

COMMENT TAGS	POST TAGS	NAVIGATION MENU TAGS
get_avatar(); comment_ID(); comment_text(); comment_type(); 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();	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(); 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();	wp_nav_menu(); wp_get_nav_menu(); walk_nav_menu_tree(); wp_get_nav_menu_item();
SHORTCUT FUNCTIONS	CONDITIONAL TAGS	PERMALINKS TAGS
do_shortcode(); add_shortcode(); shortcode_atts(); strip_shortcodes(); remove_shortcode(); remove_all_shortcodes();	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(); ping_open(); in_category(); is_trackback(); is_front_page(); is_attachment(); comments_open(); is_page_template(); is_comments_popup();	the_permalink(); get_permalink(); post_permalink(); permalink_anchor(); get_post_permalink(); permalink_single_rss();
EDIT LINK TAGS	ATTACHEMENT TAGS	AUTHOR TAGS
edit_tag_link(); edit_post_link(); edit_comment_link(); edit_bookmark_link();	is_attachment(); the_attachment_link(); wp_attachment_is_image(); wp_get_attachment_image(); wp_get_attachment_metadata(); wp_get_attachment_image_src();	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();
LIST & DROPODOWN TAGS	CATEGORY TAGS	PLUGIN API FUNCTIONS
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();	the_category(); single_cat_title(); the_category_rss(); wp_list_categories(); category_description(); wp_dropdown_categories();	has_filter(); add_filter(); apply_filters(); current_filter(); merge_filters(); remove_filter(); remove_all_filters();
LOGIN/LOGOUT TAGS	DATE & TIME TAGS	POST THUMBNAIL 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();	the_time(); the_date(); get_calendar(); get_the_date(); the_date_xml(); the_modified_date(); the_modified_time(); single_month_title(); the_modified_author();	has_post_thumbnail(); the_post_thumbnail(); get_post_thumbnail_id(); get_the_post_thumbnail();
LINKS MANAGER TAGS	TITLE TAGS	QUERY TAGS
get_bookmark(); get_bookmarks(); wp_list_bookmarks(); get_bookmark_field();	wp_title(); single_cat_title(); single_tag_title(); single_post_title(); the_search_query(); single_month_title();	get_posts(); query_posts(); rewind_posts(); wp_reset_query();
SHORTLINK TAGS	INCLUDE TAGS	INCLUDE TAGS
the_shortlink(); wp_get_shortlink(); wp_shortlink_header(); wp_shortlink_wp_head();	get_footer(); get_header(); get_sidebar(); get_search_form(); get_template_part(); comments_template();	get_footer(); get_header(); get_sidebar(); get_search_form(); get_template_part(); comments_template();
TITLE TAGS	CATEGORY TAGS	TAG TAGS
wp_title(); single_cat_title(); single_tag_title(); single_post_title(); the_search_query(); single_month_title();	the_category(); single_cat_title(); the_category_rss(); wp_list_categories(); category_description(); wp_dropdown_categories();	the_tags(); wp_tag_cloud(); tag_description(); single_tag_title(); wp_generate_tag_cloud();
DATE & TIME TAGS	BLOG INFO TAGS	BLOG INFO 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();	bloginfo(); bloginfo_rss(); get_bloginfo(); get_bloginfo_rss();	bloginfo(); bloginfo_rss(); get_bloginfo(); get_bloginfo_rss();

TEMPLATES: THE BASIC FILES

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

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');?>	Charset 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 cage 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_meta();?>	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

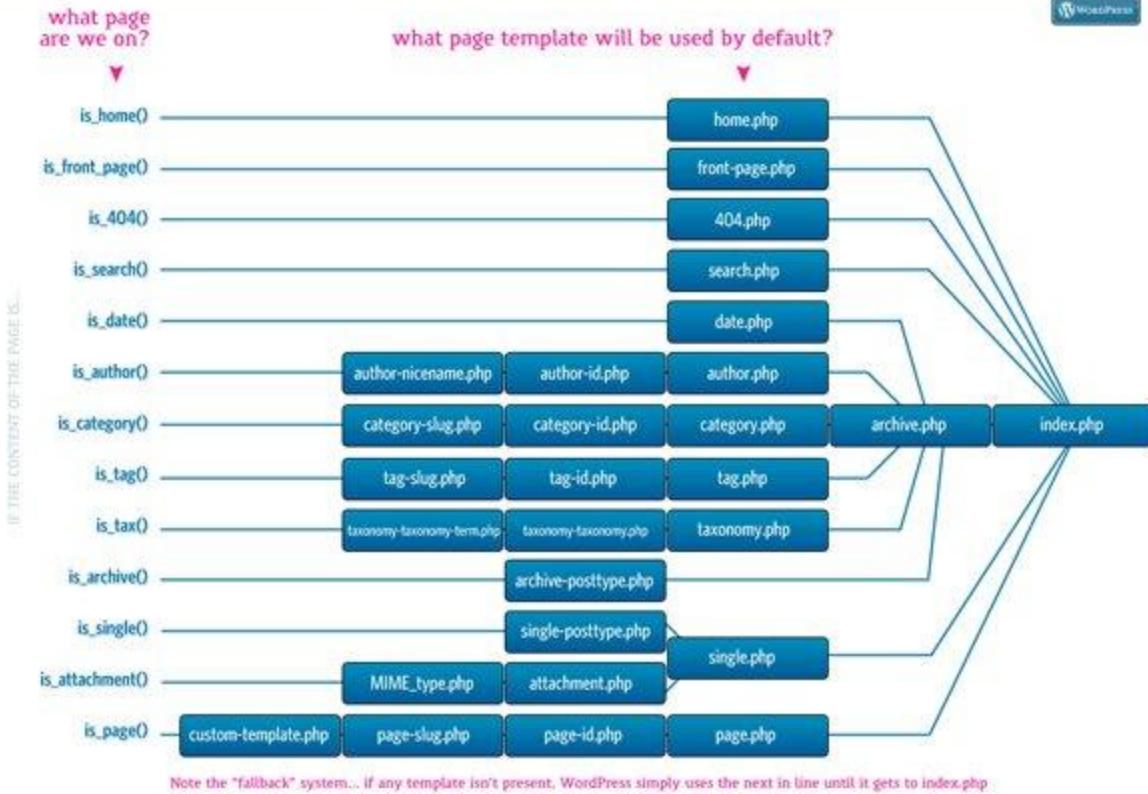
```
<ul id="menu">
<li <?php if(is_home()) { ?> class="current-cat" <?php } ?>>
<a href="php bloginfo(&amp;utmx=g_organic'home'); ?&gt;">Home</a></li>
<?php wp_list_categories('title_li=&orderby=id');?>
</ul>
```

Page Based Navigation

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

TEMPLATES: EXTRA FUNCTIONS

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



Paste this code in function.php file

```
function wc_prevent_clickjacking() {
    header( 'X-FRAME-OPTIONS: SAMEORIGIN' );
}

add_action( 'send_headers', 'wc_prevent_clickjacking', 10 );
```

WordPress Security

1. Checking User Capabilities
2. Data Validation
3. Securing (sanitizing) Input
4. Securing (escaping) Output
5. Nonces
6. XML-RPC issue
7. Clickjacking
8. Directory listing issue
9. Internal path disclosure
10. Server signature
11. SQL Injection
12. WordPress version
13. Login URL exposed
14. Robots.txt Disclosure
15. Limit Login Attempts:
16. Implement Two-Factor Authentication (2FA):
17. Use SSL/TLS Encryption:
18. Keep WordPress Updated:
19. Use Strong Passwords
20. Secure Hosting Environment
21. Regular Backups:
22. Use Trusted Themes and Plugins:

"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

Caching : Page, Browser, Object, Data Cache

Indexing & Optimize Database:

Limit and Optimize Plugins:

Use a Lightweight and Optimized Theme:

Conditionally enqueue JS/CSS

Minifying JS/CSS/HTML

Gzip File Compression

Optimize Core Web Vitals (CLS, LCP and FCP)

CDN Setup and Fix Insecure Content

Lazy Loading Images and iFrame

Cache Preloading

Fonts Preloading

Font Optimization

Deferring Unused JS/CSS

Inline & Combine JS/CSS

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.

