page/

```
$args = array(
    'post_type' => 'post',
    'post_status' => 'publish',
    'category_name' => 'wordpress',
    'posts_per_page' => 5,
);
$arr_posts = new WP_Query( $args );

if ( $arr_posts->have_posts() ) :

    while ( $arr_posts->have_posts() ) :
        $arr_posts->the_post();
        ?>
        <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
            <?php
            if ( has_post_thumbnail() ) :
                the_post_thumbnail();
            endif;
            ?>
            <header class="entry-header">
                <h1 class="entry-title"><?php the_title(); ?></h1>
            </header>
            <div class="entry-content">
                <?php the_excerpt(); ?>
                <a href="<?php the_permalink(); ?>">Read More</a>
            </div>
        </article>
        <?php
    endwhile;
    wp_reset_postdata();
endif;
```

```
$args = array(
    'post_type' => 'post',
    'post_status' => 'publish',
    'cat' => '4', //you can pass comma-separated ids here
    'posts_per_page' => 5,
);
```

Get Posts from Custom Taxonomy

You may be working with a custom post type and wish to display the posts from a custom taxonomy. It can be done easily using the tax_query. For that, you just need to change arguments in the previous array as follows:

```php
$args = array(
    'post_type' => 'CUSTOM_POST_TYPE_NAME',
    'post_status' => 'publish',
    'posts_per_page' => 5,
    'tax_query' => array(
        array(
            'taxonomy' => 'TAXONOMY_NAME',
            'field'    => 'slug',
            'terms'    => array( 'TERM_SLUG' ),
            'operator' => 'IN'
        ),
    ),
);

$arr_posts = new WP_Query( $args );
```

template-category.php

```php
<?php
/**
 * Template Name: Category Custom Page
 */

get_header(); ?>

<div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

        <?php
   $paged = (get_query_var( 'paged' )) ? get_query_var( 'paged' ) : 1;
   $args = array(
      'post_type' => 'post',
      'post_status' => 'publish',
      'category_name' => 'wordpress',
      'posts_per_page' => 5,
      'paged' => $paged,
   );
   $arr_posts = new WP_Query( $args );
```

```
  if ( $arr_posts->have_posts() ) :

    while ( $arr_posts->have_posts() ) :
      $arr_posts->the_post();
      ?>
                  <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
                        <?php
          if ( has_post_thumbnail() ) :
            the_post_thumbnail();
          endif;
          ?>
                              <header class="entry-header">
                                    <h1 class="entry-title"><?php the_title(); ?></h1>
                              </header>
                              <div class="entry-content">
                                    <?php the_excerpt(); ?>
                                    <a href="<?php the_permalink(); ?>">Read More</a>
                              </div>
                  </article>
                  <?php
    endwhile;
    wp_pagenavi(
      array(
        'query' => $arr_posts,
      )
    );
    wp_reset_postdata();
  endif;
  ?>

      </main><!-- .site-main -->
</div><!-- .content-area -->

<?php get_footer(); ?>
```

Ddd

Increase WordPress File Upload Size Limit

1 – Using htaccess

Open your `.htaccess` file from the root directory and add the below lines after the **# END WordPress**:

```
# END WordPress

php_value upload_max_filesize 64M
php_value post_max_size 72M
```

We should always have a higher value for `post_max_size` than `upload_max_filesize`. This is because when we post the file to the server additional data may also be sent at the same time.

2 – Using Code

In this approach, you need to add the below code to your active theme's `functions.php` file.

```
@ini_set( 'upload_max_size' , '64M' );
@ini_set( 'post_max_size', '72M');
```

Users can also place this code in `wp-config.php` file.

```
@ini_set( 'upload_max_size' , '64M' );
@ini_set( 'post_max_size', '72M');
/* That's all, stop editing! Happy blogging. */
```

Check again on the Media page if you get your limit increased else remove these changes.

3 – Using PHP.INI file

For some reason, if the above options did not work for you try this one. In this case, you need to update 2 lines in your `php.ini` file. Find the strings `upload_max_filesize`, `post_max_size` and set values 64M, and 72M respectively.

4 – Using Web Hosting Tool

How to Add Image Field to Taxonomy ACF Plugin

Display Category Image On the Front End

The advanced custom fields plugin provides a method called `get_field()` to display the field values. To this function, you have to pass a field name(category_image) and `category_TERM-ID`. If you are dealing with a custom taxonomy then the second parameter would be `term_TERM-ID`. For more details, please refer to the plugin's documentation.

Let's say you have uploaded an image for the category having ID 7. Then to display the image, the code will be as follows.

```
<?php
$term_id = 7;
$image = get_field('category_image', 'category_'. $term_id); //'category_image' is our field name
echo '<img src="'.$image['url'].'" alt="'.$image['alt'].'" />';
```

Category Images

```
<?php
if (have_posts()) :
  while (have_posts()) : the_post();
    // Get the current category ID
    $category_id = get_the_category()[0]->cat_ID;
    // Get the ACF category image
    $category_image = get_field('category_image', 'category_' . $category_id);
    // Display the category image
    if (!empty($category_image)) {
      echo '<img src="' . esc_url($category_image['url']) . '" alt="' . get_cat_name($category_id) . '">';
    }
    the_title();
    the_excerpt();
  endwhile;
endif;
?>
```

How to Customize Comment Form

https://artisansweb.net/how-to-customize-comment-form-in-wordpress/

Add Field to Comment Form

WordPress provides an action hook `comment_form_after_fields` using which we can add fields to the comment form.
Let's add a mobile number using this action hook. Place the below code in the `functions.php` file.

```
add_action( 'comment_form_after_fields', 'additional_fields' );
function additional_fields() {
	echo '<p class="comment-form-mobile-number">' .
	'<label for="mobile-number">' . esc_html__( 'Mobile Number' ) . '<span
class="required">*</span></label>' .
	'<input id="mobile-number" name="mobile-number" type="text" size="30" tabindex="5" /></p>';
}
```

Example wp-config.php for Debugging

```
// Enable WP_DEBUG mode
define( 'WP_DEBUG', true );

// Enable Debug logging to the /wp-content/debug.log file
define( 'WP_DEBUG_LOG', true );

// Disable display of errors and warnings
define( 'WP_DEBUG_DISPLAY', false );
@ini_set( 'display_errors', 0 );

// Use dev versions of core JS and CSS files (only needed if you
are modifying these core files)
define( 'SCRIPT_DEBUG', true );
```

- 7 Tips for Debugging Problems in WordPress
  - 1. Activate WP_DEBUG
  - 2. Enable WPDP Error Reporting
  - 3. Check Your Website's Error Logs
  - 4. Use WordPress Staging Environment when Tweaking Your Code
  - 5. Enable SCRIPT_DEBUG
  - 6. Detect PHP Errors
  - 7. Take Advantage of Some Debugging Tools

Create a Custom Search Form

The `get_search_form()` function looks for a `searchform.php` file in your active theme's directory. If this file does not exist then WordPress uses the default search form which is included in its core.
It means you should create a `searchform.php` file in your active theme's directory. Doing so WordPress takes the search form from your theme instead of its core. In this file add your markup of the search form. In my case, the HTML is as follows.

```
<form id="searchform" method="get" action="<?php echo esc_url( home_url( '/' ) ); ?>">
	<input type="text" class="search-field" name="s" placeholder="Search" value="<?php echo
get_search_query(); ?>" />
	<input type="submit" value="Search" />
</form>
```

Let's say you have created custom post types `book`, `magazine`, `ebook`, and `pdf` for the above resources. Then, your hidden fields will be as follows.

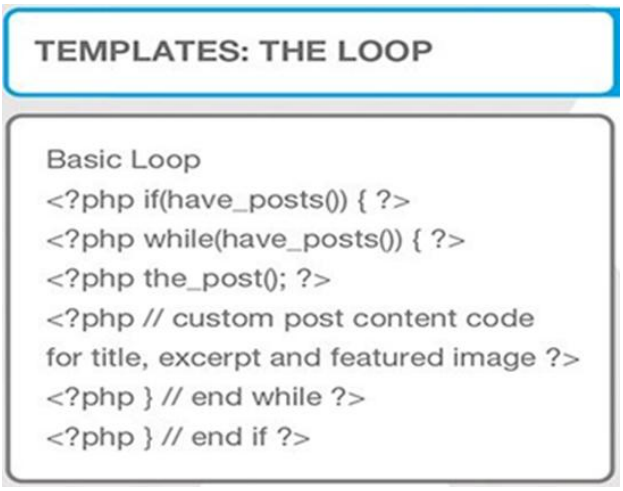
```
<form id="searchform" method="get" action="<?php echo esc_url( home_url( '/' ) ); ?>">
	<input type="text" class="search-field" name="s" placeholder="Search" value="<?php echo
get_search_query(); ?>" />
	<input type="hidden" name="post_type[]" value="book" />
	<input type="hidden" name="post_type[]" value="magazine" />
	<input type="hidden" name="post_type[]" value="ebook" />
	<input type="hidden" name="post_type[]" value="pdf" />
	<input type="submit" value="Search" />
</form>
```

WP_Query Class

Custom Query Loop

```
$args = array( 'post_type' => 'blog', 'posts_per_page' => 10 );
$loop = new WP_Query( $args );
while ( $loop->have_posts() ) : $loop->the_post();
    the_title();
    echo '<div class="entry-content">';
    the_content();
    echo '</div>';

endwhile
```

```
<?php get_template_part('template-parts/home', 'featured') ?
<?php if ( have_posts() ) : ?>
    <?php while ( have_posts() ) : the_post(); ?>
        <h1><?php the_title() ?></h1>
    <?php endwhile; ?>
<?php endif; ?>
```



Standard Loop

```
<?php

$the_query = new WP_Query( $args );

if ( $the_query->have_posts() ) {

	while ( $the_query->have_posts() ) {
		$the_query->the_post();
		echo '<li>' . esc_html( get_the_title() ) . '</li>';
	}

} else {
	esc_html_e( 'Sorry, no posts matched your criteria.' );
}
wp_reset_postdata();
```

Standard Loop (Alternate)

```
<?php

$the_query = new WP_Query( $args ); ?>

<?php if ( $the_query->have_posts() ) : ?>
	<!-- pagination here -->
```

```
        <!-- the loop -->
        <?php
        while ( $the_query->have_posts() ) :
                $the_query->the_post();
        ?>
        <?php the_title( '<h2>', '</h2>' ); ?>
        <?php endwhile; ?>
        <!-- end of the loop -->

        <!-- pagination here -->

        <?php wp_reset_postdata(); ?>

<?php else : ?>
        <p><?php esc_html_e( 'Sorry, no posts matched your criteria.' ); ?></p>
<?php endif; ?>
```

Multiple Loops

```
<?php

$query1 = new WP_Query( $args );

// The Loop
while ( $query1->have_posts() ) {
$query1->the_post();
echo '<li>' . get_the_title() . '</li>';
}

/* Restore original Post Data
* NB: Because we are using new WP_Query we aren't stomping on the
* original $wp_query and it does not need to be reset with
* wp_reset_query(). We just need to set the post data back up with
* wp_reset_postdata().
*/
wp_reset_postdata();

/* The 2nd Query (without global var) */
$query2 = new WP_Query( $args2 );

// The 2nd Loop
while ( $query2->have_posts() ) {
$query2->the_post();
echo '<li>' . get_the_title( $query2->post->ID ) . '</li>';
}
wp_reset_postdata();
?>
```

Query Sample

Refference
https://developer.wordpress.org/reference/classes/wp_query/

1. Category Parameters

- **`cat`** (*int*) – use category id.
- **`category_name`** (*string*) – use category slug.
- **`category__and`** (*array*) – use category id.
- **`category__in`** (*array*) – use category id.
- **`category__not_in`** (*array*) – use category id.

```
$query = new WP_Query( array( 'cat' => 4 ) )
$query = new WP_Query( array( 'category_name' => 'staff' ) )
$query = new WP_Query( array( 'cat' => '2,6,17,38' ) )
$query = new WP_Query( array( 'category_name' => 'staff,news' ) )
$query = new WP_Query( array( 'cat' => '-12,-34,-56' ) )
```

2. Author Parameters

- **`author`** (*int*) – use author id.
- **`author_name`** (*string*) – use '`user_nicename`' – NOT name.
- **`author__in`** (*array*) – use author id (available since version 3.7).
- **`author__not_in`** (*array*) – use author id (available since version 3.7).

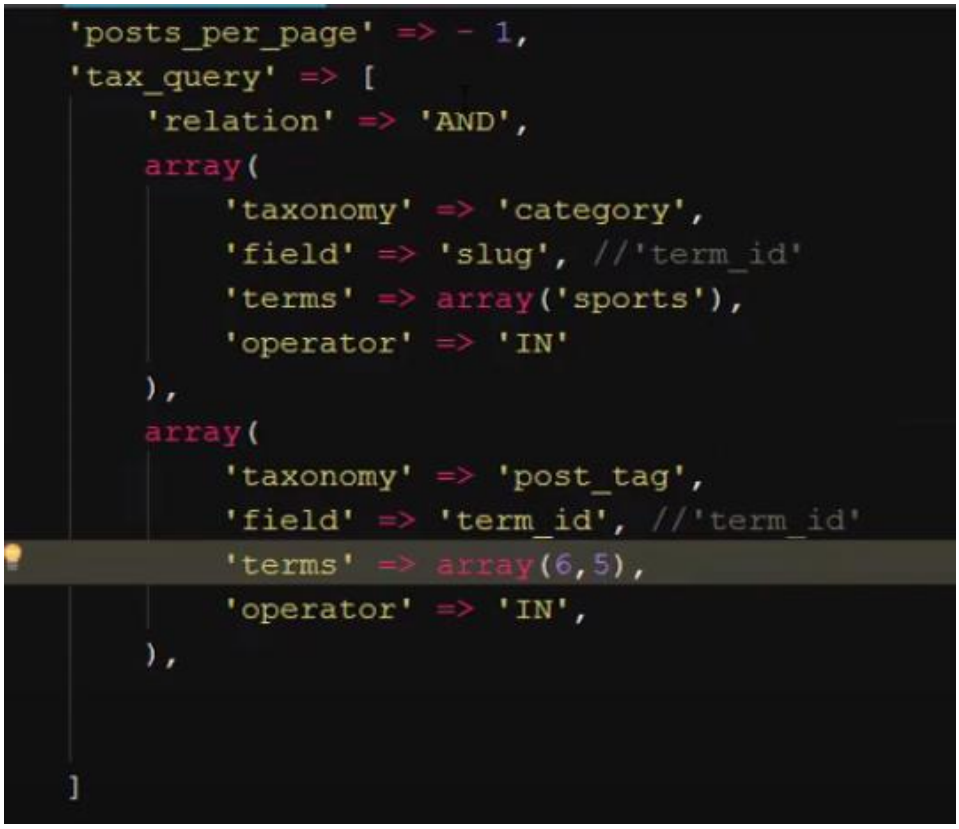
```
$query = new WP_Query( array( 'author' => 123 ) );
$query = new WP_Query( array( 'author' => '2,6,17,38' ) );
$query = new WP_Query( array( 'author' => -12 ) );
```

3. Tag Parameters

- **`tag`** (*string*) – use tag slug.
- **`tag_id`** (*int*) – use tag id.
- **`tag__and`** (*array*) – use tag ids.
- **`tag__in`** (*array*) – use tag ids.

```
$query = new WP_Query( array( 'tag' => 'cooking' ) )
$query = new WP_Query( array( 'tag_id' => 13 ) )
$query = new WP_Query( array( 'tag' => 'bread+baking+recipe' ) )
$query = new WP_Query( array( 'tag__in' => array( 37, 47 ) ) );
```

3. Taxonomy_Parameters



```
'posts_per_page' => - 1,
'tax_query' => [
    'relation' => 'AND',
    array(
        'taxonomy' => 'category',
        'field' => 'slug', //'term_id'
        'terms' => array('sports'),
        'operator' => 'IN'
    ),
    array(
        'taxonomy' => 'post_tag',
        'field' => 'term_id', //'term_id'
        'terms' => array(6,5),
        'operator' => 'IN',
    ),

]
```

Sample 1.

```php
$args = array(
        'post_type' => 'post',
        'tax_query' => array(
            'relation' => 'AND',
                array(
                'taxonomy' => 'movie_genre',
                'field' => 'slug',
                'terms' => array( 'action', 'comedy' ),
                ),
                array(
                'taxonomy' => 'actor',
                'field' => 'term_id',
                'terms' => array( 103, 115, 206 ),
                'operator' => 'NOT IN',
                ),
        ),
    );

$query = new WP_Query( $args );
```

Sample 2

```
$args = array(
        'post_type' => 'post',
            'tax_query' => array(
            'relation' => 'OR',
                        array(
                            'taxonomy' => 'category',
                            'field' => 'slug',
                            'terms' => array( 'quotes' ),
                        ),
                        array(
                            'taxonomy' => 'post_format',
                            'field' => 'slug',
                            'terms' => array( 'post-format-quote' ),
                        ),
        ),
    );
$query = new WP_Query( $args );
```

4.Post Parameters

```
$query = new WP_Query( array( 'post_type' => 'page' ) );
$query = new WP_Query( array( 'post_type' => 'any' ) );

$args = array(
    'post_type' => array( 'post', 'page', 'movie', 'book' )
    );
$query = new WP_Query( $args );
```

5.Post Status Parameter

- ‘**publish**‘ – a published post or page.
- ‘**pending**‘ – post is pending review.
- ‘**draft**‘ – a post in draft status.
- ‘**auto-draft**‘ – a newly created post, with no content.
- ‘**future**‘ – a post to publish in the future.
- ‘**private**‘ – not visible to users who are not logged in.
- ‘**inherit**‘ – a revision. see get_children() .
- ‘**trash**‘ – post is in trashbin (available since version 2.9).
- ‘**any**‘ – retrieves any status except for ‘inherit’, ‘trash’ and ‘auto-draft’. Custom post statuses with ‘exclude_from_search’ set to true are also excluded.

5.Comment Parameter

```
$args = array(
    'post_type' => 'post',
    'comment_count' => 20,
    );
```

```
$args = array(
        'post_type' => 'post',
        'comment_count' => array(
        'value' => 25,
        'compare' => '>=',
        )
        );
```

Pagination Parameter

```
$query = new WP_Query( array( 'posts_per_page' => 3 ) );
$query = new WP_Query( array( 'posts_per_page' => -1 ) );
$query = new WP_Query( array( 'nopaging' => true ) );
```

Order & Orderby Parameters

```
order (string | array)
orderby (string | array)
‘none‘ – No order (available since version 2.8).
‘ID‘ – Order by post id. Note the capitalization.
‘author‘ – Order by author.
‘title‘ – Order by title.
‘name‘ – Order by post name (post slug).
‘type‘ – Order by post type (available since version 4.0).
‘date‘ – Order by date.
‘modified‘ – Order by last modified date.
‘parent‘ – Order by post/page parent id.
‘rand‘ – Random order.
‘comment_count‘ – Order by number of comments
```

```
$args = array(
    'orderby' => 'title',
    'order' => 'DESC',
    );
$query = new WP_Query( $args );
```

```
    $args = array(
        'orderby' => 'menu_order title',
        'order' => 'DESC',
        );
$query = new WP_Query( $args );


$args = array(
    'orderby' => array( 'meta_value_num' => 'DESC', 'title' => 'ASC' ),
    'meta_key' => 'age'
    );
$query = new WP_Query( $args );
```

MetaQuery

```
$args = array(
    'post_type' => 'your_post_type', // Replace with the name of your custom post
type
    'posts_per_page' => -1, // Retrieve all posts (you can adjust this as needed)
    'meta_key' => 'price', // Name of the custom field
    'meta_value' => 100, // Replace with your desired value
    'meta_compare' => '>', // Comparison operator (e.g., '>', '<', '>=', '<=')
    'orderby' => 'date', // Order by date (you can change this to other fields)
    'order' => 'DESC', // Descending order (you can use 'ASC' for ascending)
);

$custom_query = new WP_Query($args);

if ($custom_query->have_posts()) :
    while ($custom_query->have_posts()) : $custom_query->the_post();
        the_title();
        the_content();
        // Output the custom field value
        $price = get_post_meta(get_the_ID(), 'price', true);
        echo 'Price: $' . esc_html($price);
    endwhile;
    wp_reset_postdata();
else :
       echo 'No posts found.';
endif;
```

Custom Field (post meta) Parameters

- **meta_key** (*string*) – Custom field key.
- **meta_value** (*string*) – Custom field value.
- **meta_value_num** (*number*) – Custom field value.
- **meta_compare** (*string*) – Operator to test the '**meta_value**'. Possible values are '=', '!=', '>', '>=', '<', '<=', 'LIKE', 'NOT LIKE', 'IN', 'NOT IN', 'BETWEEN', 'NOT BETWEEN', 'NOT EXISTS', 'REGEXP', 'NOT REGEXP' or 'RLIKE'. Default value is '='.
- **meta_query** (*array*) – Custom field parameters

```
$args = array(
    'post_type' => 'my_custom_post_type',
    'meta_key' => 'age',
    'orderby' => 'meta_value_num',
    'order' => 'ASC',
    'meta_query' => array(
                        array(
                        'key' => 'age',
                        'value' => array( 3, 4 ),
                        'compare' => 'IN',
                            ),
                        ),
    );
    $query = new WP_Query( $args );


$q = new WP_Query(
        array(
            'meta_query' => array(
                'relation' => 'AND',
                'state_clause' => array(
                'key' => 'state',
                'value' => 'Wisconsin',
                ),
                'city_clause' => array(
                'key' => 'city',
                'compare' => 'EXISTS',
                    ),
                ),
            'orderby' => array(
            'city_clause' => 'ASC',
            'state_clause' => 'DESC',
            ),
        ) );
```

```php
$query = new WP_Query( array( 'meta_key' => 'color' ) );
$query = new WP_Query( array( 'meta_value' => 'blue' ) );


$args = array(
    'meta_value' => 'blue',
    'post_type' => 'page'
    );
    $query = new WP_Query( $args );


$args = array(
        'meta_key' => 'color',
        'meta_value' => 'blue',
        'meta_compare' => '!='
        );
        $query = new WP_Query( $args );
$args = array(
            'post_type' => 'event',
            'meta_key' => 'event_date',
            'meta_value' => date( "Ymd" ), // change to how "event date" is
stored
            'meta_compare' => '>',
            );
$query = new WP_Query( $args );

$args = array(
                'meta_key' => 'price',
                'meta_value' => '22',
                'meta_compare' => '<=',
                'post_type' => 'product'
                );
$query = new WP_Query( $args );

$args = array(
                    'post_type' => 'product',
                    'meta_query' => array(
                                        array(
                                        'key' => 'color',
                                        'value' => 'blue',
                                        'compare' => 'NOT LIKE',
                                            ),
                        ),
                    );
$query = new WP_Query( $args );
```

Permission Parameters

```
$args = array(
    'post_status' => array( 'publish', 'private' ),
    'perm' => 'readable',
    );
$query = new WP_Query( $args );
```

Mime Type Parameters

```
$args = array(
    'post_type' => 'attachment',
    'post_status' => 'inherit',
    'post_mime_type' => 'image/gif',
    );
$query = new WP_Query( $args );
```

```
$unsupported_mimes = array( 'image/jpeg', 'image/gif', 'image/png', 'image/bmp',
'image/tiff', 'image/x-icon' );
$all_mimes = get_allowed_mime_types();
$accepted_mimes = array_diff( $all_mimes, $unsupported_mimes );
$args = array(
'post_type' => 'attachment',
'post_status' => 'inherit',
'post_mime_type' => $accepted_mimes,
);
$query = new WP_Query( $query_args );
```

Caching Parameters

- **`cache_results`** (*boolean*) – Post information cache.
- **`update_post_meta_cache`** (*boolean*) – Post meta information cache.
- **`update_post_term_cache`** (*boolean*) – Post term information cache.

```php
$args = array(
    'posts_per_page' => 50,
    'cache_results' => false
    );
$query = new WP_Query( $args );

$args = array(
        'posts_per_page' => 50,
        'update_post_meta_cache' => false
        );
$query = new WP_Query( $args );
```

Return Fields Parameter

- **`fields`** (*string*) – Which fields to return. There are three options:
  - **`'all'`** – Return all fields (default).
  - **`'ids'`** – Return an array of post IDs.
  - **`'id=>parent'`** – Return an array of stdClass objects with ID and post_parent properties.