Indexing in Database

1. **Fast Data Retrieval:** Indexes speed up data retrieval operations, such as SELECT queries, by reducing the number of rows that need to be scanned in a table. Instead of scanning the entire table, the database engine uses the index to locate the relevant rows more efficiently.
2. **Structure:** An index is a separate data structure that contains a copy of selected columns from a table, along with a reference to the corresponding rows in the table. This structure allows for quick lookups.
3. **Columns Selection:** You can create indexes on one or more columns of a table. Commonly indexed columns include primary keys, foreign keys, frequently searched columns, and columns used in JOIN and WHERE clauses.
4. **Types of Indexes:** Various types of indexes exist, including primary indexes, secondary indexes, clustered indexes, non-clustered indexes, and full-text indexes. Each type has specific use cases and characteristics.
 - **Primary Index:** A primary index uniquely identifies each row in a table and enforces data integrity. There can be only one primary index per table.
 - **Secondary Index:** A secondary index is created on columns other than the primary key. It helps optimize search queries.
 - **Clustered Index:** In some database systems (e.g., SQL Server), a table can have only one clustered index, which determines the physical order of rows in the table. It is usually created on the primary key or another unique column.
 - **Non-Clustered Index:** A non-clustered index provides an alternative path to access data rows and includes a pointer to the actual data row.
5. **Trade-offs:** While indexing can significantly improve read performance, it comes with trade-offs. Indexes consume storage space and can slow down write operations (INSERT, UPDATE, DELETE) because each modification must also update the corresponding index entries. Therefore, it's important to strike a balance between read and write performance.
6. **Maintenance:** Indexes require regular maintenance to keep them efficient. Database systems may automatically update and reorganize indexes, but it's essential to monitor index performance and occasionally rebuild or defragment them if needed.
7. **Query Optimization:** The query optimizer in a database management system uses indexes to determine the most efficient execution plan for queries. Properly designed indexes can lead to faster query execution.

In summary, indexing is a crucial database optimization technique that improves the speed and efficiency of data retrieval operations while carefully considering the trade-offs associated with storage and write performance. Effective indexing is a fundamental aspect of database design and query performance tuning.

Indexing Procedure

Indexing in MySQL, when combined with PHP, can significantly improve the performance of your database-driven web applications. Here are the steps to create and utilize indexes effectively with MySQL and PHP:

Step 1: Design Your Database Schema

The first step in effective indexing is to design your database schema correctly. Identify the columns that are frequently used in WHERE clauses, JOIN operations, and ORDER BY clauses in your SQL queries. These are the columns that should be considered for indexing.

Step 2: Choose the Right Index Types

MySQL supports various index types, including PRIMARY KEY, UNIQUE, INDEX, and FULLTEXT. Choose the appropriate index type based on your data and query requirements. Common choices include:

- **PRIMARY KEY:** Used for uniquely identifying rows in a table.
- **INDEX:** Used for non-unique indexing of columns.
- **UNIQUE:** Ensures that all values in the indexed column(s) are unique.
- **FULLTEXT:** Used for full-text searches in text-based columns.

Step 3: Create Indexes

You can create indexes using SQL statements in MySQL. For example, to create an index on the `email` column of a `users` table:

```
CREATE INDEX idx_email ON users (email);
```

Alternatively, you can define indexes when creating tables:

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(255) UNIQUE,
    email VARCHAR(255) INDEX,
    -- other columns...
);
```

Step 4: Monitor Query Performance

Once indexes are created, monitor the performance of your SQL queries using the MySQL query analyzer or other performance profiling tools. Identify queries that benefit from indexes and those that need optimization.

Step 5: Utilize EXPLAIN

Use the `EXPLAIN` statement to analyze query execution plans. For example:

```
EXPLAIN SELECT * FROM users WHERE email = 'user@example.com';
```

`EXPLAIN` will provide insight into how MySQL is executing the query and whether it's using indexes efficiently.

Step 6: Adjust Indexes

Based on query performance analysis, you may need to adjust your indexes. You might need to add additional indexes, remove redundant indexes, or modify existing indexes.

Step 7: Regularly Optimize Indexes

Periodically check the fragmentation and efficiency of your indexes. MySQL provides tools like `OPTIMIZE TABLE` and `ANALYZE TABLE` to maintain and optimize your indexes.

Step 8: Leverage PHP for Query Optimization

While most of the index creation and maintenance tasks are performed in MySQL, PHP can also play a role in query optimization. Ensure your PHP code uses parameterized queries to prevent SQL injection and leverages caching mechanisms for frequently used query results.

Step 9: Test at Scale

Finally, test your application and database performance at scale to ensure that the indexing strategy remains effective as your data grows and user traffic increases.

Remember that indexing is not a one-size-fits-all solution. The effectiveness of indexes depends on your specific application and query patterns. Regular monitoring and adjustments are essential for maintaining optimal performance.

Caching

Caching in PHP refers to the process of storing and retrieving data temporarily to reduce the workload on a web server, improve performance, and decrease load times for web pages. Caching can be applied to various types of data, such as database query results, web pages, or even entire objects.

There are different types of caching techniques in PHP, including:

1. **Opcode Caching:** PHP code is first translated into an intermediate code called opcode, and opcode caching stores this intermediate code in memory, reducing the need to recompile the PHP code on each request.
2. **Data Caching:** This involves storing the results of expensive database queries or computations so that they can be quickly retrieved without repeating the same calculations.
3. **Page Caching:** Entire HTML pages can be cached so that they don't need to be regenerated for each user request. This is particularly effective for static or semi-static content.
4. **Object Caching:** In object-oriented programming, you can cache entire objects or the results of method calls to avoid re-computing them.

Popular PHP caching mechanisms include using tools like APC (Alternative PHP Cache), OPcache, Memcached, or Redis.

Caching in a server context refers to the process of storing and reusing previously generated data or responses to reduce the time and resources required to fulfill subsequent requests. Caching is a critical technique for improving the performance and responsiveness of web servers and applications. There are several types of caching that can be implemented on a server:

1. **Page Caching:** Page caching stores entire HTML pages as static files. When a user requests a page that has been cached, the server can serve the cached page directly without processing PHP scripts or querying a database. Popular tools like Varnish and WP Super Cache for WordPress offer page caching capabilities.
2. **Object Caching:** Object caching stores frequently used data objects, such as database query results or API responses, in memory. This reduces the need to repeatedly fetch the same data from slow data sources. Memcached and Redis are commonly used object caching solutions.
3. **Opcode Caching:** Opcode caching stores compiled PHP code in memory so that it can be reused for subsequent PHP script executions. This significantly speeds up the execution of PHP scripts. Popular opcode caching tools include APCu, OPcache (built-in to PHP), and XCache.
4. **CDN (Content Delivery Network) Caching:** CDNs cache static assets like images, stylesheets, and JavaScript files on geographically distributed servers. This reduces the latency and load on your web server and speeds up the delivery of content to users.
5. **Database Query Caching:** Database query caching stores the results of database queries in memory, reducing the need to re-execute the same queries. Many database systems, such as MySQL and PostgreSQL, have built-in query caching features.
6. **Reverse Proxy Caching:** A reverse proxy server like Nginx or Apache can cache responses from your application server. This is often used in combination with load balancing and SSL termination to improve performance and security.
7. **Browser Caching:** In addition to server-side caching, instructing web browsers to cache static assets (e.g., images, CSS, JavaScript) can reduce the need for repeated downloads when users revisit your site.

Implementing caching effectively requires a good understanding of your application's architecture and performance bottlenecks. You'll need to decide what should be cached, how long data should be cached, and which caching technologies are most appropriate for your specific use case. Caching can significantly improve the speed and efficiency of your server but should be used judiciously to avoid serving outdated or stale data to users.

Enabling caching on a server can significantly improve the performance and responsiveness of websites and web applications. Caching reduces the need to regenerate or fetch data for each user request by storing and serving previously generated content. There are several ways to enable caching on a server, and the specific method you choose depends on your server setup and requirements. Here are some common caching techniques and how to enable them:

1. Browser Caching:

- Browser caching instructs web browsers to store static assets (such as images, stylesheets, and JavaScript files) locally on the user's device. This reduces the need to re-download these assets on subsequent visits.
- To enable browser caching, add cache-control headers to your web server's configuration or within your application code. For Apache, you can use `.htaccess` files, and for Nginx, you can configure it in the server block.

Example (Apache):