Passing anything else will return all fields (default) – an array of post objects.

Methods

- __call — Makes private/protected methods readable for backward compatibility.
- __construct — Constructor.
- __get — Makes private properties readable for backward compatibility.
- __isset — Makes private properties checkable for backward compatibility.
- fill_query_vars — Fills in the query variables, which do not exist within the parameter.
- generate_cache_key — Generates cache key.
- generate_postdata — Generates post data.
- get — Retrieves the value of a query variable.
- get_posts — Retrieves an array of posts based on query variables.
- get_queried_object — Retrieves the currently queried object.
- get_queried_object_id — Retrieves the ID of the currently queried object.
- get_search_stopwords — Retrieves stopwords used when parsing search terms.
- have_comments — Determines whether there are more comments available.
- have_posts — Determines whether there are more posts available in the loop.
- init — Initiates object properties and sets default values.
- init_query_flags — Resets query flags to false.
- is_404 — Determines whether the query is a 404 (returns no results).
- is_archive — Determines whether the query is for an existing archive page.
- is_attachment — Determines whether the query is for an existing attachment page.
- is_author — Determines whether the query is for an existing author archive page.
- is_category — Determines whether the query is for an existing category archive page.
- is_comment_feed — Determines whether the query is for a comments feed.

- is_comments_popup — Determines whether the current URL is within the comments popup window. — *deprecated*
- is_date — Determines whether the query is for an existing date archive.
- is_day — Determines whether the query is for an existing day archive.
- is_embed — Determines whether the query is for an embedded post.
- is_favicon — Determines whether the query is for the favicon.ico file.
- is_feed — Determines whether the query is for a feed.
- is_front_page — Determines whether the query is for the front page of the site.
- is_home — Determines whether the query is for the blog homepage.
- is_main_query — Determines whether the query is the main query.
- is_month — Determines whether the query is for an existing month archive.
- is_page — Determines whether the query is for an existing single page.
- is_paged — Determines whether the query is for a paged result and not for the first page.
- is_post_type_archive — Determines whether the query is for an existing post type archive page.
- is_preview — Determines whether the query is for a post or page preview.
- is_privacy_policy — Determines whether the query is for the Privacy Policy page.
- is_robots — Determines whether the query is for the robots.txt file.
- is_search — Determines whether the query is for a search.
- is_single — Determines whether the query is for an existing single post.
- is_singular — Determines whether the query is for an existing single post of any post type (post, attachment, page, custom post types).
- is_tag — Determines whether the query is for an existing tag archive page.
- is_tax — Determines whether the query is for an existing custom taxonomy archive page.
- is_time — Determines whether the query is for a specific time.
- is_trackback — Determines whether the query is for a trackback endpoint call.
- is_year — Determines whether the query is for an existing year archive.
- lazyload_comment_meta — Lazyloads comment meta for comments in the loop. — *deprecated*
- lazyload_term_meta — Lazyloads term meta for posts in the loop. — *deprecated*
- next_comment — Iterates current comment index and returns WP_Comment object.
- next_post — Sets up the next post and iterate current post index.
- parse_order — Parse an 'order' query variable and cast it to ASC or DESC as necessary.
- parse_orderby — Converts the given orderby alias (if allowed) to a properly-prefixed value.
- parse_query — Parses a query string and sets query type booleans.
- parse_query_vars — Reparses the query vars.
- parse_search — Generates SQL for the WHERE clause based on passed search terms.
- parse_search_order — Generates SQL for the ORDER BY condition based on passed search terms.
- parse_search_terms — Checks if the terms are suitable for searching.
- parse_tax_query — Parses various taxonomy related query vars.
- query — Sets up the WordPress query by parsing query string.
- reset_postdata — After looping through a nested query, this function restores the $post global to the current post in this query.
- rewind_comments — Rewinds the comments, resets the comment index and comment to first.
- rewind_posts — Rewinds the posts and resets post index.
- set — Sets the value of a query variable.
- set_404 — Sets the 404 property and saves whether query is feed.
- set_found_posts — Sets up the amount of found posts and the number of pages (if limit clause was used) for the current query.
- setup_postdata — Sets up global post data.
- the_comment — Sets up the current comment.
- the_post — Sets up the current post.

Creating Custom field

```php
function my_meta_fields(){
    ?>
    <label for="my-meta-field1">My Meta Field 1</label>
    <input type="text" name="my-meta-field1" id="my-meta-field1" value="<?php
    echo get_post_meta($post_id, 'my-meta-data', true)?>"/>
    <?php
}
function add_my_meta_box(){
    add_meta_box('my-meta-box', 'My Meta Box', 'my_meta_fields', 'cars');
}
add_action('add_meta_boxes', 'add_my_meta_box');
```

```php
function save_my_meta_data($post_id){
    $field_data = $_POST['my-meta-field1'];
    if(isset($_POST['my-meta-field1'])){
        if(get_post_meta($post_id, 'my-meta-data', true) != ''){
            update_post_meta($post_id, 'my-meta-data', $field_data);
        }else{
            add_post_meta($post_id, 'my-meta-data', $field_data);
        }
    }
}
add_action('save_post', 'save_my_meta_data');
```

Opps Method custom field

Refference
https://developer.wordpress.org/plugins/metadata/custom-meta-boxes/

```php
abstract class WPOrg_Meta_Box {

    /**
     * Set up and add the meta box.
     */
    public static function add() {
        $screens = [ 'post', 'wporg_cpt' ];
        foreach ( $screens as $screen ) {
            add_meta_box(
                'wporg_box_id',          // Unique ID
                'Custom Meta Box Title', // Box title
                [ self::class, 'html' ],   // Content callback, must be of type callable
                $screen                  // Post type
            );
        }
    }

    /**
     * Save the meta box selections.
     *
     * @param int $post_id  The post ID.
     */
    public static function save( int $post_id ) {
        if ( array_key_exists( 'wporg_field', $_POST ) ) {
            update_post_meta(
                $post_id,
                '_wporg_meta_key',
                $_POST['wporg_field']
            );
        }
    }

    /**
     * Display the meta box HTML to the user.
     *
```

```
     * @param WP_Post $post   Post object.
     */
    public static function html( $post ) {
        $value = get_post_meta( $post->ID, '_wporg_meta_key', true );
        ?>
        <label for="wporg_field">Description for this field</label>
        <select name="wporg_field" id="wporg_field" class="postbox">
            <option value="">Select something...</option>
            <option value="something" <?php selected( $value, 'something' );
?>>Something</option>
            <option value="else" <?php selected( $value, 'else' );
?>>Else</option>
        </select>
        <?php
    }
}

add_action( 'add_meta_boxes', [ 'WPOrg_Meta_Box', 'add' ] );
add_action( 'save_post', [ 'WPOrg_Meta_Box', 'save' ] );
```

Method 2

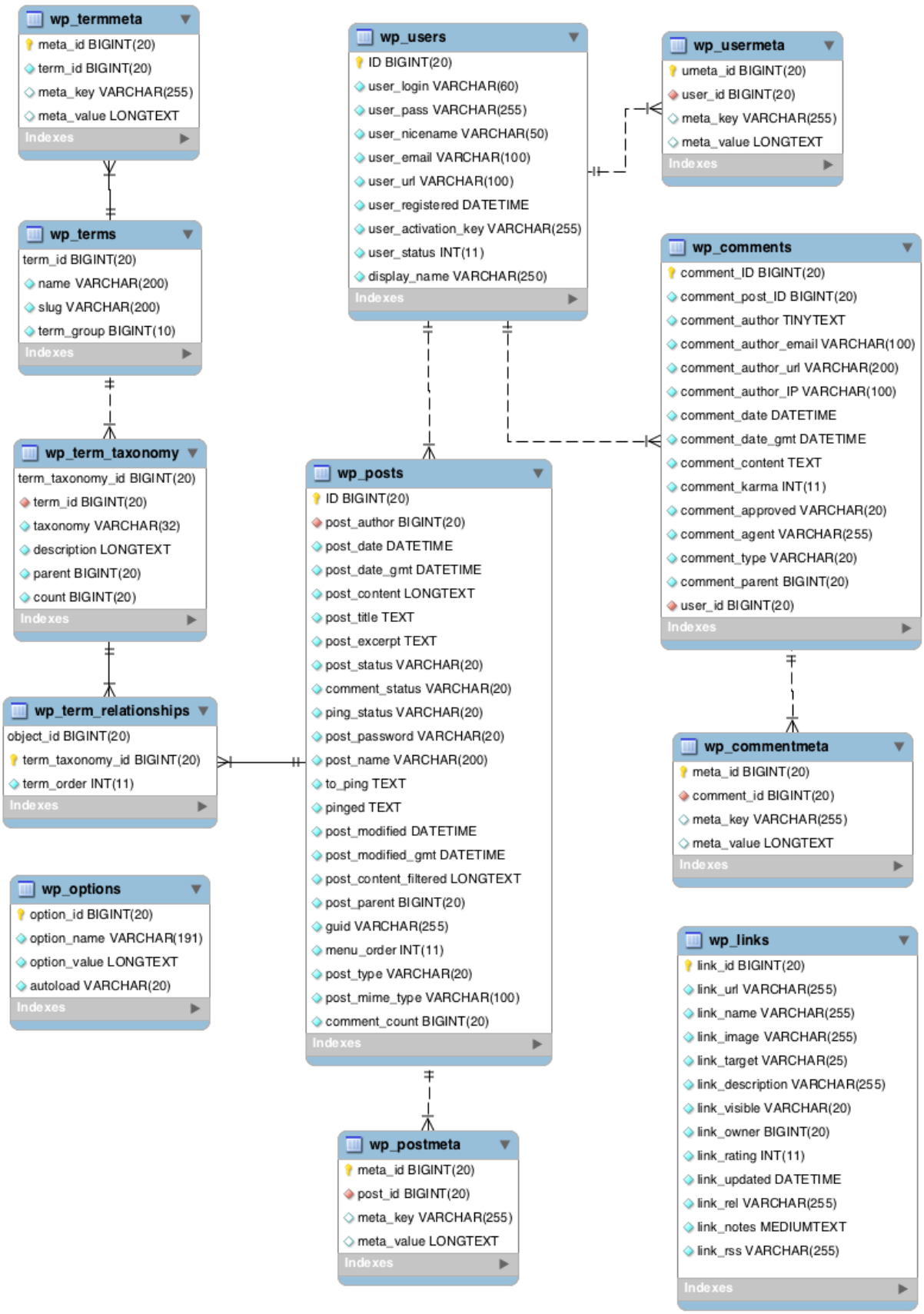
```
//adding metabox
function wporg_add_custom_box() {
    $screens = [ 'post', 'wporg_cpt' ];
    foreach ( $screens as $screen ) {
        add_meta_box(
            'wporg_box_id',                 // Unique ID
            'Custom Meta Box Title',      // Box title
            'wporg_custom_box_html',  // Content callback, must be of type
callable
            $screen                            // Post type
        );
    }
}
add_action( 'add_meta_boxes', 'wporg_add_custom_box' );
```

```php
function wporg_custom_box_html( $post ) {
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something">Something</option>
        <option value="else">Else</option>
    </select>
    <?php
}

//Adding metadata
function wporg_custom_box_html( $post ) {
    $value = get_post_meta( $post->ID, '_wporg_meta_key', true );
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something" <?php selected( $value, 'something' );
?>>Something</option>
        <option value="else" <?php selected( $value, 'else' );
?>>Else</option>
    </select>
    <?php
}


//Saving metadata
function wporg_save_postdata( $post_id ) {
    if ( array_key_exists( 'wporg_field', $_POST ) ) {
        update_post_meta(
            $post_id,
            '_wporg_meta_key',
            $_POST['wporg_field']
        );
    }
}
add_action( 'save_post', 'wporg_save_postdata' );
```

ontent > plugins > my-plugin > public > register.php

```php
<?php
    if(isset($_POST['register'])){
        global $wpdb;
        $fname = $wpdb->escape($_POST['user_fname']);
        $lname = $wpdb->escape($_POST['user_lname']);
        $username = $wpdb->escape($_POST['username']);
        $email = $wpdb->escape($_POST['user_email']);
        $pass = $wpdb->escape($_POST['user_pass']);
        $con_pass = $wpdb->escape($_POST['user_con_pass']);

        if($pass == $con_pass){
            // wp_insert_user()
            // wp_create_user()

            $result = wp_create_user($username, $pass, $email);

            if(!is_wp_error($result)){
                echo 'User Created ID:'. $result;
            }else{
                echo $result->get_error_message();
            }


        }else{
            echo 'Password must match!';
        }
```

```
        $user_data = array(
            'user_login' => $username,
            'user_email' => $email,
            'first_name' => $fname,
            'last_name' => $lname,
            'display_name' => $fname.' '.$lname,
            'user_pass' => $pass,
        );

        $result = wp_insert_user($user_data);

        if(!is_wp_error($result)){
            echo 'User Created ID:'. $result;
            add_user_meta($result, 'type', 'Faculty');
        }else{
            echo $result->get_error_message();
        }


    }else{
        echo 'Password must match!';
    }
}
```


wp_termmeta
meta_id BIGINT(20)
term_id BIGINT(20)
meta_key VARCHAR(255)
meta_value LONGTEXT
Indexes
wp_users
ID BIGINT(20)
user_login VARCHAR(60)
user_pass VARCHAR(255)
user_nicename VARCHAR(50)
user_email VARCHAR(100)
user_url VARCHAR(100)
user_registered DATETIME
user_activation_key VARCHAR(255)
user_status INT(11)
display_name VARCHAR(250)
Indexes
wp_usermeta
umeta_id BIGINT(20)
user_id BIGINT(20)
meta_key VARCHAR(255)
meta_value LONGTEXT
Indexes
wp_terms
term_id BIGINT(20)
name VARCHAR(200)
slug VARCHAR(200)
term_group BIGINT(10)
Indexes
wp_comments
comment_ID BIGINT(20)
comment_post_ID BIGINT(20)
comment_author TINYTEXT
comment_author_email VARCHAR(100)
comment_author_url VARCHAR(200)
comment_author_IP VARCHAR(100)
comment_date DATETIME
comment_date_gmt DATETIME
comment_content TEXT
comment_karma INT(11)
comment_approved VARCHAR(20)
comment_agent VARCHAR(255)
comment_type VARCHAR(20)
comment_parent BIGINT(20)
user_id BIGINT(20)
Indexes
wp_term_taxonomy
term_taxonomy_id BIGINT(20)
term_id BIGINT(20)
taxonomy VARCHAR(32)
description LONGTEXT
parent BIGINT(20)
count BIGINT(20)
Indexes
wp_posts
ID BIGINT(20)
post_author BIGINT(20)
post_date DATETIME
post_date_gmt DATETIME
post_content LONGTEXT
post_title TEXT
post_excerpt TEXT
post_status VARCHAR(20)
comment_status VARCHAR(20)
ping_status VARCHAR(20)
post_password VARCHAR(20)
post_name VARCHAR(200)
to_ping TEXT
pinged TEXT
post_modified DATETIME
post_modified_gmt DATETIME
post_content_filtered LONGTEXT
post_parent BIGINT(20)
guid VARCHAR(255)
menu_order INT(11)
post_type VARCHAR(20)
post_mime_type VARCHAR(100)
comment_count BIGINT(20)
Indexes
wp_commentmeta
meta_id BIGINT(20)
comment_id BIGINT(20)
meta_key VARCHAR(255)
meta_value LONGTEXT
Indexes
wp_term_relationships
object_id BIGINT(20)
term_taxonomy_id BIGINT(20)
term_order INT(11)
Indexes
wp_options
option_id BIGINT(20)
option_name VARCHAR(191)
option_value LONGTEXT
autoload VARCHAR(20)
Indexes
wp_links
link_id BIGINT(20)
link_url VARCHAR(255)
link_name VARCHAR(255)
link_image VARCHAR(255)
link_target VARCHAR(25)
link_description VARCHAR(255)
link_visible VARCHAR(20)
link_owner BIGINT(20)
link_rating INT(11)
link_updated DATETIME
link_rel VARCHAR(255)
link_notes MEDIUMTEXT
link_rss VARCHAR(255)
Indexes
wp_postmeta
meta_id BIGINT(20)
post_id BIGINT(20)
meta_key VARCHAR(255)
meta_value LONGTEXT
Indexes

How to create a custom plugin for your shortcode:

```php
<?php
/*
Plugin Name: Custom Shortcode Plugin
Description: A custom shortcode for demonstration purposes.
Version: 1.0
*/

// Define the shortcode function
function custom_shortcode_function($atts) {
    // Parse and sanitize shortcode attributes
    $atts = shortcode_atts(
        array(
            'parameter1' => 'default_value1',
            'parameter2' => 'default_value2',
        ),
        $atts
    );

    // Extract parameters
    $parameter1 = $atts['parameter1'];
    $parameter2 = $atts['parameter2'];

    // Your shortcode logic here
    $output = "Parameter 1: $parameter1<br>";
    $output .= "Parameter 2: $parameter2";

    return $output;
}

// Register the shortcode
add_shortcode('custom_shortcode', 'custom_shortcode_function');
```

[custom_shortcode parameter1="value1" parameter2="value2"]

This is some content before the custom shortcode.

[custom_shortcode parameter1="Hello" parameter2="World"]

This is some content after the custom shortcode.</p

Working with Users

Adding Users

To add a user you can use `wp_create_user()` or `wp_insert_user()`.

`wp_create_user()` creates a user using only the username, password and email parameters while `wp_insert_user()` accepts an array or object describing the user and its properties.

The functions available for manipulating User Metadata are: add_user_meta(), update_user_meta(), delete_user_meta() and get_user_meta().

Getting all the information of current user in wordpress using the predefined function `wp_get_current_user();`

```php
<?php

        $current_user = wp_get_current_user();

        echo "Username :".$current_user->user_login;
        echo "Username :".$current_user->ID;
        echo "Username :".$current_user->user_pass;
        echo "Username :".$current_user->user_nicename;
        echo "Username :".$current_user->user_email;
        echo "Username :".$current_user->user_url;
        echo "Username :".$current_user->user_registered;
        echo "Username :".$current_user->user_activation_key;
        echo "Username :".$current_user->user_status;
        echo "Username :".$current_user->display_name;
```

```php
// check if the username is taken
$user_id = username_exists( $user_name );

// check that the email address does not belong to a registered user
if ( ! $user_id && email_exists( $user_email ) === false ) {
    // create a random password
    $random_password = wp_generate_password( 12, false );
    // create the user
    $user_id = wp_create_user(
        $user_name,
        $random_password,
        $user_email
    );
}
```

Adding Roles

Add new roles and assign capabilities to them with add_role().

```php
function wporg_simple_role() {
    add_role(
        'simple_role',
        'Simple Role',
        array(
            'read'         => true,
            'edit_posts'   => true,
            'upload_files' => true,
        ),
    );
}

// Add the simple_role.
add_action( 'init', 'wporg_simple_role' );
```

Removing Roles

Remove roles with remove_role().

```php
function wporg_simple_role_remove() {
    remove_role( 'simple_role' );
}

// Remove the simple_role.
add_action( 'init', 'wporg_simple_role_remove' );
```

Adding Capabilities

You may define new capabilities for a role.