```
<FilesMatch "\.(jpg|jpeg|png|gif|js|css)$">
  Header set Cache-Control "max-age=31536000, public"
</FilesMatch>
```

Opcode Caching (PHP):

- Opcode caching stores compiled PHP code in memory to avoid recompilation for each request. This can significantly improve the execution speed of PHP scripts.
- Popular PHP opcode caching tools include APCu, OPcache (included in PHP by default), and XCache. To enable opcode caching, you typically need to install the appropriate extension and configure it in your `php.ini` file.

Example (php.ini for OPcache):

```
zend_extension=opcache.so
```

```
opcache.enable=1
```

1. Object Caching (PHP and Database):

- Object caching stores frequently used data objects, such as database query results or API responses, in memory. This reduces the need to repeatedly fetch the same data from a slow data source.

- You can use caching libraries like Memcached or Redis for object caching in PHP. Additionally, some database systems have built-in query caching mechanisms.

2. Content Delivery Network (CDN) Caching:

- CDNs cache static assets and deliver them from geographically distributed servers. This reduces latency and load on your web server.
- To enable CDN caching, sign up for a CDN service, configure your CDN settings, and update your DNS records to point to the CDN.

3. Page Caching (WordPress and CMS):

- Content management systems (CMS) like WordPress often have built-in page caching plugins. Enable and configure these plugins to cache full HTML pages.
- For WordPress, popular caching plugins include W3 Total Cache and WP Super Cache.

4. Reverse Proxy Caching (Nginx and Varnish):

- Reverse proxy servers like Nginx and Varnish can cache responses from your application server. This is often used in combination with load balancing and SSL termination.
- Configure reverse proxy caching in your server configuration files.

Remember to test your caching setup thoroughly to ensure that it improves performance without causing issues like serving stale content to users. Additionally, monitor cache performance and adjust settings as needed, especially as your website or application evolves.

Advantages and disadvantages of caching

Caching is a technique used in computing and web development to store and retrieve frequently used data or processes more quickly than if they were generated or calculated each time. Here are the advantages and disadvantages of caching:

Advantages of Caching:

1. Improved Performance:

- **Faster Load Times:** Caching reduces the time it takes to load content or execute processes, leading to faster response times and improved user experience.
- **Reduced Server Load:** By serving cached content, the server's load is reduced, allowing it to handle more concurrent users.

2. Bandwidth Savings:

- **Reduced Data Transfer:** Caching can reduce the amount of data transferred between the server and the client, resulting in lower bandwidth usage and cost savings, especially for content-heavy websites.

3. Server Resource Optimization:

- **Lower CPU Usage:** Caching reduces the need for the server to regenerate content or perform resource-intensive computations, leading to lower CPU usage and improved server efficiency.

4. Scalability:

- **Improved Scalability:** Caching can contribute to better scalability, as cached content can be served to a large number of users without putting excessive strain on the server.

5. Better User Experience:

- **Consistent Experience:** Users experience faster page loads and more responsive applications, contributing to an overall positive user experience.

6. Load Balancing:

- **Balancing Traffic:** Caching helps distribute traffic more evenly across servers in a load-balanced environment, preventing individual servers from becoming bottlenecks.

7. Offline Accessibility:

- **Offline Support:** Caching enables certain content or features to be accessible offline, enhancing user convenience in scenarios where internet connectivity is intermittent.

Disadvantages of Caching:

1. Stale Content:

- **Outdated Information:** If not managed properly, cached content may become outdated. This can lead to users seeing stale information if the cache is not refreshed regularly.

2. Complex Cache Invalidation:

- **Difficulty in Invalidation:** In some cases, it can be challenging to invalidate or update cached content when the underlying data changes. This requires careful management to ensure users receive the latest information.

3. Storage Concerns:

- **Disk Space Usage:** Depending on the caching method used, cached data may consume significant disk space. This can be a concern for environments with limited storage.

4. Implementation Complexity:

- **Complex Setup:** Implementing and managing caching can be complex, especially for dynamic or frequently changing content. Developers need to carefully design and configure caching mechanisms.

5. Security Risks:

- **Potential Security Issues:** Caching may introduce security risks if sensitive or private information is cached and not properly protected. Developers need to consider security implications when implementing caching.

6. Overhead for Small Applications:

- **Overhead for Small Apps:** For small-scale applications or websites with minimal traffic, the overhead of implementing and managing caching may outweigh the performance benefits.

Transient

Transients are a valuable mechanism in WordPress for caching and temporarily storing data, particularly for enhancing performance by reducing the need to make repeated and resource-intensive database queries. They serve as a way to store and retrieve data that may change infrequently, and they automatically expire after a specified period. Transients are especially useful when you need to cache data that is expensive to compute or retrieve.

Here's an overview of the role of transients in WordPress and how they differ from options:

Transients:

1. **Temporary Data Storage:** Transients are designed for storing data that is temporary or can be regenerated when needed. This data can include API responses, query results, or any data that doesn't need to be stored indefinitely.
2. **Expiration Time:** Transients have an expiration time, which you specify when setting the transient. Once the expiration time has passed, the data is considered invalid, and the transient is deleted automatically during the next access.
3. **Usage Scenario:** Use transients when you need to cache data temporarily and refresh it periodically or when certain conditions change. For example, you might use a transient to store the results of an expensive database query for a short period.
4. **Example Usage:** Here's how you can set and retrieve a transient:

Options:

1. **Permanent Data Storage:** Options, on the other hand, are intended for storing data that should persist across sessions and page loads. Options store data in the WordPress database, making it available for the long term.
2. **No Expiration:** Options do not have an expiration time, so the data remains stored until explicitly deleted or updated.
3. **Usage Scenario:** Use options for storing settings, configurations, or data that should remain consistent throughout the lifetime of your WordPress installation or until you deliberately change or delete them.
4. **Example Usage:** Here's how you can set and retrieve an option:

When to Use One Over the Other:

The choice between transients and options depends on your specific use case:

1. Use Transients When:

- You need to cache data temporarily.
- You want the data to expire and be automatically refreshed.
- You are dealing with data that is resource-intensive to generate or retrieve.
- You want to reduce the load on your database for frequently accessed data that can be cached for a short duration.

2. Use Options When:

- You need to store data permanently or for an extended period.
- You want the data to persist across sessions, page loads, and even WordPress updates.
- You are storing settings, configurations, or data that should remain consistent.

In WordPress, a "transient" refers to a type of caching mechanism that allows developers to store and retrieve temporary data in an efficient way. Transients are typically used to store results of expensive database queries, API requests, or other time-consuming operations.

The purpose of using transients is to improve the performance of a WordPress site by reducing the need to regenerate or fetch certain data on every page load. Instead, the data is stored temporarily in the site's database or in the server's memory, making it quicker to retrieve when needed.

Here's a basic overview of how transients work in WordPress:

1. Set a Transient:

To store data as a transient, you use the `set_transient` function. This function takes a name (a unique identifier for the transient), the data you want to store, and a expiration time (in seconds) after which the transient will expire and be automatically deleted.

Example:

```
php Copy code
      set_transient('example_transient', $data, 3600); // This stores $data for 1 hour (3600 seconds)
```

2. Get a Transient:

To retrieve the stored data, you use the `get_transient` function. If the transient has expired or doesn't exist, the function will return `false`.

Example:

```
php Copy code
      $cached_data = get_transient('example_transient');
      if (false === $cached_data) {
          // Transient doesn't exist or has expired, regenerate the data and set a new transient
          $cached_data = expensive_operation_to_generate_data();
          set_transient('example_transient', $cached_data, 3600);
      }
```

3. Delete a Transient:

You can manually delete a transient using the `delete_transient` function.

Example:

```
php Copy code
      delete_transient('example_transient');
```

XSS, or Cross-Site Scripting

XSS, or Cross-Site Scripting, is a security vulnerability that occurs when an attacker injects malicious scripts (usually written in JavaScript) into web applications. These scripts then get executed in the context of a victim's browser, allowing the attacker to steal information, hijack sessions, deface websites, and perform other malicious actions. XSS attacks are particularly dangerous because they can compromise the confidentiality and integrity of web applications and their users.

There are three main types of XSS attacks:

1. **Stored XSS:** The injected script is permanently stored on the target server and is served to users when they visit a particular page.
2. **Reflected XSS:** The injected script is reflected off a web server, typically by tricking a user into clicking a malicious link that contains the script.
3. **DOM-based XSS:** The attack occurs in the Document Object Model (DOM) of a web page, often through client-side JavaScript manipulation.

To prevent XSS attacks in PHP (as well as in any web development), follow these best practices:

1. Input Validation and Sanitization:

- Always validate and sanitize user inputs, both on the client and server sides.
- Use PHP functions like `htmlspecialchars()` to escape user-generated content before rendering it in HTML templates.

2. Use Content Security Policy (CSP):

- Implement a Content Security Policy that specifies which sources of content are allowed to be executed on a web page. This can help mitigate the impact of XSS attacks.

3. HTTP-Only Cookies:

- Mark cookies as "HTTP-only" to prevent JavaScript from accessing them. This helps protect sensitive session data.

4. Session Management:

- Use secure session management practices, including secure session cookies and session timeouts.

5. Use Frameworks and Libraries:

- Utilize well-established PHP frameworks and libraries that have built-in security features for handling user input and output, such as Laravel, Symfony, or CodeIgniter.

6. Content Escaping:

- When echoing dynamic data into HTML, JavaScript, or other contexts, use context-specific escaping functions (e.g., `htmlspecialchars()` for HTML, `json_encode()` for JavaScript) to prevent malicious code execution.

7. Validation of URL and File Uploads:

- Validate and sanitize URLs before rendering them on the page.
- If your application allows file uploads, ensure that files are validated, sanitized, and stored securely.

8. Use Security Headers:

- Implement security headers, such as HTTP Strict Transport Security (HSTS), to enhance security and protect against various types of attacks, including XSS.

9. Regular Security Audits:

- Conduct regular security audits and penetration testing to identify and fix vulnerabilities.

10. Stay Informed:

- Keep up-to-date with the latest security threats, patches, and best practices in web development and PHP.

It's crucial to follow these practices consistently throughout your PHP application's development and maintenance cycle to reduce the risk of XSS vulnerabilities. Security should be an ongoing concern, and proactive measures are essential to protect your application and its users from malicious attacks.

XSS, or Cross-Site Scripting in wordpress

To prevent XSS vulnerabilities in WordPress:

1. **Update WordPress Regularly:** Ensure that you're running the latest version of WordPress, as updates often include security fixes.
2. **Use Security Plugins:** Install security plugins like Wordfence or Sucuri Security, which can help detect and prevent XSS attacks.
3. **User Input Sanitization:** Validate and sanitize all user inputs, including comments and form submissions, before displaying them on your website. WordPress provides functions like `esc_html()`, `esc_attr()`, and `esc_js()` for this purpose.
4. **Content Security Policy (CSP):** Implement a Content Security Policy header to restrict the sources from which scripts can be loaded. This can help prevent unauthorized script execution.

5. **Use Frameworks and Libraries:** If you're developing custom themes or plugins, use established WordPress coding standards and frameworks that include built-in security features.
6. **Session Management:** Secure your session management to prevent session fixation attacks. You can use plugins or implement secure session handling in your custom code.
7. **Input Validation:** When accepting file uploads or user-generated content like URLs, implement strict validation and consider using a library like the WordPress HTTP API to make requests safely.
8. **Escape Output:** Use functions like `esc_html()`, `esc_url()`, and `esc_js()` to escape output data to ensure it's properly sanitized before being displayed.
9. **Educate Users:** If your WordPress site allows user-generated content, educate users about the importance of not including script or malicious code in their submissions.
10. **Regular Security Audits:** Periodically conduct security audits and vulnerability scanning to identify and address any potential XSS vulnerabilities.

CSRF stands for Cross-Site Request Forgery

CSRF stands for Cross-Site Request Forgery. It is a type of security vulnerability that occurs when an attacker tricks a user's browser into making an unintended and unauthorized request on a website where the user is authenticated. CSRF attacks exploit the trust that a website has in a user's browser.

Here's a basic scenario to illustrate how CSRF attacks work:

1. **User Authentication:** A user logs into a secure website, such as a banking site or a social media platform, and receives a session cookie to maintain their authenticated state.
2. **Malicious Website:** The same user visits a malicious website (controlled by the attacker) that contains hidden code or scripts.
3. **Unintended Request:** The malicious website, without the user's knowledge, triggers a request to the secure website where the user is authenticated. This request could be anything from changing the user's password to initiating a financial transaction.
4. **Authentication Bypass:** Since the user is already authenticated on the target website, the browser includes the necessary authentication credentials (like session cookies) with the request, making it appear legitimate to the website.
5. **Unauthorized Action:** The target website processes the request, thinking it's a legitimate action initiated by the authenticated user, and performs the unintended action.

To mitigate CSRF attacks, web developers often use anti-CSRF measures such as:

1. **CSRF Tokens:** Including a unique token in each form or request that is tied to the user's session. This token is checked on the server side to verify that the request is legitimate.
2. **SameSite Cookies:** Configuring cookies to be sent only in a first-party context, preventing them from being sent in a cross-site request.
3. **Referrer Policy:** Checking the HTTP Referer header to ensure that requests originate from the same site.
4. **Double-Submit Cookies:** Including the same token in both a cookie and as a request parameter, making it more difficult for an attacker to forge a valid request.

WCAG (Web Content Accessibility Guidelines),

Web accessibility, often referred to as WCAG (Web Content Accessibility Guidelines), is a critical aspect of WordPress development. It involves designing and developing websites and web applications in a way that ensures equal access and usability for all users, regardless of their physical or cognitive abilities. Here's why web accessibility is of paramount importance in WordPress development:

1. Inclusivity and Equal Access:

- Web accessibility ensures that everyone, including individuals with disabilities, can access and use your WordPress website. It promotes inclusivity by removing barriers and discrimination against users with disabilities.

2. Legal and Regulatory Compliance:

- Many countries have adopted accessibility regulations, and non-compliance can result in legal consequences, including lawsuits and fines. Adhering to WCAG guidelines helps WordPress developers meet these legal requirements.

3. Ethical Responsibility:

- Promoting web accessibility is an ethical responsibility for web developers. Ensuring that your website is accessible demonstrates a commitment to providing a positive and equitable user experience for all.

4. Expanding Audience and Reach:

- An accessible website opens doors to a broader audience, including people with disabilities, elderly users, and those using assistive technologies. This expanded reach can lead to increased traffic and engagement.

5. Improved SEO and Search Rankings:

- Many accessibility practices align with search engine optimization (SEO) best practices. By making your site more accessible, you can improve your SEO, resulting in better search rankings and increased organic traffic.

6. Enhancing User Experience:

- Web accessibility not only benefits users with disabilities but also improves the overall user experience for everyone. Designing with accessibility in mind often leads to cleaner, more structured, and user-friendly websites.

7. Future-Proofing Your Website:

- By building an accessible WordPress website, you future-proof your site against evolving accessibility regulations and standards. This reduces the need for extensive retroactive changes down the line.

8. Positive Brand Image:

- A commitment to accessibility reflects positively on your brand. It demonstrates a commitment to inclusivity and a customer-centric approach, which can build trust and brand loyalty.

9. Assistive Technologies Compatibility:

- Accessible websites are compatible with a wide range of assistive technologies, such as screen readers, voice recognition software, and braille displays, allowing users to navigate and interact with your content effectively.

10. Mitigating Litigation and Reputation Risks: - Legal actions related to web accessibility are on the rise. Ensuring your WordPress site is accessible helps mitigate the risk of legal challenges and preserves your organization's reputation.

11. Government and Educational Institutions: - Many government and educational institutions have strict accessibility requirements. Building accessible WordPress sites can help you win contracts and partnerships with these organizations.

In WordPress development, achieving web accessibility involves adhering to WCAG guidelines, which provide a comprehensive set of standards for creating accessible content and interfaces. Common accessibility considerations in WordPress development include providing alternative text for images, ensuring keyboard navigation, offering text alternatives for non-text content (e.g., videos and images), and semantic HTML markup.

WordPress itself has made efforts to enhance accessibility, including creating an accessible admin interface and encouraging theme and plugin developers to follow accessibility best practices. However, it remains the responsibility of individual WordPress developers to ensure their themes and plugins meet accessibility standards.

How can tools like Composer be used to manage dependencies effectively?

Composer is a powerful dependency management tool for PHP that allows you to declare, install, and manage external libraries, packages, and dependencies for your PHP projects. Here's how you can use Composer to manage dependencies effectively:

1. Install Composer:

- Before you can use Composer, you need to install it on your system. You can download and install Composer globally using the instructions provided on the [Composer website](#).

2. Create a `composer.json` File:

- In your PHP project's root directory, create a `composer.json` file. This file is used to declare your project's dependencies and configuration settings.

3. Declare Dependencies:

- Inside the `composer.json` file, declare the dependencies your project requires. You can specify dependencies by their package names and versions. For example:

jsonCopy code