Use get_role() to get the role object, then use the add_cap() method of that object to add a new capability.

```php
function wporg_simple_role_caps() {
    // Gets the simple_role role object.
    $role = get_role( 'simple_role' );

    // Add a new capability.
    $role->add_cap( 'edit_others_posts', true );
}

// Add simple_role capabilities, priority must be after the initial role definit:
add_action( 'init', 'wporg_simple_role_caps', 11 );
```

```
STEPS TO VALIDATE FORM FIELDS
=============================================
1. Check no space within username:
    strpos($username, ' ')

2. Check username must have value:
    empty($username)

3. Check username existence:
    username_exists( $username )

4. Check email validation
    is_email( $email )

5. Check email existence
     email_exists( $email )

6. Password matched or not
```

index.php | custom-registration.php

Source | History

```
3   get_header();
4   global $wpdb;
5
6   if ($_POST) {
7
8       $username = $wpdb->escape($_POST['txtUsername']);
        $email = $wpdb->escape($_POST['txtEmail']);
        $password = $wpdb->escape($_POST['txtPassword']);
        $ConfPassword = $wpdb->escape($_POST['txtConfirmPassword'])
12
        if(strpos($username, ' ')!==FALSE){
14
15      }
16
17  }
```

index.php | custom-registration.php

Source | History

```
13          $error = array();
14          if (strpos($username, ' ') !== FALSE) {
15              $error['username_space'] = "Username has Space";
16          }
17
18          if (empty($username)) {
19              $error['username_empty'] = "Needed Username must";
20          }
21
22          if (username_exists($username)) {
23              $error['username_exists'] = "Username already exists";
24          }
25
26          if (!is_email($email)) {
27              $error['email_valid'] = "Email has no valid value";
28          }
29
30          if (email_exists($email)) {
31              $error['email_existence'] = "Email already exists";
32          }
33      }
34      ?>
```

index.php | custom-registration.php

Source | History

```
31              $error['email_existence'] = "Email already exists";
32          }
33
34          if (strcmp($password, $ConfPassword) !== 0) {
35              $error['password'] = "Password didn't match";
36          }
37
38          if (count($error) == 0) {
39
40              wp_create_user($username, $password, $email);
41              echo "User Created Successfully";
42              exit();
43          }else{
44
45              print_r($error);
46
47          }
48      }
49      ?>
```

custom-login.php

Source History

```
2
3  /* Template Name: Custom Login Page */
4
5  global $user_ID;
6
7  if (!$user_ID) {
8      // user in logged out state
9      ?>
10 <form>
11     <p>
12         <label for="username">Username/Email</label>
13         <input type="text" id="username" name="username" placeholder="Enter Username/Email"/>
14     </p>
15     <p>
16         <label for="password">Password</label>
17         <input type="password" id="password" name="password" placeholder="Enter Password"/>
18     </p>
19 </form>
20     <?php
21
22 } else {
```

custom-login.php

Source History

```
5
6  if (!$user_ID) {
7      // user in logged out state
8      get_header();
9      ?>
10     <form method="post">
11         <p>
12             <label for="username">Username/Email</label>
13             <input type="text" id="username" name="username" placeholder="Enter Username/Emai
14         </p>
15         <p>
16             <label for="password">Password</label>
17             <input type="password" id="password" name="password" placeholder="Enter Password"
18         </p>
19         <p>
20             <button type="submit" name="btn_submit">Log In</button>
21         </p>
22     </form>
23     <?php
24     get_footer();
25 } else {
```

custom-login.php

Source History

```php
<?php
/* Template Name: Custom Login Page */

global $user_ID;
global $wpdb;

if (!$user_ID) {

    if ($_POST) {

        $username = $wpdb->escape($_POST['username']);
        $password = $wpdb->escape($_POST['password']);

    } else {
        // user in logged out state
        get_header();
        ?>
        <form method="post">
            <p>
                <label for="username">Username/Email</label>
                <input type="text" id="username" name="username" placeholder="En
```

custom-login.php

Source History

```php
    if ($_POST) {

        $username = $wpdb->escape($_POST['username']);
        $password = $wpdb->escape($_POST['password']);

        $login_array = array();
        $login_array['user_login'] = $username;
        $login_array['user_password'] = $password;

        $verify_user =  wp_signon($login_array, true);
        if(!is_wp_error($verify_user)){

        }else{


        }

    } else {
        // user in logged out state
        get_header();
        ?>
```

custom-login.php | functions.php

Source | History

```php
if (!$user_ID) {

    if ($_POST) {

        $username = $wpdb->escape($_POST['username']);
        $password = $wpdb->escape($_POST['password']);

        $login_array = array();
        $login_array['user_login'] = $username;
        $login_array['user_password'] = $password;

        $verify_user =  wp_signon($login_array, true);
        if(!is_wp_error($verify_user)){

            echo "<script>window.location = '".site_url()."'</script>";

        }else{

            echo "<p>Invalid Credentials</p>";
        }
```

custom-login.php | functions.php

Source | History

```php
// ends here

function redirect_to_custom_login_page() {
    wp_redirect(site_url() . "/login");
    exit();
}

add_action("wp_logout", "redirect_to_custom_login_page");


add_action("init", "fn_redirect_wp_admin");

function fn_redirect_wp_admin() {
    global $pagenow;
    if ($pagenow == "wp-login.php") {
        wp_redirect(site_url() . "/login");
        exit();
    }
}
```

```php
add_shortcode("owt", "wpl_owt_shortcode_part1");

function wpl_owt_shortcode_part1() {

    return "This is a simple plugin shortcode that we have created";
}

add_shortcode("owt-file", "wpl_owt_shortcode_part2");

function wpl_owt_shortcode_part2() {
    include_once OWT_SHORTCODE_PLUGIN_DIR_PATH . '/views/owt-shortcode-panel.php'
}

add_shortcode('owt-play')
```

```php
function wpl_owt_shortcode_part2() {
    include_once OWT_SHORTCODE_PLUGIN_DIR_PATH . '/views/owt-shortcode-panel.php';
}

add_shortcode('owt-playlist', "wpl_owt_shortcode_part3");

function wpl_owt_shortcode_part3($attributes) {
    //print_r($attributes);

    $attributes = shortcode_atts(array(
        "playlist_name" => "No Playlist defined",
        "total_videos" => "No videos found",
        "author" => "No author created",
        "channel" => "No channel found"
    ),$attributes,"");

    echo "<br/><br/>Playlist name: " . $attributes['playlist_name'];
    echo "<br/>Total videos: " . $attributes['total_videos'];
    echo "<br/>Author: " . $attributes['author'];
    echo "<br/>Channel: " . $attributes['channel'];
    //return "Hello I am a parameterized shortcode";
}
```

```php
add_action('admin_menu','da_custom_menu');

function da_custom_menu(){

    add_menu_page( page_title: 'Records', menu_title: 'Record List', capability: 'manage_option

    add_submenu_page( parent_slug: 'da-record', page_title: 'Record List Sub Menu', menu_title

}

function da_record_list_callback(){

    echo "<h1>Record List</h1>";
}

function da_submenu_callback_func(){

    echo "<h2>Sub Menu</h2>";
}
```

da_submenu_callback_func()

wpacademy-like-dislike.php

```php
add_action('admin_menu', 'wpac_register_menu_page');

//Sub-Level Administration Menu
/* function wpac_register_menu_page() {
    add_theme_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html', 30 );
}
add_action('admin_menu', 'wpac_register_menu_page'); */

function wpac_plugin_settings(){

    register_setting( option_group, option_name, sanitize_callback );

    add_settings_section( id, title, callback, page );

    add_settings_field( id, title, callback, page, section, args );
}
add_action('admin_init', 'wpac_plugin_settings');
?>
```

```php
//Include Scripts & Styles
if( !function_exists('wpac_plugin_scripts')) {
    function wpac_plugin_scripts() {
        wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
        wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true );
    }
    add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
}

add_menu_page( page_title, menu_title, capability, menu_slug, function, icon_url, position );
?>
```

pwspk-plugin-dev.php — wpdev

pwspk-plugin-dev.php | template.php | option.php | plugin.php

```php
    add_submenu_page('pwspk-options', 'PWSPK Layout', 'PWSPK Layout', 'manage_options', '
        pwspk-layout', 'pwspk_layout_func');

}
register_activation_hook(__FILE__, function(){
    add_option('pwspk_option_1', '');
});
register_deactivation_hook(__FILE__, function(){
    delete_option('pwspk_option_1');
});
function pwspk_process_form_settings(){
    register_setting('pwspk_option_group', 'pwspk_option_name' );
    if(isset($_POST['action']) && current_user_can('manage_options')){
        update_option('pwspk_option_1', sanitize_text_field($_POST['pwspk_option_1']));
    }
}
function pwspk_options_func(){ ?>
    <div class="wrap">
        <h1>PWSPK Options Menu</h1>
        <?php settings_errors(); ?>
        <form action="options.php" method="post">
            <?php settings_fields('pwspk_option_group'); ?>
            <label for="">Setting One: <input type="text" name="pwspk_option_1" value="<?php
                echo esc_html(get_option('pwspk_option_1')); ?>" /></label>
            <?php submit_button('Save Changes'); ?>
        </form>
    </div>
<?php
```

Predefined Sub-Menus

Wouldn't it be nice if we had helper functions that define the **$parent_slug** for WordPress built-in Top-level menus and save us from manually searching it through the source code?

Below is a list of parent slugs and their helper functions:

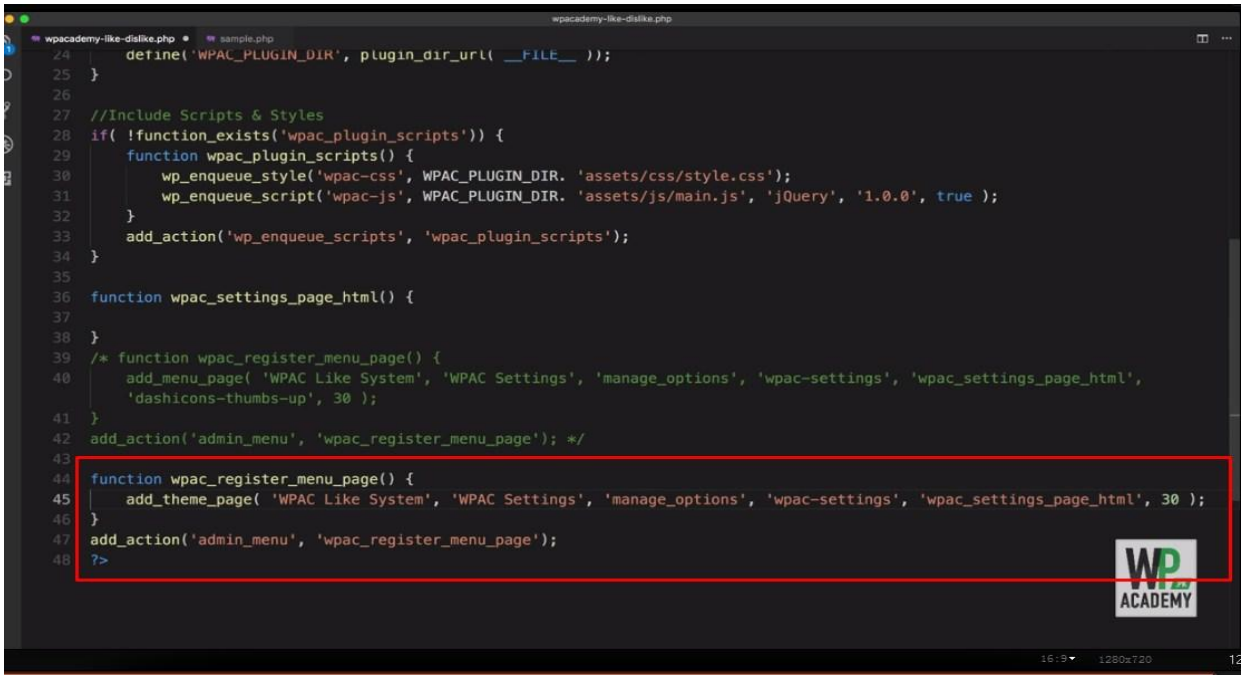
- add_dashboard_page() - `index.php`
- add_posts_page() - `edit.php`
- add_media_page() - `upload.php`
- add_pages_page() - `edit.php?post_type=page`
- add_comments_page() - `edit-comments.php`
- add_theme_page() - `themes.php`
- add_plugins_page() - `plugins.php`
- add_users_page() - `users.php`
- add_management_page() - `tools.php`
- add_options_page() - `options-general.php`
- add_options_page() - `settings.php`
- add_links_page() - `link-manager.php` - requires a plugin since WP 3.5+
- Custom Post Type - `edit.php?post_type=wporg_post_type`
- Network Admin - `settings.php`

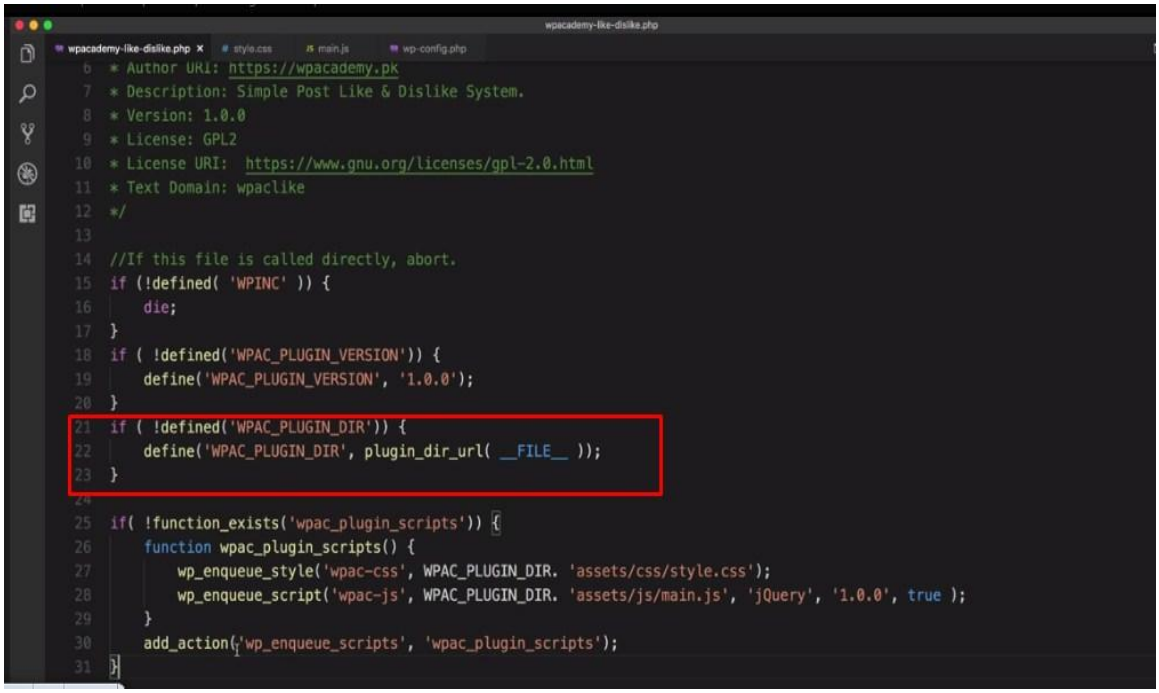
Top ↑

```php
//Settings Page HTML
function wpac_settings_page_html() {

        if(!is_admin()) {
            return;
        }
        ?>
            <div class="wrap">
                <form action="options.php" method="post">
                    <?php
                        settings_fields( 'wpac-settings' );
                        do_settings_sections( 'wpac-settings' );
                        submit_button( 'Save Changes' );
                    ?>
                </form>
            </div>
        <?

}
```

```php
function wpac_plugin_settings(){

    register_setting( 'wpac-settings', 'wpac_like_btn_label' );
    register_setting( 'wpac-settings', 'wpac_dislike_bth_label' );

    add_settings_section( 'wpac_label_settings_section', 'WPAC Button Labels', 'wpac_plugin_settings_section_cb',
    'wpac-settings' );

    add_settings_field( 'wpac_like_label_field', 'Like Button Label', 'wpac_like_label_field_cb', 'wpac-settings',
    'wpac_label_settings_section' );
}
add_action('admin_init', 'wpac_plugin_settings');

function wpac_plugin_settings_section_cb(){
    echo '<p>Define Button Labels</p>';
}

function wpac_like_label_field_cb(){

}
?>
```

```php
//Include Scripts & Styles
if( !function_exists('wpac_plugin_scripts')) {
    function wpac_plugin_scripts() {
        wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
        wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true );
    }
    add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
}

function wpac_settings_page_html() {

}
/* function wpac_register_menu_page() {
    add_menu_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html',
    'dashicons-thumbs-up', 30 );
}
add_action('admin_menu', 'wpac_register_menu_page'); */

function wpac_register_menu_page() {
    add_options_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html', 30 );
}
add_action('admin_menu', 'wpac_register_menu_page');
?>
```


wpacademy-like-dislike.php
wpacademy-like-dislike.php
sample.php
define('WPAC_PLUGIN_DIR', plugin_dir_url(__FILE__));
}
//Include Scripts & Styles
if(!function_exists('wpac_plugin_scripts')) {
function wpac_plugin_scripts() {
wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true);
}
add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
}
function wpac_settings_page_html() {
}
/* function wpac_register_menu_page() {
add_menu_page('WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html',
'dashicons-thumbs-up', 30);
}
add_action('admin_menu', 'wpac_register_menu_page'); */
function wpac_register_menu_page() {
add_theme_page('WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html', 30);
}
add_action('admin_menu', 'wpac_register_menu_page');
?>
WP
ACADEMY
16:9
1280x720


wpacademy-like-dislike.php
wpacademy-like-dislike.php
style.css
main.js
wp-config.php
* Author URI: https://wpacademy.pk
* Description: Simple Post Like & Dislike System.
* Version: 1.0.0
* License: GPL2
* License URI: https://www.gnu.org/licenses/gpl-2.0.html
* Text Domain: wpaclike
*/
//If this file is called directly, abort.
if (!defined('WPINC')) {
die;
}
if (!defined('WPAC_PLUGIN_VERSION')) {
define('WPAC_PLUGIN_VERSION', '1.0.0');
}
if (!defined('WPAC_PLUGIN_DIR')) {
define('WPAC_PLUGIN_DIR', plugin_dir_url(__FILE__));
}
if(!function_exists('wpac_plugin_scripts')) {
function wpac_plugin_scripts() {
wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true);
}
add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
}

wpacademy-like-dislike.php • main.js wp-config.php

```
*/

//If this file is called directly, abort.
if (!defined( 'WPINC' )) {
    die;
}
if ( !defined('WPAC_PLUGIN_VERSION')) {
    define('WPAC_PLUGIN_VERSION', '1.0.0');
}
if ( !defined('WPAC_PLUGIN_DIR')) {
    define('WPAC_PLUGIN_DIR', plugin_dir_path( __FILE__ ));
}

if( !function_exists('wpac_plugin_scripts')) {
    function wpac_plugin_scripts() {
        wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js' );
    }
}
```

Plugin directory setting path constant

```
if ( $query->have_posts() ) :
  while ( $query->have_posts() ) : $query->the_post();
  $product = new WC_Product ( get_the_ID() );
  ?>
    <div class="wpa-product pull-left col-lg-3 col-lmd-3 col-sm-6 col-xs-12">
      <div class="product-inner">
        <div class="product-thumb">
          <a href="<?php the_permalink(); ?>"><?php the_post_thumbnail('medium'); ?>
        </div>
        <div class="product-title">
          <h3><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h3>
        </div>
        <div class="product-price">
          <?php echo $product->get_price_html() ?>
        </div>
        <div class="product-buttons">

        </div>
      </div>
    </div>
  <?php
  endwhile;

else :
```

```php
//Settings Page HTML
function wpac_settings_page_html() {

        if(!is_admin()) {
            return;
        }
        ?>
            <div class="wrap">
                <h1><?= esc_html(get_admin_page_title()); ?></h1>
                <form action="options.php" method="post">
                    <?php
                        settings_fields( 'wpac-settings' );
                        do_settings_sections( 'wpac-settings' );
                        submit_button( 'Save Changes' );
                    ?>
                </form>
            </div>
        <?

}
```

```php
//Settings Page HTML
function wpac_settings_page_html() {

        if(!is_admin()) {
            return;
        }
        ?>
            <div class="wrap">
                <h1><?= esc_html(get_admin_page_title()); ?></h1>
                <form action="options.php" method="post">
                    <?php
                        settings_fields( 'wpac-settings' );
                        do_settings_sections( 'wpac-settings' );
                        submit_button( 'Save Changes' );
                    ?>
                </form>
            </div>
        <?

}
```

What Is The WordPress Plugin Boilerplate?
==========
The boilerplate is a standardized, organized, object-oriented foundation for building high-quality WordPress plugins. There are some simple examples of how the parts work together as examples for building your plugin. In addition, the boilerplate includes easily overlooked plugin features like inline documentation and translation files.