```
"require": { "monolog/monolog": "^2.0", "symfony/http-foundation": "^5.0" }
```

- You can also specify other information like minimum stability, autoload settings, and more in the `composer.json` file.

4. Run `composer install`:

- Open your terminal or command prompt, navigate to your project's directory, and run `composer install`. Composer will read the `composer.json` file, download the specified dependencies, and install them in the `vendor` directory.

5. Autoload Configuration:

- Composer generates an autoloader for your project, allowing you to autoload classes from the installed packages. Include the autoloader in your PHP scripts:

phpCopy code

```
require_once 'vendor/autoload.php'
```

6. Update Dependencies:

- If you need to update your project's dependencies, modify the `composer.json` file with the new version constraints or package names and run `composer update`. Composer will update the packages accordingly.

7. Lock File:

- Composer generates a `composer.lock` file to lock the versions of your project's dependencies. This ensures that every developer working on the project uses the same package versions, enhancing consistency across environments.

8. Commit `composer.lock`:

- It's essential to commit the `composer.lock` file to your version control system (e.g., Git) so that other team members can install the exact same package versions as specified in the lock file.

9. Use Custom Repositories (Optional):

- Composer allows you to use custom package repositories if your project requires packages that are not available on the default package repositories (Packagist). You can specify custom repositories in the `composer.json` file.

10. Version Constraints:

- When declaring dependencies, you can specify version constraints using operators like `^`, `~`, `>=`, `<=`, etc., to control which versions of a package are acceptable. This helps ensure compatibility and allows for automatic updates within defined constraints.

11. Security and Vulnerability Scanning:

- Consider using Composer security tools and plugins like `sensiolabs/security-checker` or integrated services like Snyk to scan your dependencies for security vulnerabilities regularly.

12. Optimize Autoloading:

- Use Composer's autoloader optimization (`composer dump-autoload -o`) to generate a more efficient autoloader for your project, especially if you have a large number of classes.

what is wp engin in wordpress

WP Engine is a managed WordPress hosting provider. It offers hosting services specifically tailored for WordPress websites, providing a platform that is optimized for performance, security, and scalability. Here are some key features and aspects of WP Engine:

- Managed Hosting:** WP Engine is a fully managed hosting service, meaning they take care of various technical aspects of hosting, such as server maintenance, security, updates, and backups. This allows website owners to focus more on their content and business, rather than managing server infrastructure.
- Performance Optimization:** WP Engine is known for its emphasis on performance. They use technologies like EverCache, a proprietary caching system, to ensure fast page load times and optimal performance for WordPress sites.
- Security Measures:** Security is a top priority for WP Engine. They implement a range of security features and best practices to protect websites from common threats, including regular security scans, firewalls, and proactive monitoring.
- Scalability:** WP Engine provides hosting plans that can scale with the growth of a website. This scalability is important for websites experiencing increased traffic or resource demands.
- Staging Environments:** WP Engine offers staging environments, allowing users to create a clone of their website for testing and development purposes. This helps users test changes or new features before deploying them to the live site.
- Support:** WP Engine provides customer support, and their support teams are knowledgeable about WordPress. Users can reach out for assistance with hosting-related issues or inquiries.
- WordPress-Specific Features:** As a hosting provider focused on WordPress, WP Engine often includes features that are specifically designed for WordPress websites, such as compatibility with popular WordPress plugins and tools.

WP Engine is a managed WordPress hosting provider, and like any service, it comes with its own set of advantages and disadvantages. Here's an overview:

Advantages of WP Engine:

- 1. Managed Hosting:**
 - WP Engine is a fully managed WordPress hosting service, meaning they handle server maintenance, security, updates, and other technical aspects. This allows users to focus on content and business rather than server management.
- 2. Performance Optimization:**
 - WP Engine is known for its emphasis on performance. It uses technologies like EverCache for caching to ensure fast page load times, contributing to an improved user experience.
- 3. Security Features:**
 - WP Engine places a high priority on security. It includes features like firewalls, regular security scans, and proactive monitoring to protect websites from common threats.
- 4. Scalability:**
 - WP Engine provides scalable hosting plans to accommodate the growth of a website. This scalability is crucial for websites experiencing increased traffic or resource demands.
- 5. Staging Environments:**
 - WP Engine offers staging environments, allowing users to create clones of their websites for testing and development. This helps users test changes before deploying them to the live site.
- 6. WordPress Expertise:**
 - The support team at WP Engine is knowledgeable about WordPress. Users can seek assistance with hosting-related issues or inquiries related specifically to the WordPress platform.
- 7. Content Delivery Network (CDN):**
 - WP Engine includes a built-in CDN (Content Delivery Network) in its plans, helping to distribute website content across servers globally, reducing latency and improving load times.

Disadvantages of WP Engine:

- 1. Cost:**
 - WP Engine is a premium hosting service, and its plans are typically priced at a higher tier compared to basic shared hosting providers. This can be a disadvantage for budget-conscious users or small websites.
- 2. Limited Flexibility:**
 - While the managed nature of WP Engine is a benefit for many users, it can be a drawback for those who prefer more control over server configurations. Certain advanced configurations may be restricted.
- 3. Plugin Restrictions:**
 - WP Engine may restrict or disallow certain plugins that could affect server performance or security. Users need to be aware of the platform's plugin policies.
- 4. Data Center Locations:**
 - The availability of data center locations may be limited compared to some other hosting providers, which could affect website performance for users in certain geographic regions.
- 5. Learning Curve:**
 - Users who are new to managed WordPress hosting might experience a learning curve when transitioning to WP Engine, especially if they are used to more traditional hosting environments.

wordpress theme optimization technique

Optimizing your WordPress theme is crucial for improving website performance, user experience, and search engine rankings. Here are some techniques you can use to optimize your WordPress theme:

- 1. Choose a Lightweight Theme:**