Why we use Boilerplate?
==========
Here are at least five different ways the boilerplate helps when building plugins.
1. Organized
2. Object-Oriented
3. WordPress Coding Standards
4. WordPress Documentation Standards
5. Translatable
6. Plugin Generator

```php
function wpacademy_publish_send_mail(){
    global $post;
    $author = $post->post_author; /* Post author ID. */
    $name = get_the_author_meta( 'display_name', $author );
    $email = get_the_author_meta( 'user_email', $author );
    $title = $post->post_title;
    $permalink = get_permalink( $ID );
    $edit = get_edit_post_link( $ID, '' );
    $to[] = sprintf( '%s <%s>', $name, $email );
    $subject = sprintf( 'Published: %s', $title );
    $message = sprintf ('Congratulations, %s! Your article "%s" has been published.' . "\n\n", $name, $title );
    $message .= sprintf( 'View: %s', $permalink );
    $headers[] = '';
    wp_mail( $to, $subject, $message, $headers );
}
add_action( 'publish_post', 'wpacademy_publish_send_mail' );
```

```php
function wpacademy_like_filter_example( $words ) {
    return 10;
}
add_filter('excerpt_length', 'wpacademy_like_filter_example');

function wpacademy_like_filter_example_2( $more ) {
    $more = '<a href="'.get_the_permalink().'">More</a>';
    return $more;
}
add_filter('excerpt_more', 'wpacademy_like_filter_example_2');
```

```php
function get_widgets(){
    register_sidebar(array(
        "name"=> __('Footer_copywrite','vishal'),
        "id"=>'footer_copywrite',
        'before_widget'=>'<p>',
        'after_widget'=>'</p>',
        'before_title'=>'<h2>',
        'after_title'=>'</h2>',
    ));
}

add_action('widgets_init','get_widgets');
```

```php
<?php if ( is_active_sidebar( index: 'custom-widget' ) ) : ?>
    <ul id="sidebar">
        <?php dynamic_sidebar( index: 'custom-widget' ); ?>
    </ul>
<?php endif; ?>
```

```php
<!-- Nav -->
    <nav id="menu">
        <?php
        wp_nav_menu(
            array(
                'container'=>false,
                'menu_class'=>'links',
                'theme_location'=>'top_menu'
            )
        );
        ?>
    </nav>
```

```php
add_theme_support('menus');

register_nav_menus(
    array('top_menu'=>'Top Menu','theme')
);
```

```php
* This is our functions file
*/
function wplearning_theme_scripts() {
    wp_enqueue_style('style', get_stylesheet_uri());
    wp_enqueue_script('my-slider', get_template_directory_uri().'/assets/js/my-js.js');
}
add_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );
```

```php
<title>Urban by TEMPLATED</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="stylesheet" href="<?php echo get_template_directory_uri()?>/assets/css/main.css" />
<?php wp_head();?>
```

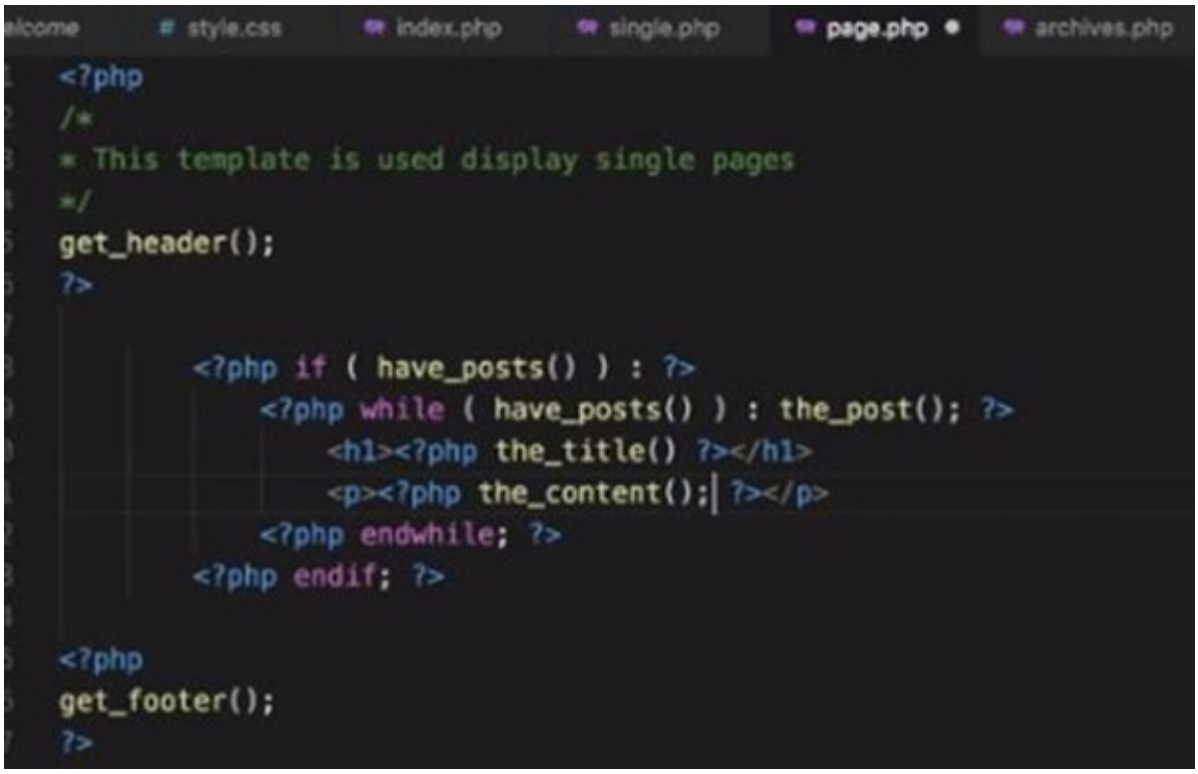
```php
<?php
get_header();

$hero=get_field('hero_banner');
echo '<pre>';
print_r($hero);
die();
?>
<section id="banner">
                <div class="inner">
                    <header>
                        <h1>This is Urban</h1>
                        <p>Aliquam libero augue varius non odio nec faucibus
                        rutrum tempus massa accumsan faucibus purus.</p>
                    </header>
                    <a href="#main" class="button big scrolly">Learn More</a>
                </div>
            </section>
```

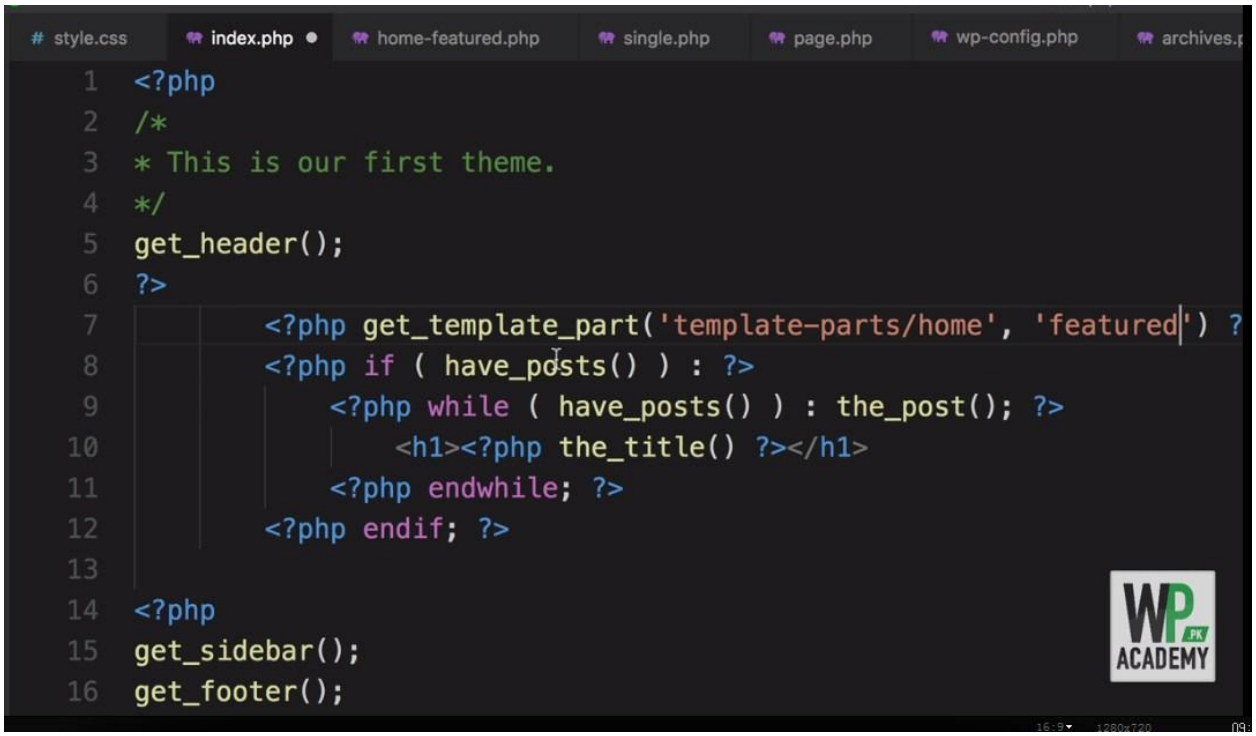
```php
<?php
get_header();

$hero=get_field('hero_banner');
?>
<section id="banner" style="background-image: url('<?php echo $hero['background_image']['url']?>')">
                <div class="inner">
                    <header>
                        <h1><?php echo $hero['heading']?></h1>
                        <p><?php echo $hero['subheading']?></p>
                    </header>
                    <a href="<?php echo $hero['btn_link']?>" class="button big scrolly"><?php echo $hero['btn_txt'
                    ]?></a>
                </div>
            </section>

        <!-- Main -->
            <div id="main">

                <!-- Section -->
                    <section class="wrapper style1">
                        <div class="inner">
```

```
<section class="wrapper style1">
    <div class="inner">
        <header class="align-center">
            <h2>Aliquam ipsum purus dolor</h2>
            <p>Cras sagittis turpis sit amet est tempus, sit amet consectetur purus tincidunt.
        </header>
        <div class="flex flex-3">

            <?php
            $arr=array('category'=>5,'post_type'=>'post');
            $posts=get_posts($arr);
            foreach($posts as $post):setup_postdata($post);
                ?>
                <div class="col align-center">
                    <div class="image round fit">
                        <?php the_post_thumbnail()?>
                    </div>
                    <p><?php the_content()?></p>
                </div>
                <?php
            endforeach;
            ?>

        </div>
```

```
 * This is our functions file
 */
 function wplearning_theme_scripts() {
     wp_enqueue_style('style', get_stylesheet_uri());
     wp_enqueue_script('my-slider', get_template_directory_uri().'/assets/js/my-js.js');
 }
 add_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );
```

```
* This is our functions file
*/
function wplearning_theme_scripts() {
    wp_enqueue_style('style', get_stylesheet_uri());
    wp_enqueue_style('bootstrap-css', get_template_directory_uri().'/assets/bootstrap/css/bootstrap.min.css');

    wp_enqueue_script('bootstrap-js', get_template_directory_uri().'/assets/bootstrap/js/bootstrap.min.js');
}
add_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );
```


style.css
index.php
single.php
page.php
archives.php
<?php
/*
* This template is used display single pages
*/
get_header();
?>
<?php if (have_posts()) : ?>
<?php while (have_posts()) : the_post(); ?>
<h1><?php the_title() ?></h1>
<?php the_content(); ?>
<?php endwhile; ?>
<?php endif; ?>
<?php
get_footer();
?>


style.css
index.php
home-featured.php
single.php
page.php
wp-config.php
archives.p
1 <?php
2 /*
3 * This is our first theme.
4 */
5 get_header();
6 ?>
7 <?php get_template_part('template-parts/home', 'featured') ?
8 <?php if (have_posts()) : ?>
9 <?php while (have_posts()) : the_post(); ?>
10 <h1><?php the_title() ?></h1>
11 <?php endwhile; ?>
12 <?php endif; ?>
13
14 <?php
15 get_sidebar();
16 get_footer();
WP PK ACADEMY

```php
<html>
<meta charset="UTF-8">
<head>
    <title>
        <?php echo get_the_title(); ?> |
        <?php bloginfo('name') ?>
    </title>
    <?php wp_head() ?>
</head>
<body>
    <div class="site-main container">
        <header class="site-header">
            <div class="site-branding">
                <a href="<?php bloginfo('url') ?>">
                    <?php the_custom_logo() ?>
                </a>
            </div>
        </header>
        <nav class="site-navigation">

        </nav>
```

```php
add_theme_support('custom-logo');

register_nav_menus( array(
    'primary'   => __( 'Primary Menu', 'wplearning' )
) );

function wplearning_theme_scripts() {

    wp_enqueue_style('style', get_stylesheet_uri());

    wp_enqueue_style('bootstrap-css', get_template_directory_uri().'/assets/bootstrap/css/bootstrap.min.css');

    wp_enqueue_script('jquery');

    wp_enqueue_script('bootstrap-js', get_template_directory_uri().'/assets/bootstrap/js/bootstrap.min.js');

}
add_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );
```

index.php | footer.php | header.php • | functions.php | page.php | single.php | wp-config.php | archives.php

```php
*/
?>
<!DOCTYPE html>
    <html>
    <meta charset="UTF-8">
    <head>
        <title>
            <?php echo get_the_title(); ?> |
            <?php bloginfo('name') ?>
        </title>
        <?php wp_head() ?>
    </head>
    <body>
        <div class="site-main container">
            <header class="site-header">
                <div class="site-branding">
                    <?php the_custom_logo() ?>
                </div>
            </header>
            <nav class="site-navigation">
                <?php
                wp_nav_menu( array(
                    'theme_location' => 'primary'
                ) );
                ?>
            </nav>
```

```php
34
35  function wplearning_widgets_init() {
36      register_sidebar( array(
37          'name'          => __( 'Primary Sidebar', 'theme_name' ),
38          'id'            => 'main-sidebar',
39          'description'   => 'Main Sidebar on Right Side',
40          'before_widget' => '<aside id="%1$s" class="widget %2$s">',
41          'after_widget'  => '</aside>',
42          'before_title'  => '<h3 class="widget-title">',
43          'after_title'   => '</h3>',
44      ) );
45  }
46  add_action( 'widgets_init', 'wplearning_widgets_init' );
47
48
49
```

index.php footer.php header.php functions.php page.php single.php wp-config.php archives.php 404.php search.php

```php
<?php
/*
* This is our functions file
*/
function wplearning_theme_setup() {

    add_theme_support('custom-logo');

    add_theme_support('title-tag');

    add_theme_support('post-thumbnails');

    register_nav_menus( array(
        'primary'   => __( 'Primary Menu', 'wplearning' )
    ) );
};
add_action('after_setup_theme', 'wplearning_theme_setup');


function wplearning_theme_scripts() {

    wp_enqueue_style('style', get_stylesheet_uri());

    wp_enqueue_style('bootstrap-css', get_template_directory_uri().'/assets/bootstrap/css/bootstrap.min.css');

    wp_enqueue_script('jquery');

    wp_enqueue_script('bootstrap-js', get_template_directory_uri().'/assets/bootstrap/js/bootstrap.min.js');


d_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );
```

index.php footer.php header.php functions.php page.php single.php wp-config.php archives.php 404.php

```php
<?php
/*
* This template is used display sidebar
*/
?>
<div class="main-sidebar-inner">
    <?php
        dynamic_sidebar('main-sidebar');
    ?>
</div>
```

index.php × | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archives.php | 404.php

```php
<?php
/*
* This is our first theme.
*/
get_header();
?>

        <div class="home-main">
            <div class="row mr-0 ml-0">
             <div class="home-posts col-lg-8 col-xs-12">
                <?php if ( have_posts() ) : ?>
                    <?php while ( have_posts() ) : the_post(); ?>
                        <h1><?php the_title() ?></h1>
                        <p><?php the_content(); ?></p>
                    <?php endwhile; ?>
                <?php endif; ?>
             </div>
             <div class="home-sidebar col-lg-4 col-xs-12">
                <?php get_sidebar(); ?>
             </div>
            </div>

        </div>


<?php
get_footer();
```

sidebar.php

style.css × | index.php | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archives.php | 404.php

```php
<?php
/*
* This template is used display sidebar
*/
?>
<div class="main-sidebar-inner">
    <?php
        dynamic_sidebar('main-sidebar');
    ?>
</div>
```

```
}
add_action('wp_enqueue_scripts', 'wplearning_theme_scripts' );

function wplearning_widgets_init() {
    register_sidebar( array(
        'name'          => __( 'Primary Sidebar', 'theme_name' ),
        'id'            => 'main-sidebar',
        'description'   => 'Main Sidebar on Right Side',
        'before_widget' => '<aside id="%1$s" class="widget %2$s">',
        'after_widget'  => '</aside>',
        'before_title'  => '<h3 class="widget-title">',
        'after_title'   => '</h3>',
    ) );

    register_sidebar( array(
        'name'          => __( 'Footer Widget 1', 'theme_name' ),
        'id'            => 'main-sidebar',
        'description'   => 'Main Sidebar on Right Side',
        'before_widget' => '<aside id="%1$s" class="widget %2$s">',
        'after_widget'  => '</aside>',
        'before_title'  => '<h3 class="widget-title">',
        'after_title'   => '</h3>',
    ) );
}
add_action( 'widgets_init', 'wplearning_widgets_init' );
```

single.php

```
<?php
/*
* This template is used display single posts
*/
get_header();
?>

        <?php if ( have_posts() ) : ?>
            <?php while ( have_posts() ) : the_post(); ?>
                <h1><?php the_title() ?></h1>
                <p><?php the_content(); ?></p>
            <?php endwhile; ?>
        <?php endif; ?>

<?php
get_footer();
?>
```

```php
function wplearning_theme_setup() {

    add_theme_support('custom-logo');

    add_theme_support('title-tag');

    add_theme_support('post-thumbnails');

    add_image_size ('home-featured', 640, 400, array('center', 'center'));

    add_theme_support('automatic-feed-links');

    register_nav_menus( array(
        'primary'   => __( 'Primary Menu', 'wplearning' )
    ) );
};
add_action('after_setup_theme', 'wplearning_theme_setup');


function wplearning_theme_scripts() {

    wp_enqueue_style('style', get_stylesheet_uri());

    wp_enqueue_style('bootstrap-css', get_template_directory_uri().'/assets/bootstrap/css/bootstrap.min.css');

    wp_enqueue_script('jquery');

    wp_enqueue_script('bootstrap-js', get_template_directory_uri().'/assets/bootstrap/js/bootstrap.min.js');
```

index.php | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archives.php | 404.php | search.php | sidebar

```php
<?php
/*
* This is our first theme.
*/
get_header();
?>

    <div class="home-main">
        <div class="row mr-0 ml-0">
         <div class="home-posts col-lg-8 col-xs-12">
            <?php if ( have_posts() ) : ?>
                <?php while ( have_posts() ) : the_post(); ?>
                    <article class="home-post">
                        <div class="post-header">
                            <div class="post-thumbnail row ml-0 mr-0">
                                <a href="<?php the_permalink() ?>"><?php the_post_thumbnail('home-featured') ?></
                            </div>
                            <h1><a href="<?php the_permalink() ?>"><?php the_title(); ?></a></h1>
                        </div>
                        <div class="post-description">
                            <?php the_excerpt(); ?>
                        </div>
                        <div class="post-footer">

                        </div>
                    </article>
                <?php endwhile; ?>
            <?php endif; ?>
        </div>
        <div class="home-sidebar col-lg-4 col-xs-12 pl-0 pr-0">
            <?php get_sidebar(); ?>
        </div>
        </div>
```

```
index.php ● | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archives.php | 404.php | search.php

get_header();
?>

    <div class="home-main">
        <div class="row mr-0 ml-0">
         <div class="home-posts col-lg-8 col-xs-12">
            <?php if ( have_posts() ) : ?>
                <?php while ( have_posts() ) : the_post(); ?>
                    <article class="home-post">
                        <div class="post-header">
                            <div class="post-thumbnail row ml-0 mr-0">
                                <a href="<?php the_permalink() ?>"><?php the_post_thumbnail('home-featured')
                            </div>
                            <h1><a href="<?php the_permalink() ?>"><?php the_title(); ?></a></h1>
                        </div>
                        <div class="post-description">
                            <?php the_excerpt(); ?>
                        </div>
                        <div class="post-footer row ml-0 mr-0">
                            <div class="post-meta">
                                <strong>Author: </strong> <?php the_author(); ?>
                            </div>
                            <div class="post-meta">
                                <strong>Posted on: </strong> <?php the_time(); ?>
                            </div>
                        </div>
                    </article>
                <?php endwhile; ?>
            <?php endif; ?>
        </div>
        <div class="home-sidebar col-lg-4 col-xs-12 pl-0 pr-0">
            <?php get_sidebar(); ?>
        </div>
```

```
template-topics.php | content-single.php | index.php × | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archi

<?php
/*
* This is our first theme.
*/
get_header();
?>

    <div class="home-main">
        <div class="custom-header">
            <img src="<?php header_image(); ?>" width="<?php echo absint(get_custom_header()->width) ?>"
            width="<?php echo absint(get_custom_header()->height) ?>" class="img-fluid">
        </div>
        <div class="row mr-0 ml-0">
         <div class="home-posts col-lg-8 col-xs-12">
            <?php if ( have_posts() ) : ?>
                <?php while ( have_posts() ) : the_post(); ?>
                    <article class="home-post">
                        <div class="post-header">
                            <div class="post-thumbnail row ml-0 mr-0">
                                <a href="<?php the_permalink() ?>"><?php the_post_thumbnail('home-featured') ?
                                ></a>
                            </div>
                            <h1><a href="<?php the_permalink() ?>"><?php the_title(); ?></a></h1>
                        </div>
                        <div class="post-description">
                            <?php the_excerpt(); ?>
                        </div>
```

```php
function wplearning_theme_setup() {

    add_theme_support('custom-logo');

    add_theme_support('title-tag');

    add_theme_support('post-thumbnails');

    add_image_size ('home-featured', 680, 400, array('center', 'center'));

    add_theme_support('automatic-feed-links');

    register_nav_menus( array(
        'primary'   => __( 'Primary Menu', 'wplearning' )
    ) );

    $args = array (
        'default-image' => get_template_directory_uri().'/assets/img/header-1.jpg',
        'default-text-color'    => '000',
        'width'         => 1920,
        'height'        => 400,
        'flex-width'    => true,
        'flex-height'   => true,
    );
    add_theme_support( 'custom-header', $args );

d_action('after_setup_theme', 'wplearning_theme_setup');
```

index.php

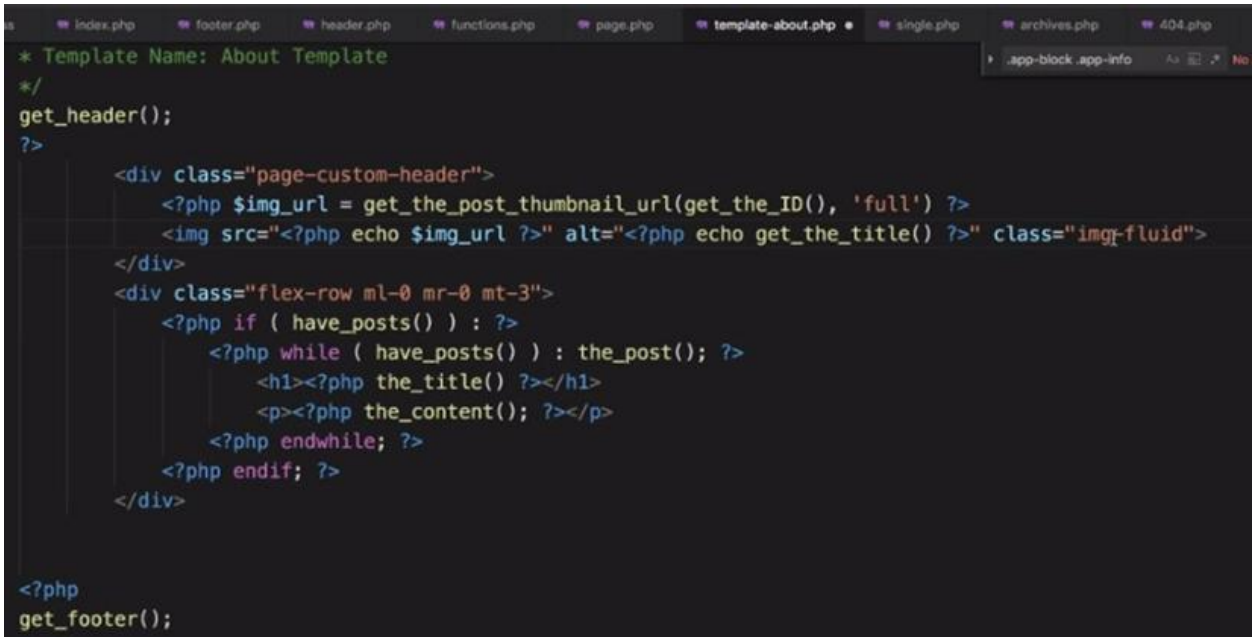
template-topics.php | content-single.php | index.php | footer.php | header.php | functions.php | page.php | single.php | wp-config.php | archives

```php
                        <div class="post-header">
                            <div class="post-thumbnail row ml-0 mr-0">
                                <a href="<?php the_permalink() ?>"><?php the_post_thumbnail('home-featured') ?><
                            </div>
                            <h1><a href="<?php the_permalink() ?>"><?php the_title(); ?></a></h1>
                        </div>
                        <div class="post-description">
                            <?php the_excerpt(); ?>
                        </div>
                        <div class="post-footer row ml-0 mr-0">
                            <div class="post-meta">
                                <strong>Author: </strong> <?php the_author(); ?>
                            </div>
                            <div class="post-meta">
                                <strong>Posted on: </strong> <?php the_time(); ?>
                            </div>
                        </div>
                    </article>
                <?php endwhile; ?>
            <?php endif; ?>
            <div class="pagination row ml-0 mr-0">
                <?php echo paginate_links() ?>
            </div>

        </div>
        <div class="home-sidebar col-lg-4 col-xs-12 pl-0 pr-0">
            <?php get_sidebar(); ?>
        </div>
    </div>
```

index.php | footer.php | header.php | functions.php | page.php | template-home.php | single.php | archives.php | 404.php

.app-block .app-info

```php
<?php
/*
* This template is used to display about us page
* Template Name: Homepage Template
*/
get_header();
?>
        <div class="page-custom-header">
            <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
            <img src="<?php echo $img_url ?>" alt="<?php echo get_the_title() ?>" class="img-fluid">
        </div>
        <div class="flex-row ml-0 mr-0 mt-3">
            <?php if ( have_posts() ) : ?>
                <?php while ( have_posts() ) : the_post(); ?>
                    <h1><?php the_title() ?></h1>
                    <p><?php the_content(); ?></p>
                <?php endwhile; ?>
            <?php endif; ?>
        </div>


<?php
get_footer();
?>
```



index.php | footer.php | header.php | functions.php | page.php | template-about.php | single.php | archives.php | 404.php

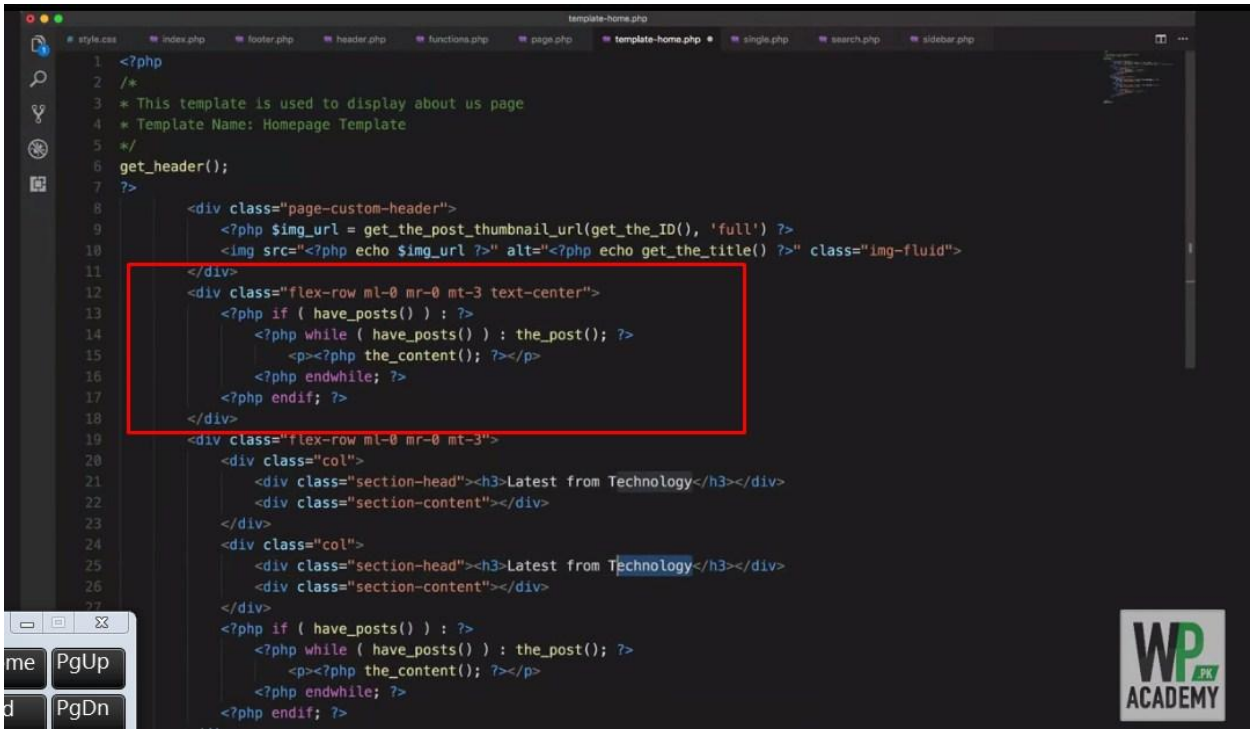
.app-block .app-info

```php
* Template Name: About Template
*/
get_header();
?>
        <div class="page-custom-header">
            <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
            <img src="<?php echo $img_url ?>" alt="<?php echo get_the_title() ?>" class="img-fluid">
        </div>
        <div class="flex-row ml-0 mr-0 mt-3">
            <?php if ( have_posts() ) : ?>
                <?php while ( have_posts() ) : the_post(); ?>
                    <h1><?php the_title() ?></h1>
                    <p><?php the_content(); ?></p>
                <?php endwhile; ?>
            <?php endif; ?>
        </div>


<?php
get_footer();
```

template-topics.php | content-single.php | index.php | footer.php | header.php | functions.php | page.php | single.php

```php
<?php
/*
* This template is used display single posts
*/
get_header();
?>
    <div class="post-content mt-20">

        <?php if ( have_posts() ) : ?>
            <?php while ( have_posts() ) : the_post(); ?>
            <div class="post-image">
                <?php the_post_thumbnail('home-featured') ?>
            </div>
            <div class="post-title">
                <h1><?php the_title() ?></h1>
            </div>
            <div class="post-meta-row">
                <div class="post-meta">
                    <strong>Author: </strong> <?php the_author(); ?>
                </div>
                <div class="post-meta">
                    <strong>Posted on: </strong> <?php the_time(); ?>
                </div>
            </div>
            <div class="post-content">
                <?php the_content() ?>
            </div>
            <?php endwhile; ?>
        <?php endif; ?>
    </div>
```

```php
* This template is used display header
*/
?>
<!DOCTYPE html>
    <html>
    <meta charset="UTF-8">
    <head>
        <?php wp_head() ?>
        <style type="text/css">
            .site-navigation {
                background: <?php echo get_theme_mod('wplearning_nav_bg_color', '#2ca358') ?>
            }
        </style>
    </head>
    <body>
        <div class="site-main container">
            <header class="site-header">
                <div class="site-branding">
                    <?php the_custom_logo() ?>
                </div>
            </header>
            <nav class="site-navigation">
                <?php
                wp_nav_menu( array(
                    'theme_location' => 'primary'
                ) );
                ?>
            </nav>
```


template-home.php
style.css
index.php
footer.php
header.php
functions.php
page.php
template-home.php
single.php
search.php
sidebar.php
<?php
/*
* This template is used to display about us page
* Template Name: Homepage Template
*/
get_header();
?>

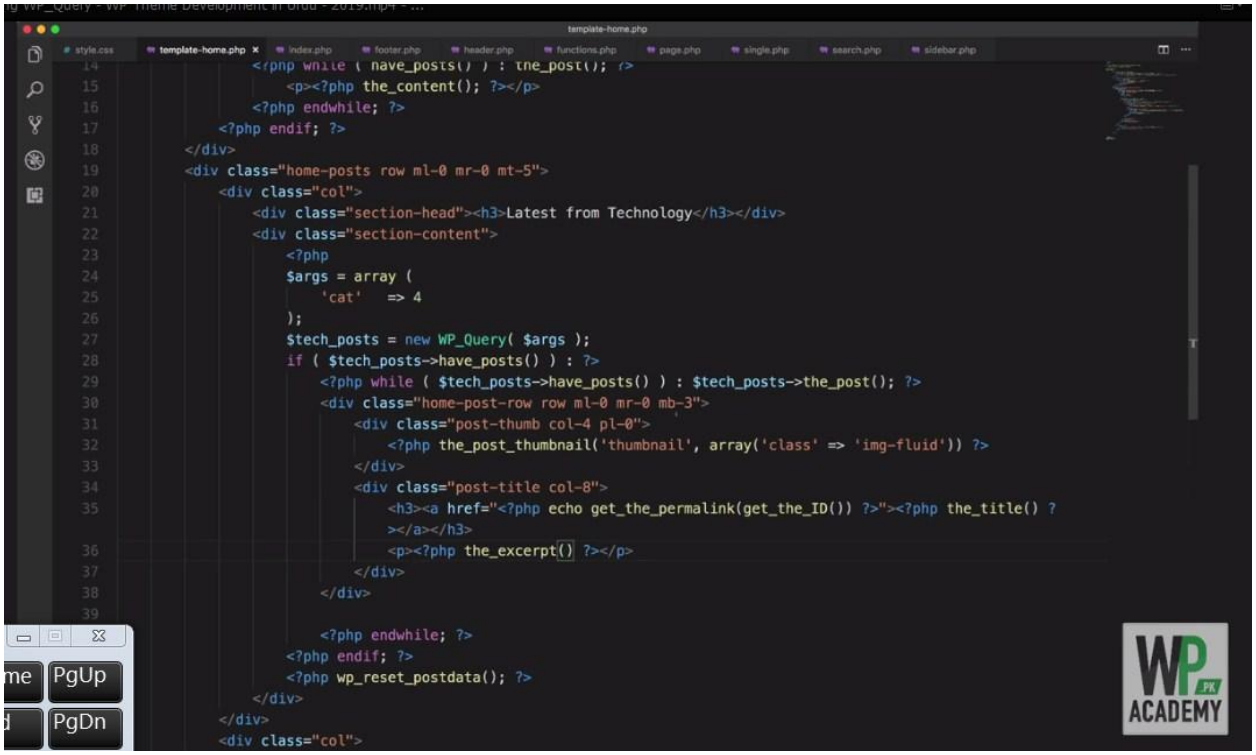
<?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
<img src="<?php echo $img_url ?>" alt="<?php echo get_the_title() ?>" class="img-fluid">

<?php if (have_posts()) : ?>
<?php while (have_posts()) : the_post(); ?>
<?php the_content(); ?>
<?php endwhile; ?>
<?php endif; ?>

<h3>Latest from Technology</h3>

<h3>Latest from Technology</h3>

<?php if (have_posts()) : ?>
<?php while (have_posts()) : the_post(); ?>
<?php the_content(); ?>
<?php endwhile; ?>
<?php endif; ?>
me
PgUp
d
PgDn
WP PK ACADEMY


template-home.php
style.css
template-home.php
index.php
footer.php
header.php
functions.php
page.php
single.php
search.php
sidebar.php
<?php the_content(); ?>
<?php endwhile; ?>
<?php endif; ?>

<h3>Latest from Technology</h3>

<?php
$args = array (
'cat' => 4
);
$tech_posts = new WP_Query($args);
if ($tech_posts->have_posts()) : ?>
<?php while ($tech_posts->have_posts()) : $tech_posts->the_post(); ?>

<?php the_post_thumbnail('thumbnail', array('class' => 'img-fluid')) ?>

<h3><a href="<?php echo get_the_permalink(get_the_ID()) ?>"><?php the_title() ?
></h3>
<?php the_excerpt() ?>

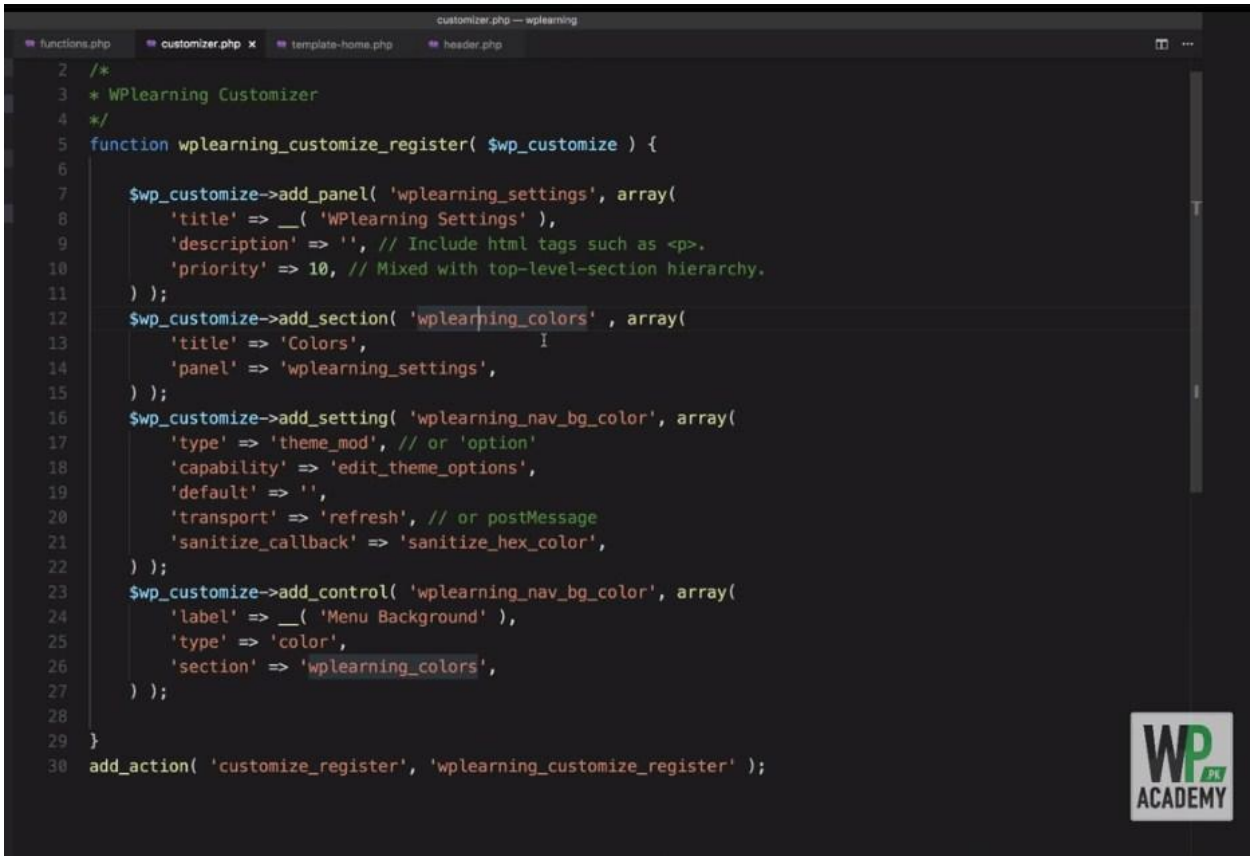
<?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>

me
PgUp
d
PgDn
WP PK ACADEMY

```
}
add_action( 'customize_register', 'themeslug_customize_register' );
```

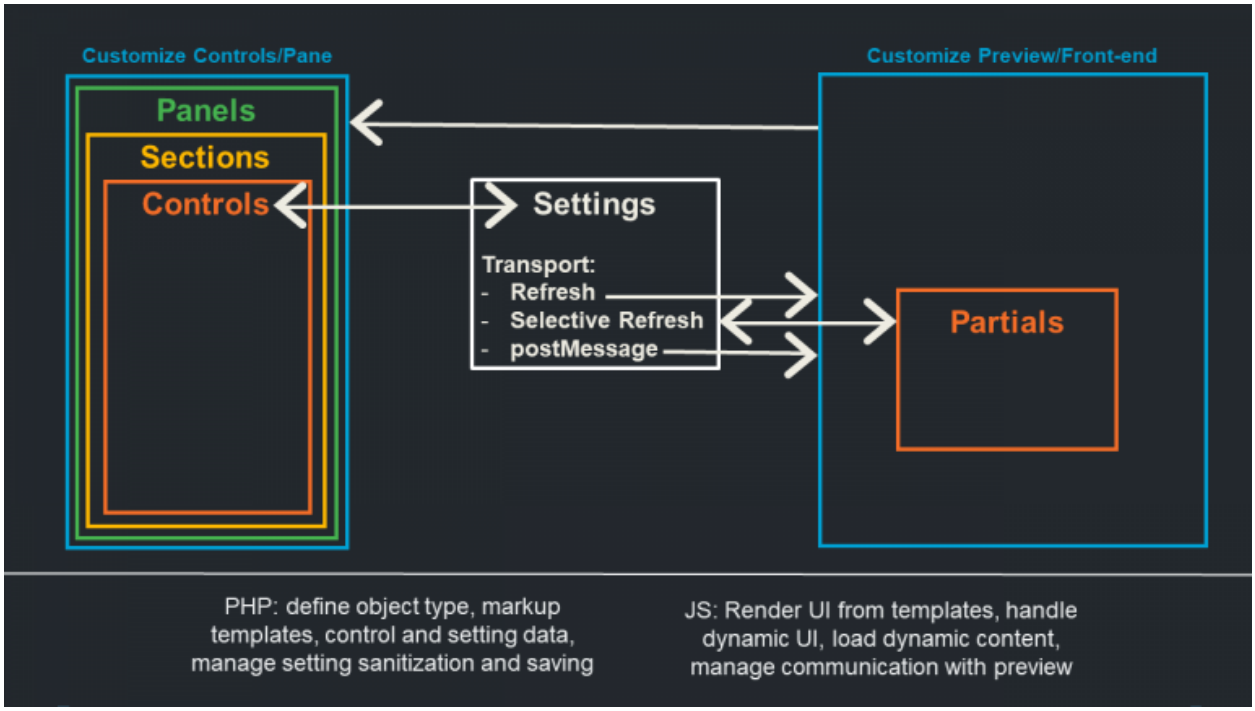
The Customizer Manager provides add_, get_, and remove_ methods for each Customizer object type; each works with an id. The get_ methods allow for direct modification of parameters specified when adding a control.

```
add_action('customize_register','my_customize_register');
function my_customize_register( $wp_customize ) {
  $wp_customize->add_panel();
  $wp_customize->get_panel();
  $wp_customize->remove_panel();

  $wp_customize->add_section();
  $wp_customize->get_section();
  $wp_customize->remove_section();

  $wp_customize->add_setting();
  $wp_customize->get_setting();
  $wp_customize->remove_setting();

  $wp_customize->add_control();
  $wp_customize->get_control();
  $wp_customize->remove_control();
}
```