 - Start with a well-coded, lightweight theme. Avoid themes with excessive features or complex code that can slow down your site.
- 2. Responsive Design:**
 - Ensure that your theme is responsive, meaning it adapts well to different screen sizes and devices. This is not only important for user experience but also for SEO.
- 3. Minimize HTTP Requests:**
 - Reduce the number of elements on a page that require separate HTTP requests, such as images, scripts, and stylesheets. Combine and minify CSS and JavaScript files to decrease load times.
- 4. Optimize Images:**
 - Compress and optimize images before uploading them to WordPress. You can use image optimization plugins or online tools to reduce file sizes without sacrificing quality.
- 5. Lazy Loading:**
 - Implement lazy loading for images, so they only load when they come into the user's viewport. This can significantly improve initial page load times.
- 6. Use a Content Delivery Network (CDN):**
 - Utilize a CDN to distribute your static assets (like images, stylesheets, and scripts) across multiple servers worldwide. This reduces latency and speeds up content delivery.
- 7. Browser Caching:**
 - Enable browser caching to store static files in visitors' browsers. This reduces the need to download the same files on subsequent visits, improving load times for returning users.
- 8. Enable GZIP Compression:**
 - **Enable GZIP compression on your server to reduce the size of your web pages, making them load faster.** Most modern web servers support GZIP compression.
- 9. Optimize CSS and JavaScript:**
 - Optimize and minimize your CSS and JavaScript files. Remove unnecessary code, spaces, and comments. Consider using tools or plugins that automate this process.
- 10. Clean Up Your Database:**
 - Regularly clean up your WordPress database by removing unnecessary data, such as post revisions, spam comments, and unused plugins. This can help improve database performance.
- 11. Limit External Requests:**
 - Minimize the number of external requests your theme makes, such as requests to external APIs or third-party services. These requests can introduce latency.
- 12. Optimize WordPress Queries:**
 - Optimize your WordPress queries to ensure efficient database performance. Consider using caching plugins to store query results and reduce the load on the database.
- 13. Update Your Theme and Plugins:**
 - Keep your theme and plugins up to date. Developers often release updates to improve performance, fix bugs, and enhance security.
- 14. Disable Unused Features:**
 - Disable any theme features or functionalities that you don't use. This can include unnecessary widgets, post types, or custom features that may add unnecessary bloat.
- 15. Implement AMP (Accelerated Mobile Pages):**
 - Consider implementing AMP for your theme, especially if a significant portion of your traffic comes from mobile devices. AMP pages load quickly on mobile devices, enhancing the user experience.

how to optimize plugin of wordpress

Optimizing WordPress plugins is crucial for maintaining a fast and efficient website. Here are several techniques to optimize plugins in WordPress:

1. Choose Lightweight Plugins:

- Select plugins that are well-coded and lightweight. Avoid plugins that have excessive features or unnecessary functionalities, as they can add unnecessary bloat to your website.

2. Update Plugins Regularly:

- Keep all your plugins up to date. Developers release updates not only for new features but also to fix bugs, enhance security, and improve performance.

3. Limit the Number of Plugins:

- Evaluate the necessity of each plugin and only keep the ones that are essential for your site's functionality. Having too many plugins, even if they are lightweight, can still impact performance.

4. Use a Plugin Performance Profiler:

- Consider using a plugin performance profiler, such as the P3 (Plugin Performance Profiler) plugin, to identify which plugins are consuming the most resources. This can help you make informed decisions about whether to keep or replace certain plugins.

5. Optimize Plugin Settings:

- Review and optimize the settings of each plugin. Some plugins come with default settings that may not be optimal for your site. Adjust configurations to match your specific needs.

6. Caching Plugins:

- Implement a caching plugin to cache dynamic content and reduce the server load. Popular caching plugins like W3 Total Cache or WP Super Cache can significantly improve performance.

7. Database Cleanup:

- Use plugins or tools to clean up your WordPress database regularly. This can help remove unnecessary data such as post revisions, spam comments, and transient options, reducing the database size and improving efficiency.

8. Lazy Load for Images and Videos:

- Utilize lazy loading for images and videos to defer the loading of off-screen media until the user scrolls to them. This can improve initial page load times.

9. Minify CSS and JavaScript:

- Minify CSS and JavaScript files used by your plugins. This reduces file sizes and speeds up loading times. Some caching plugins have built-in options for minification.

10. Optimize Images:

- If a plugin handles images, ensure it optimizes them for the web. Consider using an image optimization plugin or an external service to compress and optimize images.

11. Disable Unused Features:

- Disable any unnecessary features or functionalities provided by plugins. Some plugins offer a range of features, and you may not need all of them. Only enable what you require.

12. Limit External Requests:

- Minimize the number of external requests made by plugins. External requests to third-party APIs or services can introduce latency. Evaluate whether each external request is necessary for your site.

13. Database Query Optimization:

- If a plugin performs database queries, ensure they are optimized. Avoid unnecessary or poorly optimized database queries, as they can impact page load times.

14. Monitor Performance:

- Regularly monitor your site's performance using tools like Google PageSpeed Insights or GTmetrix. Keep an eye on the impact of new plugins or plugin updates on your site's performance.

how to optimize database in wordpress

Optimizing the WordPress database is essential for maintaining good performance, reducing bloat, and ensuring efficient operations. Here are steps to optimize the database in WordPress:

1. Backup Your Database:

- Before making any changes, it's crucial to create a backup of your WordPress database. You can use a plugin like UpdraftPlus or manually export your database using tools like phpMyAdmin.

2. Use a Database Optimization Plugin:

- There are several plugins available that can help you optimize your database. Popular choices include "WP-Optimize" and "WP-Sweep." Install and activate one of these plugins to simplify the optimization process.

3. Remove Unused Themes and Plugins:

- Deactivate and delete any unused themes and plugins. These can sometimes leave behind unnecessary data in the database. Always keep the active theme and necessary plugins up to date.

4. Delete Post Revisions:

- WordPress automatically saves revisions of your posts as you write. Over time, these revisions can accumulate and bloat your database. You can delete old post revisions with plugins like "WP-Optimize" or by running SQL queries.

"WP-Optimize" or by running SQL queries.

sql

 Copy code