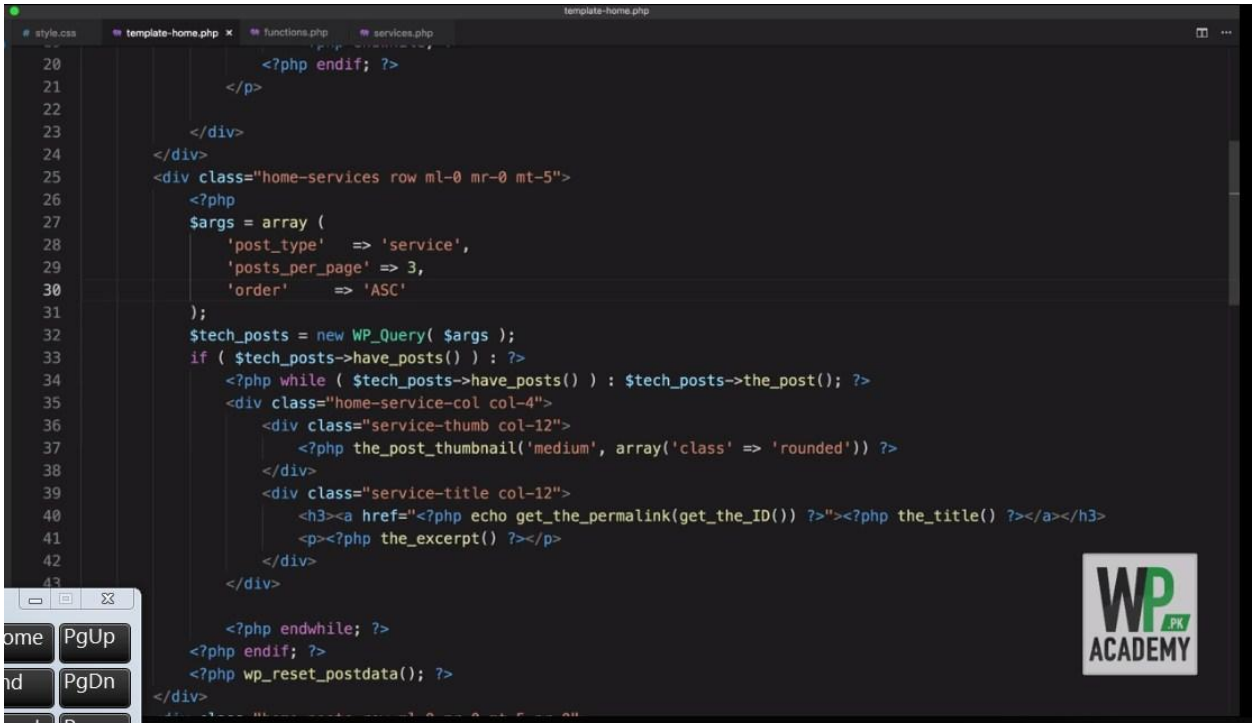


functions.php | customizer.php | style.css | template-home.php | header.php

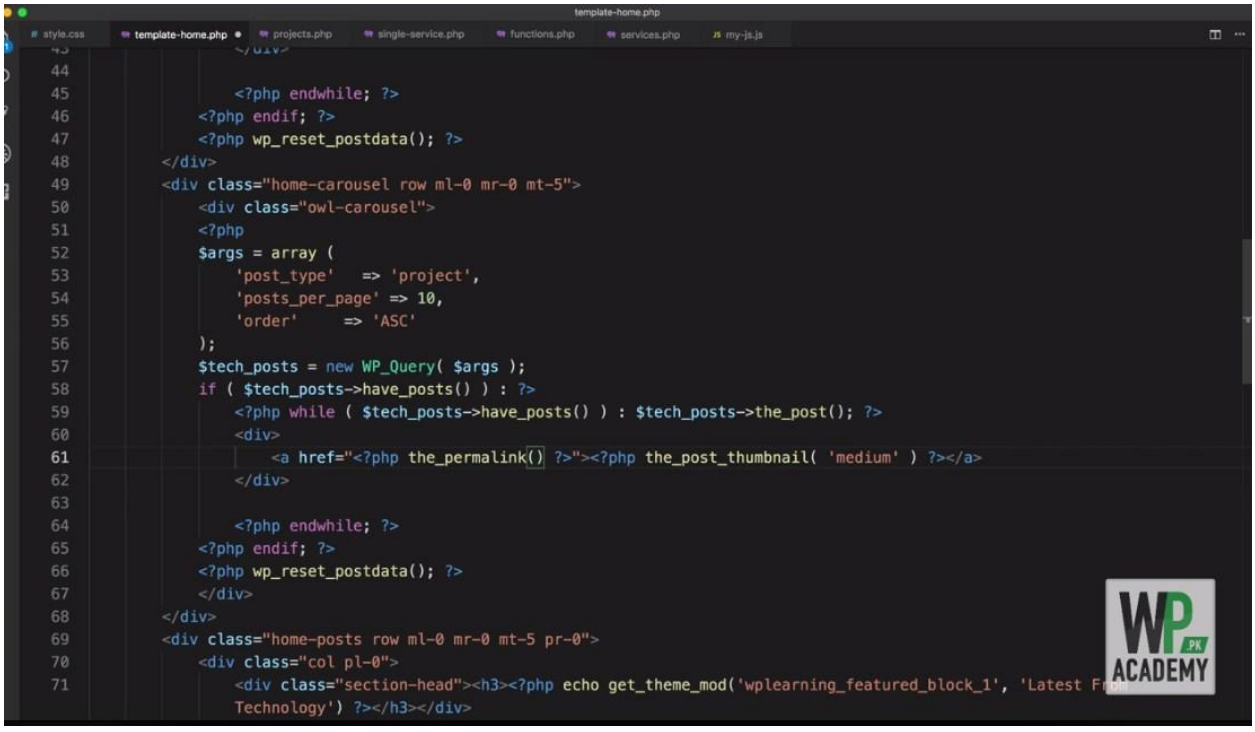
```
        'section' => 'wplearning_labels',
    ) );

    $wp_customize->add_setting( 'wplearning_home_banner', array(
        'type' => 'theme_mod', // or 'option'
        'capability' => 'edit_theme_options',
        'default' => 'Latest From Technology',
        'transport' => 'refresh', // or postMessage
        'sanitize_callback' => 'esc_attr',
    ) );
    $wp_customize->add_control( 'wplearning_home_banner', array(
        'label' => __( 'Show/Hide Home Banner' ),
        'type' => 'radio',
        'section' => 'wplearning_misc',
        'choices' => array (
            'yes'   => 'Yes',
            'no'    => 'No'
        )
    ) );
}
add_action( 'customize_register', 'wplearning_customize_register' );
```



```php
<?php
/*
* This template is used to display about us page
* Template Name: Homepage Template
*/
get_header();
?>
        <div class="page-custom-header">
        <?php
            $banner_status = get_theme_mod('wplearning_home_banner', 'yes');
        ?>
        <?php if($banner_status == "yes"): ?>
            <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
            <img src="<?php echo $img_url ?>" alt="<?php echo get_the_title() ?>" class="img-fluid">
        <?php endif; ?>
        </div>
        <div class="flex-row ml-0 mr-0 mt-3 text-center">
            <div class="home-main-content">
                <h1>WHO WE ARE</h1>
                <p>
                    <?php if ( have_posts() ) : ?>
                        <?php while ( have_posts() ) : the_post(); ?>
                            <p><?php the_content(); ?></p>
                        <?php endwhile; ?>
                    <?php endif; ?>
                </p>
```



```php
                    <?php endif; ?>
                </p>

            </div>
        </div>
        <div class="home-services row ml-0 mr-0 mt-5">
            <?php
            $args = array (
                'post_type'   => 'service',
                'posts_per_page' => 3,
                'order'     => 'ASC'
            );
            $tech_posts = new WP_Query( $args );
            if ( $tech_posts->have_posts() ) : ?>
                <?php while ( $tech_posts->have_posts() ) : $tech_posts->the_post(); ?>
                <div class="home-service-col col-4">
                    <div class="service-thumb col-12">
                        <?php the_post_thumbnail('medium', array('class' => 'rounded')) ?>
                    </div>
                    <div class="service-title col-12">
                        <h3><a href="<?php echo get_the_permalink(get_the_ID()) ?>"><?php the_title() ?></a></h3>
                        <p><?php the_excerpt() ?></p>
                    </div>
                </div>

                <?php endwhile; ?>
            <?php endif; ?>
            <?php wp_reset_postdata(); ?>
        </div>
```

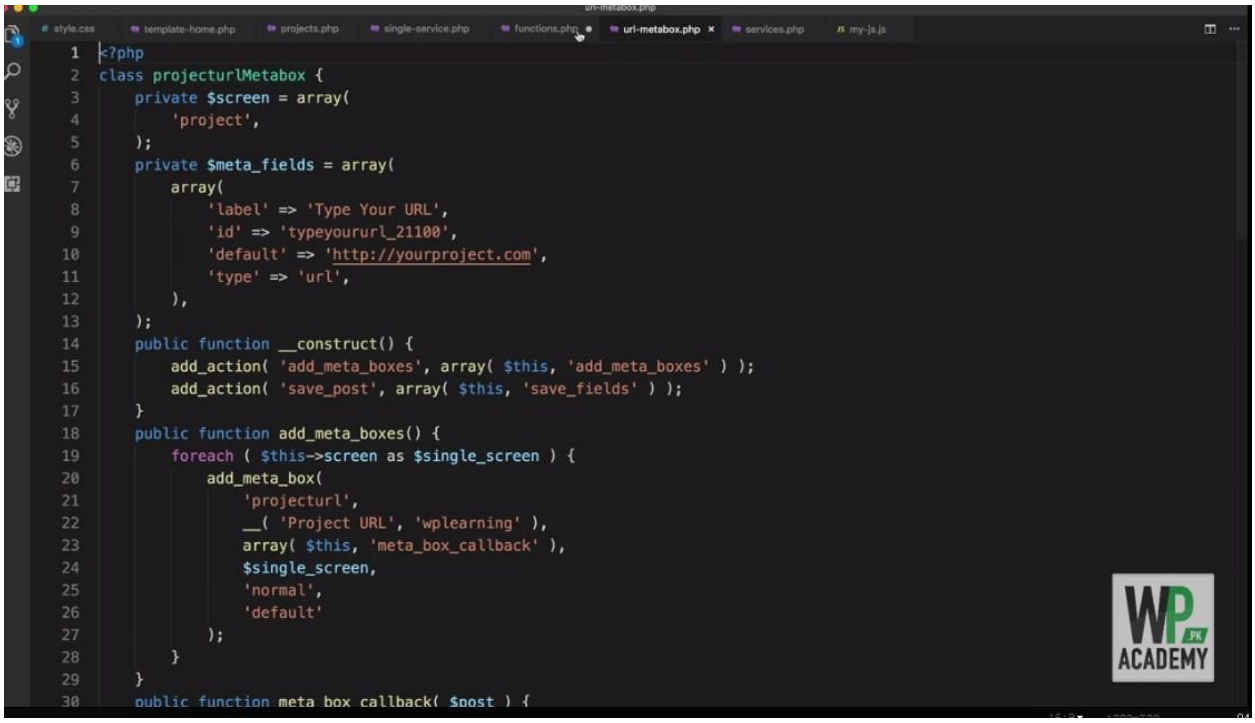

template-home.php
style.css
template-home.php
projects.php
single-service.php
functions.php
services.php
my-js.js
<?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>

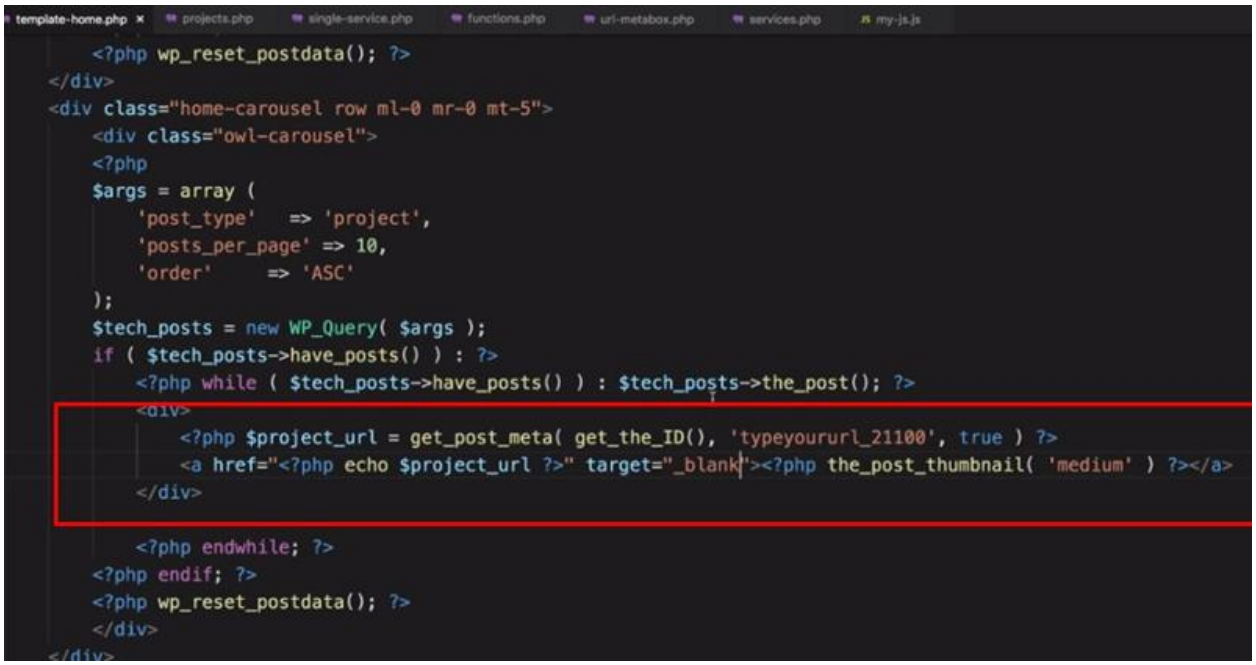
<?php
$args = array (
'post_type' => 'project',
'posts_per_page' => 10,
'order' => 'ASC'
);
$tech_posts = new WP_Query($args);
if ($tech_posts->have_posts()) : ?>
<?php while ($tech_posts->have_posts()) : $tech_posts->the_post(); ?>

<a href="<?php the_permalink() ?>"><?php the_post_thumbnail('medium') ?>

<?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>

<h3><?php echo get_theme_mod('wplearning_featured_block_1', 'Latest
Technology') ?></h3>
WP .PK ACADEMY

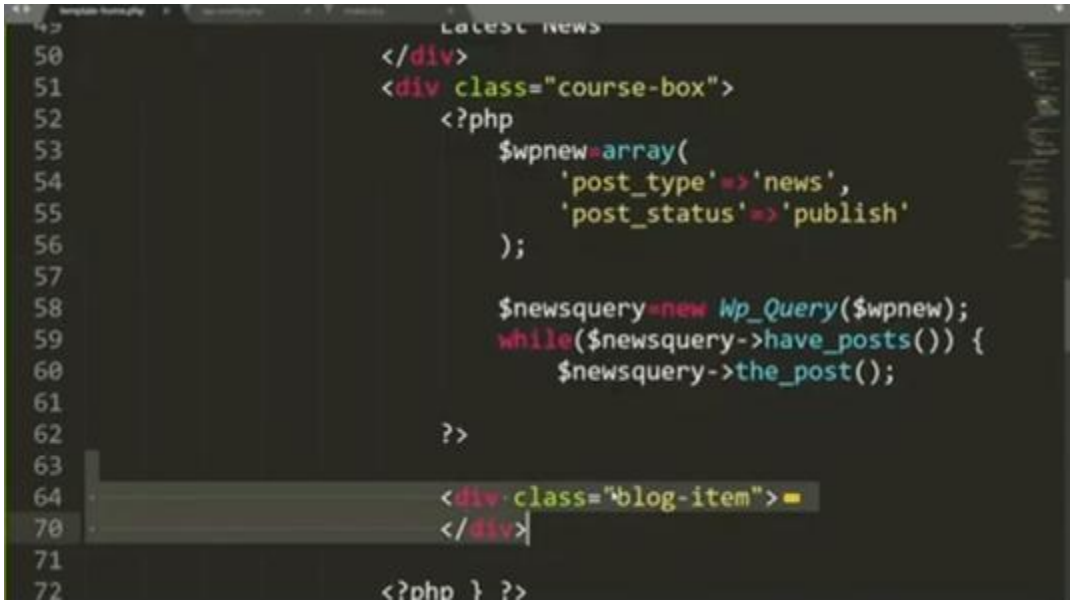

style.css
template-home.php
projects.php
single-service.php
functions.php
url-metabox.php
services.php
my-js.js
<?php
class projecturlMetabox {
private $screen = array(
'project',
);
private $meta_fields = array(
array(
'label' => 'Type Your URL',
'id' => 'typeyoururl_21100',
'default' => 'http://yourproject.com',
'type' => 'url',
),
);
public function __construct() {
add_action('add_meta_boxes', array($this, 'add_meta_boxes'));
add_action('save_post', array($this, 'save_fields'));
}
public function add_meta_boxes() {
foreach ($this->screen as $single_screen) {
add_meta_box(
'projecturl',
__('Project URL', 'wplearning'),
array($this, 'meta_box_callback'),
$single_screen,
'normal',
'default'
);
}
}
public function meta_box_callback($post) {
WP .PK ACADEMY


template-home.php
projects.php
single-service.php
functions.php
uri-metabox.php
services.php
my-js.js
<?php wp_reset_postdata(); ?>

<?php
$args = array (
'post_type' => 'project',
'posts_per_page' => 10,
'order' => 'ASC'
);
$tech_posts = new WP_Query($args);
if ($tech_posts->have_posts()) : ?>
<?php while ($tech_posts->have_posts()) : $tech_posts->the_post(); ?>

<?php $project_url = get_post_meta(get_the_ID(), 'typeyoururl_21100', true) ?>
<a href="<?php echo $project_url ?>" target="_blank"><?php the_post_thumbnail('medium') ?>

<?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>


50
51
52 <?php
53 $wpnew=array(
54 'post_type'=>'news',
55 'post_status'=>'publish'
56);
57
58 $newsquery=new Wp_Query($wpnew);
59 while($newsquery->have_posts()) {
60 $newsquery->the_post();
61
62 ?>
63
64
70
71
72 <?php } ?>

my-plugin.php

wp-content > plugins > my-plugin > my-plugin.php

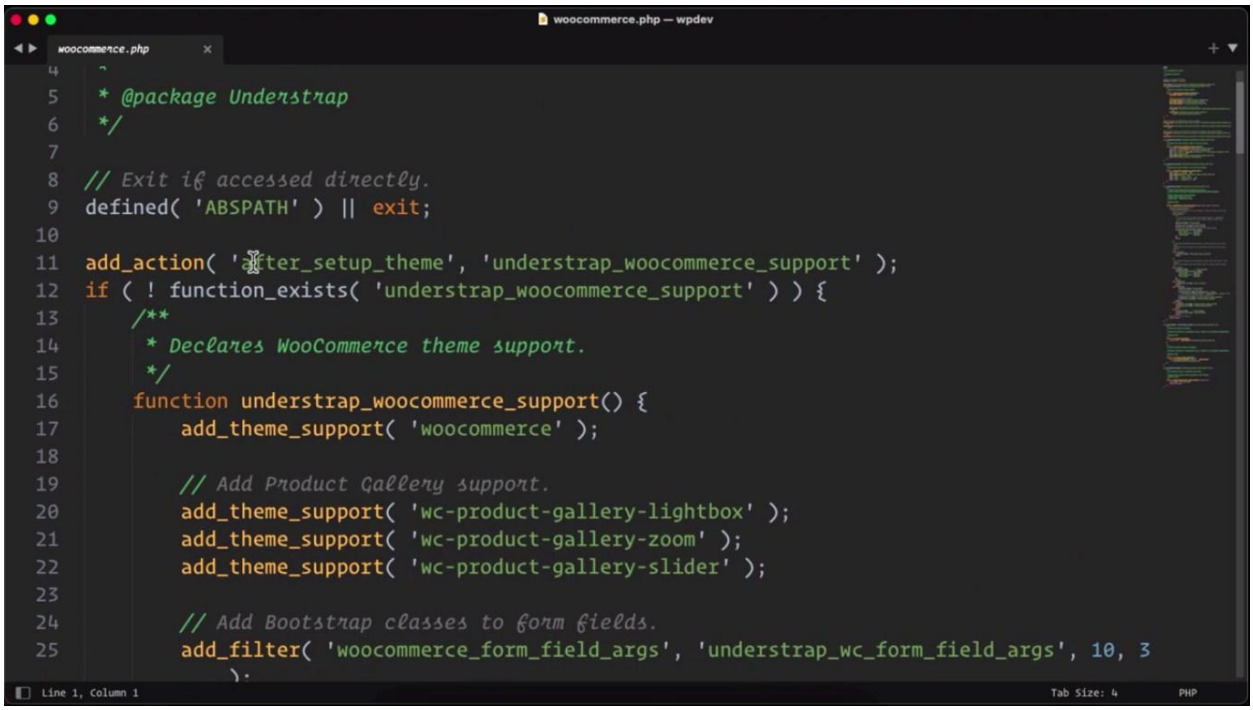
```php
add_action('wp_logout', 'redirect_after_loguot');


function my_meta_fields(){
    ?>
    <label for="my-meta-field1">My Meta Field 1</label>
    <input type="text" name="my-meta-field1" id="my-meta-field1" />
    <?php
}

function add_my_meta_box(){
    add_meta_box('my-meta-box', 'My Meta Box', 'my_meta_fields', 'cars');
}
add_action('add_meta_boxes', 'add_my_meta_box');
```

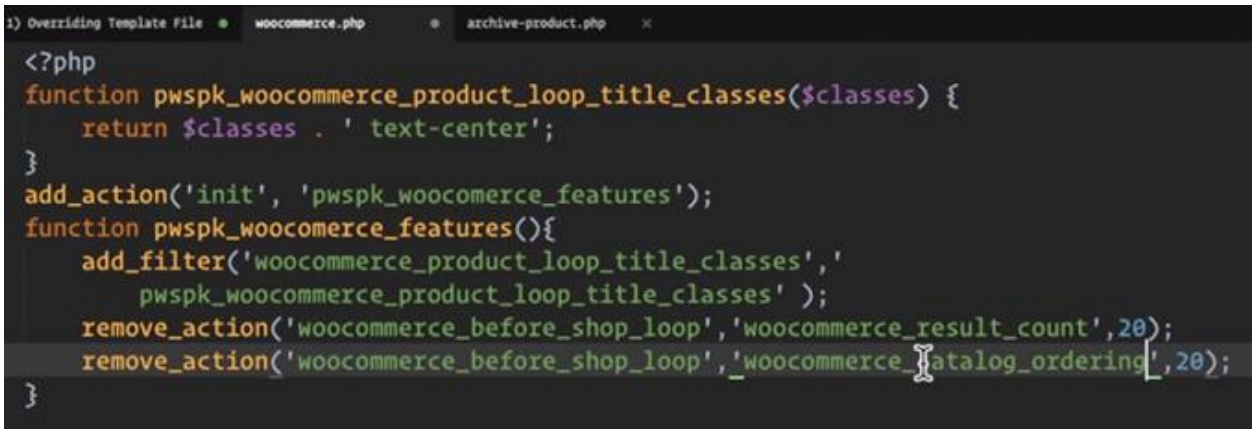
Checklist of Technical SEO

WSCUBE TECH

1. Sitemap.XML
2. Sitemap.HTML
3. Robots.txt File
4. Page Load Time
5. Optimisation of JS & CSS
6. SSL Certificate
7. Canonical Tag
8. Redirection (404, 301, 302)
9. W3C Validation
10. Open Graph Tag

```
1) Overriding Template File
2) Using Hooks ( filter hooks / action hooks)
3) Adding extra css/js to apply changes
4) function_exists() / class_exists()
```





```php
<?php
// Start the loop.
while ( have_posts() ) : the_post();
    ?>
    <header class="entry-header">
        <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
    </header><!-- .entry-header -->
    <div class="entry-content">
        <form method="post">
            <input type="text" name="uemail" placeholder="Enter Email">
            <input type="hidden" name="_nonce" value="<?php echo wp_create_nonce('
                submit-user') ?>">
            <input type="submit" name="submit" value="Submit">
        </form>
    </div>
    <?php
// End the loop.
endwhile;
?>
```

```php
<?php
if ( isset($_GET['my-nonce']) && wp_verify_nonce($_GET['my-nonce'], 'page-settings') ) {
    //safe
} else {
    die('Security check!');
}

get_header(); ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

        <?php
        // Start the loop.
        while ( have_posts() ) : the_post();
            ?>
            <header class="entry-header">
                <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
            </header><!-- .entry-header -->
            <div class="entry-content">
```

File Edit Selection Find View Goto Tools Project Preferences Help

page-how-to-use-nonce.php

```php
<?php
if ( isset($_POST['submit']) ) {
    if ( wp_verify_nonce($_POST['_nonce'], 'submit-user') ) {
        //safe to submit data
    } else {
        die('security check!');
    }
}

get_header(); ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

        <?php
        // Start the loop.
        while ( have_posts() ) : the_post();
            ?>
            <header class="entry-header">
                <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
            </header><!-- .entry-header -->
```

```php
<?php
get_header(); ?>

    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

        <?php
        // Start the loop.
        while ( have_posts() ) : the_post();
            ?>
            <header class="entry-header">
                <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
            </header><!-- .entry-header -->
            <div class="entry-content">
                <a href="<?php echo wp_nonce_url(get_bloginfo('url').'/settings', 'page-
                    settings', 'my-nonce') ?>">Go to settings page</a>
            </div>
            <?php
        // End the loop.
        endwhile;
```

Method 1: `register_uninstall_hook`

To set up an uninstall hook, use the register_uninstall_hook() function:

```php
register_uninstall_hook(__FILE__, 'pluginprefix_function_to_run');
```

my-plugin.php | uninstall.php ● | index.php

wp-content > plugins > my-plugin > uninstall.php

```php
<?php

if(!defined('WP_UNINSTALL_PLUGIN')){
    header("Location: /youtube");
    die();
}

/
```

```php
register_activation_hook(__FILE__, 'my_plugin_activation');

function my_plugin_deactivation(){
    global $wpdb, $table_prefix;
    $wp_emp = $table_prefix.'emp';

    $q = "TRUNCATE `$wp_emp`";
    $wpdb->query($q);
}
register_deactivation_hook(__FILE__, 'my_plugin_deactivation');
```

my-plugin.php × uninstall.php ×

wp-content > plugins > my-plugin > uninstall.php

```php
<?php

if(!defined('WP_UNINSTALL_PLUGIN')){
    header("Location: /youtube");
    die();
}

global $wpdb, $table_prefix;

$wp_emp = $table_prefix.'emp';

$q = "DROP TABLE `$wp_emp`;";
$wpdb->query($q);
```

```php
function my_shortcode($atts){

    return 'This is from Shortcode'. $atts['msg'];
}
add_shortcode('youtube', 'my_shortcode');

```

```php
function my_shortcode($atts){

    $atts = shortcode_atts(array(
        'msg'   =>  'I am Good'
    ), $atts);

    return 'Result : '. $atts['msg'];
}
add_shortcode('youtube', 'my_shortcode');
```

```php
function my_shortcode($atts){
    $atts = array_change_key_case($atts, CASE_LOWER);

    $atts = shortcode_atts(array(
        'msg'   =>  'I am Good',
        'note'  =>  'default'
    ), $atts);

    ob_start();
    ?>
    <h1>Youtube</h1>
    <?php
    $html = ob_get_clean();

    return $html;
}
add_shortcode('youtube', 'my_shortcode');
```

```php
function my_shortcode($atts){
    $atts = array_change_key_case($atts, CASE_LOWER);

    $atts = shortcode_atts(array(
        'msg'   =>  'I am Good',
        'note'  =>  'default'
    ), $atts);

    include 'notice.php';
}
add_shortcode('youtube', 'my_shortcode');
```

```php

function my_register_form(){
    ob_start();
    include 'public/register.php';
    return ob_get_clean();
}
add_scortcode('my-register-form', 'my_register_form');
```

```php
$args = array(
    'post_type' => 'post',
    'posts_per_page' => 3,
    'offset' => 0,
    'orderby' => 'ID',
    'order' => 'ASC',
    // 'tag'=> 'river-view,with-lawn',
    'tax_query' => array(
        'relation' => 'OR',
        array(
            'taxonomy' => 'category',
            'field' => 'slug',
            'terms' => array('flat'),
            'operator' => 'NOT IN'
        ),
        array(
            'taxonomy' => 'category',
            'field' => 'slug',
            'terms' => array('Plot')
        )
    ),
);
```

Inside the Loop variables

Top ↑

While inside the loop, these globals are set, containing information about the current post being processed.

- `$post` (WP_Post): The post object for the current post. Object described in WP_Post Class Reference.
- `$posts`: Used by some core functions, not to be mistaken for `$query->$posts`.
- `$authordata` (WP_User): The author object for the current post. Object described in WP_User Class Reference.
- `$currentday` (string): Day that the current post was published.
- `$currentmonth` (string): Month that the curent post was published.
- `$page` (int): The page of the current post being viewed. Specified by the query var page.
- `$pages` (array): The content of the pages of the current post. Each page elements contains part of the content separated by the `<!--nextpage-->` tag.
- `$multipage` (boolean): Flag to know if the current post has multiple pages or not. Returns `true` if the post has multiple pages, related to `$pages`.
- `$more` (boolean): Flag to know if WordPress should enforce the `<!--more-->` tag for the current post. WordPress will not enforce the more tag if `true`.
- `$numpages` (int): Returns the number of pages in the post, related to `$pages`.

```
242
243  function my_register_form(){
244      ob_start();
245      include 'public/register.php';
246      return ob_get_clean();
247  }
248  add_scortcode('my-register-form', 'my_register_form');
```

Nonce in WordPress

A WordPress nonce is a “number used once” security token to protect URLs and forms from malicious attacks. It helps WordPress to determine whether a request is valid, preventing unauthorized actions and inputs.
These nonce-related functions are essential for adding security to WordPress forms, AJAX requests, and other actions to protect against Cross-Site Request Forgery (CSRF) attacks. Depending on your specific use case, you'll choose the appropriate function to generate and verify nonces in your WordPress code.

1. `wp_nonce_field()`: Generates a nonce and outputs it as a hidden form field. This function is often used in HTML forms to include a nonce for verification when the form is submitted.
2. `wp_create_nonce($action)`: Generates a nonce for a specific action. You provide a unique identifier (action) to associate the nonce with a particular operation.
3. `wp_verify_nonce($nonce, $action)`: Verifies that a nonce matches the provided action. It returns `true` if the nonce is valid and `false` if it is not.
4. `check_admin_referer($action, $query_arg)`: Verifies a nonce used in the administration area. It checks if the nonce in the request matches the one associated with the given action.
5. `check_ajax_referer($action, $query_arg, $die)`: Similar to `check_admin_referer()`, but specifically designed for AJAX requests. It verifies that the nonce in the request matches the one associated with the given action.
6. `wp_nonce_url($url, $action, $name)`: Adds a nonce to a URL. This function can be used to create secure URLs for actions that require nonces.
7. `wp_nonce_ays($action)`: Outputs a "Are you sure you want to do this?" message along with a nonce verification field. It's often used in confirmation dialogs before performing critical actions.
8. `wp_nonce_tick()`: Returns the current "tick" value, which is used to generate and validate nonces. It's primarily an internal function and is not typically used directly in plugins or themes.

Use cases for nonces

- **Form submissions (e.g., saving settings, updating a post).**
- **AJAX requests that modify data.**
- **Logins and password resets.**
- **Actions related to user authentication and authorization.**
- **REST API**
- **against Cross-Site Request Forgery (CSRF) attacks**

```
        </label>
        <input type="text" name="txtemail" value="<?php echo isset($row_details['e
    </p>
    <?php //wp_nonce_field("student_action_nonce", "student_name_nonce"); ?>
    <p>
        <button type="submit" name="btnsubmit">Submit</button>
    </p>
</form>

<script>

    jQuery(function () {
        jQuery("#frmsubmitdata").validate({
            submitHandler: function () {
                var nonce = "<?php echo wp_create_nonce('student_nonce'); ?>";
                var postdata = jQuery("#frmsubmitdata").serialize() + "&action=mya
                jQuery.post("<?php echo admin_url('admin-ajax.php') ?>", postdata,
                    console.log(response);
                })
            }
        });
    });
```

```
add_action("wp_ajax_myajax", "myjaxfunction");

function myjaxfunction() {
    check_ajax_referer('student_nonce1');
    echo "This is running";
    /*if (wp_verify_nonce($_REQUEST['student_name_nonce'], "student_action_nonce")
        echo "Status = 1";
    } else {
        echo "Status = 0";
    }*/
    wp_die();
}
```

How to use nonce in wp

1.Generate a Nonce:

2. Include the Nonce in Your Form or Request

```
<form method="post" action="">
<!-- Other form fields -->
<?php wp_nonce_field('my_action', 'my_nonce'); ?>
<input type="submit" value="Submit">
</form>
```

```
In an AJAX request:
var myNonce = '<?php echo $nonce; ?>';
var data = {
    action: 'my_ajax_action',
    nonce: myNonce,
    // Other data to send

};
```

3. Verify the Nonce: When processing the form submission or handling the AJAX request on the server side, you should verify the nonce to ensure that the request is legitimate and authorized. Use the `check_admin_referer()` or `check_ajax_referer()` function to do this.

```
if ( ! isset( $_POST['my_nonce'] ) || ! wp_verify_nonce( $_POST['my_nonce'], 'my_action' ) ) {
    // Nonce verification failed, handle the error or exit.}
```

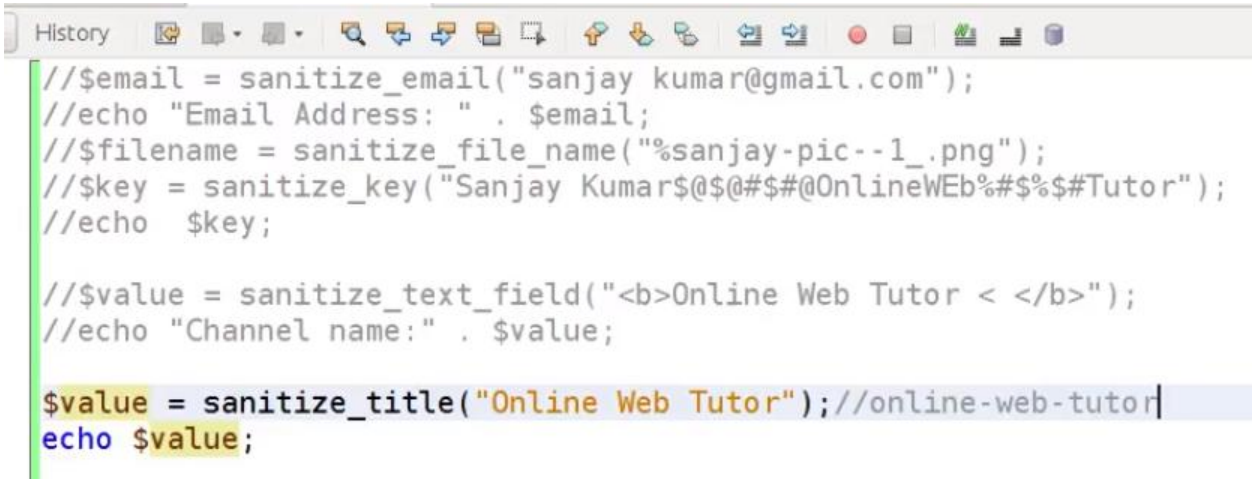
Or for AJAX:

```
if ( ! check_ajax_referer( 'my_action', 'nonce', false ) ) {
    // Nonce verification failed, handle the error or exit.
}
```

4.Perform the Authorized Action: If the nonce verification passes, you can proceed to perform the authorized action, such as saving data, updating settings, or any other operation.

Sanitizing input data to enhance security

WordPress provides several built-in functions for sanitizing input data to enhance security and prevent various types of vulnerabilities, such as SQL injection and Cross-Site Scripting (XSS) attacks. Here is a list of some of the key sanitization functions in WordPress:



1. **sanitize_text_field($str)**: Sanitizes a string by removing potentially harmful characters and tags. This function is often used to sanitize user input from text fields.
2. **sanitize_email($email)**: Sanitizes an email address by removing potentially harmful characters and validating the email format.
3. **sanitize_file_name($filename)**: Sanitizes a file name by removing potentially harmful characters and ensuring it adheres to the allowed file name characters.
4. **sanitize_title($title)**: Sanitizes a title or post name by removing potentially harmful characters and transforming it into a URL-friendly slug.
5. **sanitize_title_for_query($title)**: Sanitizes a title for use in database queries, ensuring it is safe to include in SQL statements.
6. **sanitize_key($key)**: Sanitizes a key or identifier by removing potentially harmful characters, making it suitable for use in array keys or query variables.
7. **esc_html($text)**: Escapes a string for use in HTML content, preventing XSS attacks by encoding special characters.
8. **esc_attr($text)**: Escapes a string for use in HTML attributes, ensuring that special characters do not disrupt the HTML structure.
9. **esc_url($url)**: Escapes a URL to prevent potential security issues, such as injecting malicious links.
10. **esc_sql($sql)**: Escapes a SQL query string to prevent SQL injection attacks when building custom database queries.
11. **wp_kses($data, $allowed_html)**: Sanitizes and validates HTML content based on an array of allowed HTML elements and attributes.
12. **wp_kses_data($data)**: Sanitizes and validates HTML content, allowing only safe tags and attributes.
13. **wp_strip_all_tags($string, $remove_breaks)**: Strips all HTML and PHP tags from a string. You can choose to keep or remove line breaks.
14. **balanceTags($text, $force)**: Ensures that HTML tags in a string are properly balanced, preventing broken or mismatched tags.
15. **filter_var($variable, $filter, $options)**: PHP's built-in function for filtering and validating data. WordPress developers may use this function in custom code to validate various types of input.
16. **intval($var, $base)**: Casts a variable to an integer, which can be useful for sanitizing numeric input.
17. **absint($maybeint)**: Casts a value to an absolute integer, ensuring it's a non-negative integer.

Escaping output

Escaping output is a crucial security practice to prevent Cross-Site Scripting (XSS) attacks and maintain the integrity of your website. When you output data to the browser, it's important to make sure that any potentially unsafe content is properly escaped to prevent malicious code execution. WordPress provides several functions to help you escape output in different contexts. Here are some of the key functions:

esc_html($text): Use this function to escape and sanitize text for output in HTML content. It encodes special characters like `<`, `>`, `&`, and quotes (`"`) to their HTML entities.

```
echo esc_html($text);
```

esc_attr($text): This function is used to escape text for use in HTML attributes. It encodes special characters and ensures that the content is safe within an attribute.

```
echo '<input type="text" value="' . esc_attr($value) . '">';
```

esc_url($url): Use this function to escape URLs to prevent any potential security issues. It ensures that URLs are properly formatted and safe for use in links and attributes.

```
echo '<a href="' . esc_url($url) . '">Link</a>';
```

esc_js($javascript): When outputting JavaScript code, use this function to escape it properly. It can help prevent script injection vulnerabilities.

```
echo '<script>' . esc_js($javascript_code) . '</script>';
```

esc_textarea($text): Use this function to escape text for output within a `<textarea>` element. It ensures that line breaks and special characters are properly encoded.

```
echo '<textarea>' . esc_textarea($text) . '</textarea>';
```

esc_sql($sql): When constructing custom database queries, use `esc_sql()` to escape and sanitize SQL query strings. This helps prevent SQL injection attacks.

```
$query = "SELECT * FROM {$wpdb->prefix}my_table WHERE name = " . esc_sql($name);
```

wp_kses($data, $allowed_html): Use this function to sanitize and validate HTML content based on an array of allowed HTML elements and attributes. This is useful when you want to allow certain HTML tags while disallowing others.

echo wp_kses($data, $allowed_html);

wp_kses_post($data): Similar to `wp_kses()`, this function is specifically designed to sanitize and validate HTML content within WordPress posts, preserving certain tags allowed in posts.

echo wp_kses_post($data);

sanitize_text_field($text): Although primarily used for sanitizing input, this function can also be used to escape and sanitize text for output in HTML content.

echo sanitize_text_field($text);

When working with WordPress themes or plugins, it's important to choose the appropriate escaping function based on the context of your output. Always use these functions to escape user-generated content and data coming from untrusted sources to ensure the security of your WordPress website.

```php
<?php
/*
Plugin Name: Escaping Functions Demo
Description: A simple plugin to demonstrate the use of escaping functions in
WordPress.
*/

// Display a user comment safely.
function display_comment_safely() {
    $comment_text = get_comment_text(); // Get the user's comment text from
the database.
    echo esc_html($comment_text); // Display the comment text safely.
}
add_shortcode('display_comment', 'display_comment_safely');

// Display a post title safely.
function display_post_title_safely() {
    $post_title = get_the_title(); // Get the post title from the database.
    echo esc_html($post_title); // Display the post title safely.
}
```

```php
add_shortcode('display_post_title', 'display_post_title_safely');

// Display an image URL safely within an attribute.
function display_image_url_safely() {
    $image_url = get_post_meta(get_the_ID(), '_image_url', true); // Get an
image URL from custom field.
    echo '<img src="' . esc_url($image_url) . '" alt="Image">';
}
add_shortcode('display_image', 'display_image_url_safely');

// Display JavaScript code safely.
function display_javascript_safely() {
    $javascript_code = 'alert("Hello, world!");';
    echo '<script>' . esc_js($javascript_code) . '</script>';
}
add_shortcode('display_javascript', 'display_javascript_safely');

// Display text content safely within a textarea.
function display_textarea_content_safely() {
    $textarea_content = get_user_meta(get_current_user_id(), 'description',
true); // Get user description.
    echo '<textarea>' . esc_textarea($textarea_content) . '</textarea>';
}
add_shortcode('display_textarea_content', 'display_textarea_content_safely');

// Custom database query with proper escaping.
function custom_database_query() {
    $name = sanitize_text_field($_GET['name']); // Sanitize user input.
    global $wpdb;
    $query = "SELECT * FROM {$wpdb->prefix}my_table WHERE name = " .
esc_sql($name);
    // Execute the query and process the results.
}
```

`esc_html()` is a WordPress function used for escaping and sanitizing HTML content to protect against cross-site scripting (XSS) attacks. It ensures that any potentially dangerous HTML or special characters are properly encoded before being displayed on a web page. The primary purpose of **`esc_html()`** is to make user-generated or untrusted data safe for output in your WordPress templates.

Here's how you use **`esc_html()`**:

php Copy code

```php
<?php
$untrusted_html = '<script>alert("XSS Attack");</script>';
$escaped_html = esc_html($untrusted_html);
echo $escaped_html;
?>
```

- `esc_html()` – Use anytime an HTML element encloses a section of data being displayed. This will remove HTML.

Copy

```
<h4><?php echo esc_html( $title ); ?></h4>
```

- `esc_js()` – Use for inline Javascript.

Copy

```
<div onclick='<?php echo esc_js( $value ); ?>' />
```

- `esc_url()` – Use on all URLs, including those in the src and href attributes of an HTML element.

Copy

```
<img alt="" src="<?php echo esc_url( $media_url ); ?>" />
```

- `esc_attr()` – Use on everything else that's printed into an HTML element's attribute.

Copy

```
<ul class="<?php echo esc_attr( $stored_class ); ?>">
```

- `esc_textarea()` – Use this to encode text for use inside a textarea element.
- `wp_kses()` – Use to safely escape for all non-trusted HTML (post text, comment text, etc.). This preserves HTML.
- `wp_kses_post()` – Alternative version of `wp_kses()`that automatically allows all HTML that is permitted in post content.
- `wp_kses_data()` – Alternative version of `wp_kses()`that allows only the HTML permitted in post comments.

Escaping with Localization

Top ↑

Rather than using `echo` to output data, it's common to use the WordPress localization functions, such as `_e()` or `__()`.

These functions simply wrap a localization function inside an escaping function:

Copy

```
esc_html_e( 'Hello World', 'text_domain' );
// Same as
echo esc_html( __( 'Hello World', 'text_domain' ) );
```

These helper functions combine localization and escaping:

- esc_html__()
- esc_html_e()
- esc_html_x()
- esc_attr__()
- esc_attr_e()
- esc_attr_x()

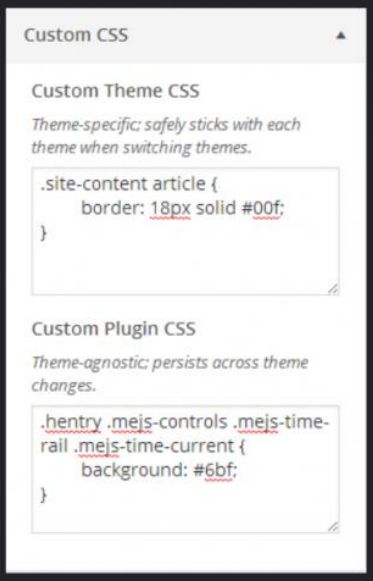
Misc

Top ↑

- `$super_admins` (array): An array of user IDs that should be granted super admin privileges (multisite). This global is only set by the site owner (e.g., in `wp-config.php`), and contains an array of IDs of users who should have super admin privileges. If set it will override the list of super admins in the database.
- `$wp_query` (object): The global instance of the WP_Query class.
- `$wp_rewrite` (object): The global instance of the WP_Rewrite class.
- `$wp` (object): The global instance of the WP environment setup class.
- `$wpdb` (object): The global instance of the wpdb class.
- `$wp_locale` (object): The global instance of the WP_Locale class.
- `$wp_admin_bar` (object): The global instance of the WP_Admin_Bar class.
- `$wp_roles` (object): The global instance of the WP_Roles class.
- `$wp_meta_boxes` (array): Object containing all registered metaboxes, including their id's, args, callback functions and title for all post types including custom.
- `$wp_registered_sidebars` (array)
- `$wp_registered_widgets` (array)
- `$wp_registered_widget_controls` (array)
- `$wp_registered_widget_updates` (array)

Theme_mods vs. Options

```
$wp_customize->add_setting(
     'custom_theme_css' , array(
          'type'      => 'theme_mod',
) );

$wp_customize->add_setting(
     'custom_plugin_css' , array(
          'type'      => 'option',
) );
```



* This is not pseudo-code, this is it. Copied and pasted from the Modular Custom CSS Plugin.

Sanitization functions

There are many functions that will help you sanitize your data.

- sanitize_email()
- sanitize_file_name()
- sanitize_hex_color()
- sanitize_hex_color_no_hash()
- sanitize_html_class()
- sanitize_key()
- sanitize_meta()
- sanitize_mime_type()
- sanitize_option()
- sanitize_sql_orderby()
- sanitize_term()
- sanitize_term_field()
- sanitize_text_field()
- sanitize_textarea_field()
- sanitize_title()
- sanitize_title_for_query()
- sanitize_title_with_dashes()
- sanitize_user()
- sanitize_url()
- wp_kses()
- wp_kses_post()

Escaping with Localization

Top ↑

Rather than using `echo` to output data, it's common to use the WordPress localization functions, such as `_e()` or `__()`.

These functions simply wrap a localization function inside an escaping function:

Copy

```
esc_html_e( 'Hello World', 'text_domain' );
// Same as
echo esc_html( __( 'Hello World', 'text_domain' ) );
```

These helper functions combine localization and escaping:

- esc_html__()
- esc_html_e()
- esc_html_x()
- esc_attr__()
- esc_attr_e()
- esc_attr_x()

Escaping any numeric variable used anywhere

Top ↑

```
echo $int;
```

Depending on whether it is an integer or a float, `(int)`, `absint()`, `(float)` are all correct and acceptable.
At times, `number_format()` or `number_format_i18n()` might be more appropriate.

`intval()`, `floatval()` are acceptable, but are outdated (PHP4) functions.

Escaping arbitrary variable within HTML attribute

Top ↑

```
echo '<div id="', $prefix, '-box', $id, '">';
```

Internationalization

Internationalization (often abbreviated as i18n) is the process of making your WordPress plugin translatable, allowing it to be easily translated into different languages. This is done using the built-in localization functions provided by WordPress. Here's a step-by-step guide on how to add internationalization to a WordPress plugin:

Prepare Your Plugin for Translation:
Before you can translate your plugin, you need to make sure it's ready for localization. This involves wrapping all translatable text in your plugin with translation functions. Replace any hardcoded strings with these functions.
Example:

```
// Hardcoded string (not translatable):

echo 'Hello, World!';

// Translatable string:

echo __('Hello, World!', 'your-plugin-text-domain');
```

Ensure that you replace all such strings with `__()` or `_e()` functions.
Load Your Text Domain:
In your plugin file (e.g., `your-plugin.php`), you need to load your plugin's text domain. This is typically done in the `init` or `plugins_loaded` action hook. The text domain should match the unique identifier of your plugin.
Example:

```
function load_plugin_textdomain() {

    load_plugin_textdomain('your-plugin-text-domain', false, dirname(plugin_basename(__FILE__)) . '/languages/');

}

add_action('init', 'load_plugin_textdomain');
```

In this example, replace `'your-plugin-text-domain'` with a unique identifier for your plugin. The third argument specifies the path to the directory where your translation files (`.mo` files) are stored.

Create Translation Files:

Next, you need to create translation files for your plugin. These files will contain translations for different languages. Use a tool like Poedit to create and manage translation files. Save these files with the `.po` extension.

Example file structure:

/your-plugin-directory/

├── your-plugin.php

└── languages/

├── your-plugin-text-domain-en_US.po (English)

└── your-plugin-text-domain-fr_FR.po (French)

Generate .mo Files:

After you've translated your strings in the `.po` files, you need to compile them into `.mo` files. Many translation tools, like Poedit, can do this for you. These `.mo` files contain the actual translated text.

Add Translations to Your Plugin:

Upload the compiled `.mo` files to the `languages/` directory of your plugin.

Use Translation Functions in Your Plugin:

Now that your plugin is set up for internationalization, use the translation functions (`__()` and `_e()`) to display text throughout your plugin.

Example:

```
// Display a translatable string:

echo __('Hello, World!', 'your-plugin-text-domain');
```

Create and Distribute Language Files:

Encourage users to translate your plugin into their languages using translation tools like GlotPress or by editing the `.po` files directly. You can also contribute your translations to the WordPress.org translation community.

By following these steps, you've made your WordPress plugin translatable and ready for localization, allowing users from different regions to use your plugin in their preferred language.

```php
<?php
/*
Plugin Name: My Internationalization Plugin
Description: A simple internationalization example.
*/

// Load the plugin's text domain for translation.
function load_plugin_textdomain() {
    load_plugin_textdomain('my-i18n-plugin', false,
dirname(plugin_basename(__FILE__)) . '/languages/');
}
add_action('plugins_loaded', 'load_plugin_textdomain');

// Add a shortcode to display a translatable message.
function hello_world_shortcode() {
    return __('Hello, World!', 'my-i18n-plugin');
}
add_shortcode('hello_world', 'hello_world_shortcode');
```

1.Create a `languages` directory inside your plugin folder to store translation files.
2.Create a `.pot` file for your plugin: You can use a tool like Poedit to create a `.pot` file, which serves as a template for translations.
3.Generate translation files (`.po` and `.mo`): Use Poedit to create translation files (`.po` and `.mo`) for different languages based on the `.pot` file. Save them in the `languages` directory you created.
4.Use the `hello_world` shortcode: In your WordPress posts or pages, you can use the `[hello_world]` shortcode to display the "Hello, World!" message, which will be translatable into the user's language.

1.Activate it in your WordPress admin dashboard.
2.Create or upload translation files for your desired languages.
3.Use the `[hello_world]` shortcode in a post or page to display the translatable message.

Internationalization plugin code

```php
<?php
/**
 * Plugin Name: Sample Plugin
 * Description: A sample WordPress plugin for internationalization.
 */

// Load the text domain for translation

function sample_plugin_load_textdomain() {
    load_plugin_textdomain('sample-plugin', false,
dirname(plugin_basename(__FILE__)) . '/languages/');
}

add_action('plugins_loaded', 'sample_plugin_load_textdomain');



// Add a sample filter that displays a translatable string

function sample_plugin_filter_content($content) {
    $translated_text = __('This is a sample plugin content.', 'sample-
plugin');
    return $content . '<p>' . $translated_text . '</p>';
}

add_filter('the_content', 'sample_plugin_filter_content');
```

Localization

Ajax localization in WordPress refers to the process of dynamically loading and displaying translated content on a web page using the Asynchronous JavaScript and XML (Ajax) technique. It allows you to change the language or region of a WordPress site without requiring a full page reload. This is particularly useful for creating multilingual websites where users can switch between languages without experiencing interruptions in their browsing experience.

Here are the key components and steps involved in implementing Ajax localization in a WordPress site:

1. Internationalization and Localization (i18n and l10n):

Before implementing Ajax localization, you need to ensure that your WordPress theme and plugins support internationalization and localization (i18n and l10n). This involves using functions like `__('text', 'textdomain')` for translatable strings and preparing translation files.

2. Language Switcher:

Create a user interface element, such as a dropdown menu or flags, that allows users to select their preferred language or region.

3. Ajax Requests:

Use JavaScript and Ajax to handle user interactions with the language switcher. When a user selects a different language, send an Ajax request to the server with the selected language as a parameter.

4. Server-Side Handling:

On the server side, handle the Ajax request by detecting the selected language and setting it for the current user's session or in a cookie. You can also store the selected language in the user's profile.

5. Translation Loading:

1.When a page is loaded or reloaded, check the user's language preference (from the session, cookie, or user profile) on the server side.

2.Load the appropriate translation files and set the WordPress locale to the user's selected language.

6. Dynamically Translate Content:

With the user's preferred language set, use WordPress localization functions like `__('text', 'textdomain')` in your theme and plugins to dynamically translate content based on the current language.

7. Update Content via Ajax:

When a user switches the language using the language switcher, send another Ajax request to the server with the selected language.
1.On the server side, update the user's language preference, load the appropriate translation files, and return the translated content.
2.Use JavaScript to replace or update the content on the page without requiring a full page reload.

8. Caching and Optimization:

Implement caching mechanisms to optimize performance. Caching translated content can reduce the server load and improve page load times.

9. Testing and Debugging:

Thoroughly test your Ajax localization implementation to ensure that content is translated correctly, and there are no issues with user experience or functionality.

Implementing Ajax localization in WordPress can be complex, and it requires a good understanding of both WordPress development and JavaScript programming. There are also WordPress plugins available that can assist with multilingual functionality, such as WPML (WordPress Multilingual Plugin) and Polylang, which offer built-in support for Ajax-based language switching. These plugins can simplify the implementation process for multilingual WordPress websites.

Any specific plugins or strategies for internationalization (i18n) and localization (l10n)?

Internationalization (i18n) and localization (l10n) are essential for making your WordPress website accessible to users from different regions and languages. Here are specific plugins and strategies for implementing i18n and l10n in WordPress:

1. Use the WordPress Built-in i18n/l10n Features:

- WordPress comes with built-in support for internationalization and localization. Utilize functions like `__()` and `_e()` for translating text, and use the `load_theme_textdomain()` and `load_plugin_textdomain()` functions to load translation files.

2. WPML (WordPress Multilingual Plugin):

- WPML is a popular and feature-rich plugin for creating multilingual WordPress websites. It allows you to translate posts, pages, custom post types, and even theme and plugin texts. WPML also provides features for language switching and SEO optimization.

3. Polylang:

- Polylang is a free and user-friendly multilingual plugin for WordPress. It enables you to create multilingual content, translate posts, pages, categories, tags, and more. Polylang is known for its simplicity and integration with popular page builders.

4. TranslatePress:

- TranslatePress is another user-friendly translation plugin. It provides a visual translation editor that allows you to translate content directly from the front end of your website. TranslatePress supports SEO-friendly URLs and multilingual SEO optimization.

5. Weglot:

- Weglot is a cloud-based translation service that offers a WordPress plugin for automatic translation of your website content. It's known for its ease of use and ability to translate content into multiple languages quickly.

6. Loco Translate:

- Loco Translate is a free and lightweight plugin that enables you to translate and manage translations of your theme and plugins directly from your WordPress dashboard. It's ideal for developers and site owners who want more control over translations.

7. Poedit:

- Poedit is a standalone translation software that is widely used by developers for creating and managing translation files (PO and MO files) for WordPress themes and plugins. You can use it in conjunction with gettext functions in your code.

8. GlotPress:

- GlotPress is an open-source translation management tool developed by the WordPress community. It's used for translating WordPress core, themes, and plugins. Some companies and projects host their own GlotPress installations for collaborative translation efforts.

9. Language Switcher Plugins:

- Consider using language switcher plugins like "Polylang for WooCommerce" or "WooCommerce Multilingual" (for e-commerce sites) to provide seamless language switching and currency conversion features.

10. RTL (Right-to-Left) Language Support: - For languages that are written from right to left (e.g., Arabic, Hebrew), ensure that your theme and content are compatible with RTL layouts. Many popular themes and plugins provide RTL support out of the box.

11. Custom Styles for Different Languages: - If your website uses CSS styles specific to languages or regions, consider using a plugin or custom code to load different stylesheets based on the selected language.

12. Translation Quality Assurance: - Use professional translators or translation services for critical content to ensure accuracy. Automated translations are convenient but may not always provide the desired quality.

13. SEO for Multilingual Sites: - Implement hreflang tags to inform search engines about the language and regional targeting of your pages. This helps with SEO and ensures the right content is displayed to users based on their language preferences.

Razorpay integration

Donation button example

omains/programmingwithvishal.com/public_html/demo/razorpay/index.php

```
<button id="rzp-button1">Pay</button>
<script src="https://checkout.razorpay.com/v1/checkout.js"></script>
<script>
var options = {
    "key": "rzp_test_UY1y7bu0apmIK4", // Enter the Key ID generated from the Dashboard
    "amount": "50000", // Amount is in currency subunits. Default currency is INR. Hence, 50000 refers to 50000 paise
    "currency": "INR",
    "name": "Acme Corp",
    "description": "Test Transaction",
    "image": "https://example.com/your_logo",
    "handler": function (response){
        console.log(response);
    }
};
var rzp1 = new Razorpay(options);
document.getElementById('rzp-button1').onclick = function(e){
    rzp1.open();
    e.preventDefault();
}
</script>
```

Database Method Integration

index.php

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script src="https://checkout.razorpay.com/v1/checkout.js"></script>
<form>
    <input type="textbox" name="name" id="name" placeholder="Enter your
name"/><br/><br/>
    <input type="textbox" name="amt" id="amt" placeholder="Enter amt"/><br/><br/>
    <input type="button" name="btn" id="btn" value="Pay Now"
onclick="pay_now()"/>
```

```
</form>

<script>
    function pay_now(){
        var name=jQuery('#name').val();
        var amt=jQuery('#amt').val();

         jQuery.ajax({
               type:'post',
               url:'payment_process.php',
               data:"amt="+amt+"&name="+name,
               success:function(result){
                   var options = {
                        "key": "key",
                        "amount": amt*100,
                        "currency": "INR",
                        "name": "Acme Corp",
                        "description": "Test Transaction",
                        "image": "https://image.freepik.com/free-vector/logo-
sample-text_355-558.jpg",
                        "handler": function (response){
                           jQuery.ajax({
                               type:'post',
                               url:'payment_process.php',
                               data:"payment_id="+response.razorpay_payment_id,
                               success:function(result){
                                   window.location.href="thank_you.php";
                               }
                           });
                        }
                    };
                    var rzp1 = new Razorpay(options);
                    rzp1.open();
               }
           });


    }
</script>
```

payment_process.php

```php
<?php
session_start();
include('db.php');
if(isset($_POST['amt']) && isset($_POST['name'])){
    $amt=$_POST['amt'];
    $name=$_POST['name'];
    $payment_status="pending";
    $added_on=date('Y-m-d h:i:s');
    mysqli_query($con,"insert into payment(name,amount,payment_status,added_on)
values('$name','$amt','$payment_status','$added_on')");
    $_SESSION['OID']=mysqli_insert_id($con);
}


if(isset($_POST['payment_id']) && isset($_SESSION['OID'])){
    $payment_id=$_POST['payment_id'];
    mysqli_query($con,"update payment set
payment_status='complete',payment_id='$payment_id' where
id='".$_SESSION['OID']."'");
}
?>
```

db.php

```php
<?php
$con=mysqli_connect('host','username','password','dbname');
?>
```

thankyou.php

```
<h1>Payment Complete</h1>
```

Woocommerce Payment Gateway

Step 3: Create the Main Plugin File

Inside your plugin directory, create a main PHP file (e.g., `custom-payment-gateway.php`) and add the following code:

php Copy code

```php
<?php
/*
Plugin Name: Custom Payment Gateway
Description: Custom payment gateway integration for WooCommerce.
Version: 1.0
Author: Your Name
*/

// Add your custom payment gateway code here
?>
```

Regene

Step 4: Add Payment Gateway Class

Inside your main plugin file, create a class for your payment gateway. This class should extend the `WC_Payment_Gateway` class provided by WooCommerce. Here's a simplified example:

```php
class Custom_Payment_Gateway extends WC_Payment_Gateway {
    public function __construct() {
        $this->id = 'custom_payment';
        $this->icon = ''; // URL to an image for your gateway
        $this->method_title = 'Custom Payment Gateway';
        $this->method_description = 'Pay with Custom Payment Gateway';
        $this->has_fields = false;

        $this->init_settings();

        $this->title = $this->get_option('title');
```

```
        $this->description = $this->get_option('description');

        add_action('woocommerce_update_options_payment_gateways_' . $this->id, array($this, 'process_admin_options'));
    }

    public function init_form_fields() {
        $this->form_fields = array(
            'title' => array(
                'title' => 'Title',
                'type' => 'text',
                'description' => 'Title displayed to customers during checkout.',
                'default' => 'Custom Payment'
            ),
            'description' => array(
                'title' => 'Description',
                'type' => 'textarea',
                'description' => 'Description displayed to customers during checkout.',
                'default' => 'Pay securely with Custom Payment Gateway.'
            )
        );
    }

    public function process_payment($order_id) {
        // Handle payment processing here and return the result
    }
}

function add_custom_payment_gateway($methods) {
    $methods[] = 'Custom_Payment_Gateway';
    return $methods;
}

add_filter('woocommerce_payment_gateways', 'add_custom_payment_gateway');
```

In this code:

- We create a class `Custom_Payment_Gateway` that extends `WC_Payment_Gateway`. Customize the class properties and methods as needed.
- The `init_form_fields` method defines settings fields in WooCommerce admin.
- The `process_payment` method handles payment processing logic.
- `add_custom_payment_gateway` adds your custom gateway to WooCommerce.

Step 5: Activate the Plugin

Go to your WordPress admin dashboard, navigate to "Plugins," and activate your custom payment gateway plugin.

Step 6: Configure and Test

Costom plugin for payumoney

```
class Custom_PayUmoney_Gateway extends WC_Payment_Gateway {
    public function __construct() {
        $this->id = 'custom_payumoney';
        $this->icon = ''; // URL to an image for your gateway
        $this->method_title = 'Custom PayUmoney Gateway';
        $this->method_description = 'Pay with Custom PayUmoney Gateway';
        $this->has_fields = false;

        $this->init_settings();

        $this->title = $this->get_option('title');
        $this->description = $this->get_option('description');

        add_action('woocommerce_update_options_payment_gateways_' . $this-
>id, array($this, 'process_admin_options'));
    }

    public function init_form_fields() {
        $this->form_fields = array(
            'title' => array(
                'title' => 'Title',
```

```
                'type' => 'text',
                'description' => 'Title displayed to customers during
checkout.',
                'default' => 'Custom PayUmoney'
            ),
            'description' => array(
                'title' => 'Description',
                'type' => 'textarea',
                'description' => 'Description displayed to customers during
checkout.',
                'default' => 'Pay securely with Custom PayUmoney Gateway.'
            ),
            'merchant_key' => array(
                'title' => 'Merchant Key',
                'type' => 'text',
                'description' => 'Your PayUmoney merchant key.',
                'default' => ''
            ),
            'salt' => array(
                'title' => 'Salt',
                'type' => 'text',
                'description' => 'Your PayUmoney merchant salt.',
                'default' => ''
            ),
        );
    }

    public function process_payment($order_id) {
        // Handle payment processing using the PayUmoney API here
    }
}

function add_custom_payumoney_gateway($methods) {
    $methods[] = 'Custom_PayUmoney_Gateway';
    return $methods;
}

add_filter('woocommerce_payment_gateways', 'add_custom_payumoney_gateway');
```

Costom plugin for CCavenue

```
class Custom_CCAvenue_Gateway extends WC_Payment_Gateway {
    public function __construct() {
        $this->id = 'custom_ccavenue';
        $this->icon = ''; // URL to an image for your gateway
        $this->method_title = 'Custom CCAvenue Gateway';
        $this->method_description = 'Pay with Custom CCAvenue Gateway';
        $this->has_fields = false;

        $this->init_settings();

        $this->title = $this->get_option('title');
        $this->description = $this->get_option('description');

        add_action('woocommerce_update_options_payment_gateways_' . $this-
>id, array($this, 'process_admin_options'));
    }

    public function init_form_fields() {
        $this->form_fields = array(
            'title' => array(
                'title' => 'Title',
                'type' => 'text',
                'description' => 'Title displayed to customers during
checkout.',
                'default' => 'Custom CCAvenue'
            ),
            'description' => array(
                'title' => 'Description',
                'type' => 'textarea',
                'description' => 'Description displayed to customers during
checkout.',
                'default' => 'Pay securely with Custom CCAvenue Gateway.'
            ),
            'merchant_id' => array(
                'title' => 'Merchant ID',
                'type' => 'text',
                'description' => 'Your CCAvenue merchant ID.',
                'default' => ''
            ),
            'access_code' => array(
                'title' => 'Access Code',
                'type' => 'text',
                'description' => 'Your CCAvenue access code.',
                'default' => ''
```

```
            ),
            'working_key' => array(
                'title' => 'Working Key',
                'type' => 'text',
                'description' => 'Your CCAvenue working key.',
                'default' => ''
            ),
        );
    }

    public function process_payment($order_id) {
        // Handle payment processing using the CCAvenue API here
    }
}

function add_custom_ccavenue_gateway($methods) {
    $methods[] = 'Custom_CCAvenue_Gateway';
    return $methods;
}

add_filter('woocommerce_payment_gateways', 'add_custom_ccavenue_gateway');
```