```
DELETE FROM wp_posts WHERE post_type = 'revision';
```

5. Clear Expired Transients:

- Transients are temporary data stored in the database. Use the following SQL query to delete expired transients:

sql

 Copy code

```
DELETE FROM wp_options WHERE option_name LIKE '_transient_timeout%';
```

sql

 Copy code

```
DELETE FROM wp_options WHERE option_name LIKE '_transient%';
```

6. Optimize Database Tables:

- Optimize your database tables to reclaim space and defragment data. Some optimization plugins have a feature to do this, or you can run the SQL query:

```
sql
```

 Copy code

```
OPTIMIZE TABLE your_table_name;
```

Replace `your_table_name` with the actual name of your database table.

7. Remove Spam Comments:

- Delete spam comments, which can accumulate in the `wp_comments` table. You can use the following SQL query:

```
sql
```

 Copy code

```
DELETE FROM wp_comments WHERE comment_approved = 'spam';
```

8. Limit Post Revisions:

- Limit the number of post revisions WordPress saves. Add the following line to your wp-config.php file:

```
php
```

 Copy code

```
define('WP_POST_REVISIONS', 5);
```

Adjust the number to limit the total revisions stored.

9. Optimize Options Table:

- The `wp_options` table can grow over time. Some plugins and themes may leave behind unused options. You can use the following SQL query to clean them up:

```
sql
```

 Copy code

```
DELETE FROM wp_options WHERE option_name LIKE '%_transient_%';
```

10. Check and Repair Database:

- Some database issues can be fixed by checking and repairing the database. You can do this using the following SQL query:

```
sql
```

 Copy code

```
REPAIR TABLE your_table_name;
```

Replace `your_table_name` with the actual name of your database table.

11. Use PHPMyAdmin:

- If you are comfortable with database management, you can use phpMyAdmin to manually run queries and optimize tables. Be cautious and make sure you have a backup before making changes.

Performance

1. Run performance tests.
2. Choose a reliable hosting provider.
3. Update everything.
4. Use the latest version of PHP.
5. Delete unused plugins.
6. Install high-quality plugins only.
7. Use a lightweight theme.
8. Optimize images.
9. Try lazy loading your content.
10. Don't host videos on your server.
11. Reduce CSS and JavaScript file sizes.
12. Install a WordPress caching plugin.
13. Simplify your page designs and content.
14. Tidy up your WordPress database.
15. Limit or disable post revisions.
16. Avoid redirects.
17. Turn off pingbacks and trackbacks.
18. Use a CDN.
19. Compress files with GZIP.
20. Limit external scripts.
21. Don't fall victim to hotlinking.
22. Schedule tasks for low-traffic periods.
23. Split long posts into multiple parts.
24. Paginate posts.
25. Paginate comments.

gzip compression

Gzip compression is a method used to reduce the size of files, particularly text files, on computer systems. The term "gzip" stands for GNU zip, and it refers to the software application and file format that implements this compression algorithm.

Here's how gzip compression works:

1. **Compression Algorithm:** Gzip uses the DEFLATE compression algorithm to compress files. DEFLATE is a combination of LZ77 (Lempel-Ziv 1977) and Huffman coding.
2. **Lossless Compression:** Gzip compression is lossless, meaning that the original data can be perfectly reconstructed from the compressed data. No information is lost during the compression and decompression processes.
3. **File Format:** Gzip not only refers to the compression algorithm but also to the file format associated with it. Files compressed with gzip typically have a ".gz" extension.
4. **Usage:** Gzip is commonly used in Unix and Linux environments. It is often employed to compress web pages on the server side before they are sent to a client's web browser. This reduces the amount of data that needs to be transmitted over the network, leading to faster loading times for web pages.
5. **Command-Line Tool:** Gzip is also the name of the command-line tool used to compress and decompress files. Users can use commands like `gzip` to compress a file (creating a .gz file) and `gunzip` to decompress a gzipped file.

Here's a simple example of compressing a file using gzip on the command line:

```
bash                                     ⌂ Copy code  
gzip filename.txt
```

This command will create a compressed file named `filename.txt.gz` while leaving the original file (`filename.txt`) intact.

To decompress a gzipped file, you can use the `gunzip` command:

```
bash                                     ⌂ Copy code  
gunzip filename.txt.gz
```

After decompression, the original file (`filename.txt`) is restored.

Gzip is widely supported across different operating systems and is an efficient way to reduce file sizes for storage or transmission.



Hooks

Hooks in WordPress are essentially a system for allowing developers to modify or extend the behavior of WordPress core, themes, or plugins without directly modifying their code.

Hooks are a way to customize and extend the functionality of the core software or plugins.

Action hooks

Action hooks allow you to execute custom functions at specific points in the WordPress execution process.

The **init** action hook is commonly used to perform actions when WordPress is initializing.

Filter hooks

Filter hooks allow you to modify data before it is displayed or processed by WordPress.

The `the_content` filter hook is often used to modify the content of a post before it's displayed.

Priority

Priority is a concept associated with hooks, and it determines the order in which functions are executed when a hook is triggered.

Lower Priority Numbers Execute Earlier: If you assign a lower number to the priority, your function will be executed earlier in the process.

Higher Priority Numbers Execute Later: If you assign a higher number to the priority, your function will be executed later in the process.

For example, a priority of 5 will execute before a priority of 10.

Why Priorities Matter:

1. Order of Execution:

- When multiple functions are attached to the same hook, priorities determine the order in which they are executed.
- Lower priority numbers go first, and higher priority numbers go later.

2. Dependencies and Interactions:

- If one function relies on the results of another, you might need to control the order of execution using priorities.

3. Ensuring Correct Timing:

- Sometimes, it's crucial to ensure that certain actions happen before or after others during the WordPress lifecycle.

XML-RPC (XML Remote Procedure Call)

XML-RPC (XML Remote Procedure Call) is a remote procedure call (RPC) protocol encoded in XML.

In the context of WordPress, **XML-RPC** is a feature that enables communication between different systems, allowing you to perform various actions on a WordPress site remotely.

It provides a way for external applications or services to interact with a WordPress site, such as posting content, managing comments, and more, without requiring direct access to the WordPress admin interface.

Here are some key points about XML-RPC in WordPress:

1. Purpose:

- XML-RPC is designed to facilitate communication between different platforms and systems over the web.
- In the context of WordPress, XML-RPC allows external applications, clients, or services to interact with a WordPress site by sending and receiving XML-formatted requests.

2. Functionality:

- XML-RPC in WordPress supports a range of functionalities, including:
 - Creating and editing posts.
 - Retrieving post lists.
 - Managing categories and tags.
 - Uploading media files.
 - Managing comments.
 - Authenticating users.
 - and more.

3. Endpoints:

- The XML-RPC endpoint for a WordPress site is typically located at `http://yoursite.com/xmlrpc.php`.
- External applications or services send XML-RPC requests to this endpoint to perform various actions on the WordPress site.

4. Security Concerns:

- XML-RPC has been a source of security concerns in the past, leading some users to disable it for improved security.
- WordPress has implemented measures to address security issues related to XML-RPC, and newer versions have improved security features.
- If you're concerned about security, you can use security plugins or other methods to control or limit XML-RPC access.

5. Usage:

- XML-RPC is used by various applications and services, such as mobile apps, desktop blogging clients, and third-party integrations, to interact with WordPress sites remotely.

6. Disabling XML-RPC:

- Some users may choose to disable XML-RPC if they don't need its functionalities and want to reduce potential security risks.
- Keep in mind that certain applications or services may rely on XML-RPC, so disabling it may affect their ability to communicate with your WordPress site.

Click jacking in wordpress

Clickjacking is a type of attack where a malicious website or element is positioned transparently over a legitimate website, tricking the user into clicking on something different from what they perceive.

Clickjacking could potentially be used to deceive users and make them perform unintended actions on a WordPress site.

To mitigate clickjacking in WordPress, you can take several measures:

1. X-Frame-Options Header:

- Configure your web server or use a security plugin to set the X-Frame-Options header. This header instructs the browser whether a page should be allowed to be displayed in a frame or iframe. The `SAMEORIGIN` value ensures that the page can only be embedded on the same origin.

apache

 Copy code

```
Header always append X-Frame-Options SAMEORIGIN
```

In the context of WordPress, some security plugins might offer settings to enable this header.

2. Content Security Policy (CSP):

- Implement a Content Security Policy on your WordPress site. CSP is a security standard that helps prevent various types of attacks, including clickjacking. It allows you to define where resources can be loaded from, reducing the risk of malicious embedding.

apache

 Copy code

```
Content-Security-Policy: frame-ancestors 'self';
```

This policy ensures that your pages can only be framed by pages from the same origin.

3. WordPress Security Plugins:

- Consider using a reputable WordPress security plugin. Some security plugins include features to prevent clickjacking by adding the necessary headers automatically.

4. Theme and Plugin Security:

- Ensure that the themes and plugins you use on your WordPress site are from trusted sources and regularly updated. Vulnerabilities in themes or plugins could potentially be exploited for clickjacking attacks.

5. Stay Informed and Update:

- Keep your WordPress installation, themes, and plugins up to date. Developers release updates to address security vulnerabilities, and maintaining an updated site is crucial for security.

6. .htaccess Configuration:

- If you have access to your server's .htaccess file, you can add directives to enhance security. For example, you can implement the X-Frame-Options header using .htaccess:

```
apache                                         Copy code

<IfModule mod_headers.c>
    Header always append X-Frame-Options SAMEORIGIN
</IfModule>
```

Ensure you have a backup before modifying .htaccess, and be cautious to avoid syntax errors.

API security features

1. Authentication:

- **Token-Based Authentication:** Use tokens (such as API keys, JSON Web Tokens - JWTs, or OAuth tokens) to authenticate API requests. Require clients to include these tokens in their requests to verify their identity.
- **OAuth 2.0:** If your API will be accessed by third-party applications, consider implementing OAuth 2.0 for secure and standardized authentication.

2. Authorization:

- Implement proper access controls and permissions. Ensure that users and applications have the appropriate level of access to resources based on their roles and responsibilities.
- Regularly review and update access controls to reflect changes in your application's requirements and user roles.

3. SSL/TLS Encryption:

- Use HTTPS (SSL/TLS) to encrypt data transmitted between clients and the API server. This helps protect sensitive information, such as authentication tokens and user data, from eavesdropping and man-in-the-middle attacks.

4. Rate Limiting:
<ul style="list-style-type: none">Implement rate limiting to prevent abuse and protect your API from denial-of-service (DoS) attacks. This restricts the number of requests a client can make within a specified time period.
5. Input Validation and Sanitization:
<ul style="list-style-type: none">Validate and sanitize all incoming data to prevent common security vulnerabilities, such as injection attacks (e.g., SQL injection, NoSQL injection) and cross-site scripting (XSS) attacks.
6. Output Encoding:
<ul style="list-style-type: none">Encode output data to prevent XSS attacks. Ensure that data returned by the API is properly escaped to prevent malicious scripts from being executed in the context of a user's browser.
7. Error Handling:
<ul style="list-style-type: none">Implement proper error handling to avoid leaking sensitive information in error responses. Provide generic error messages to clients and log detailed errors on the server side for debugging purposes.
8. Audit Logs:
<ul style="list-style-type: none">Maintain detailed logs of API requests and responses, including information about the client, timestamp, and actions performed. These logs can be invaluable for debugging, monitoring, and investigating security incidents.
9. Cross-Origin Resource Sharing (CORS):
<ul style="list-style-type: none">If your API is meant to be accessed by clients from different domains, configure CORS headers to control which domains are allowed to make requests to your API. This helps prevent unauthorized cross-origin requests.
10. JWT (JSON Web Tokens) Validation:
<ul style="list-style-type: none">If you're using JWTs for authentication, validate the tokens carefully, checking their expiration, issuer, and signature. This ensures that the tokens are legitimate and have not been tampered with.
11. Security Headers:
<ul style="list-style-type: none">Include security headers in your API responses, such as Content Security Policy (CSP), Strict-Transport-Security (HSTS), and X-Content-Type-Options. These headers enhance the overall security posture of your API.
12. API Versioning:
<ul style="list-style-type: none">Implement versioning in your API to allow for backward compatibility. This helps prevent disruptions to existing clients when making changes to the API.
13. Security Testing:
<ul style="list-style-type: none">Regularly conduct security assessments, including penetration testing and code reviews, to identify and address potential vulnerabilities in your API.
14. Security Education:
<ul style="list-style-type: none">Educate developers, API consumers, and anyone interacting with the API about security best practices. Promote awareness of potential risks and the importance of secure coding practices.

CSRF stands for Cross-Site Request Forgery

CSRF stands for Cross-Site Request Forgery, and it is a type of security vulnerability that can affect web applications, including WordPress. In a CSRF attack, an attacker tricks a user's browser into making an unintended and potentially harmful request to a web application on which the user is authenticated. This occurs when the user visits a malicious website or clicks on a manipulated link, leading to the execution of unauthorized actions on the target site.

In the context of WordPress, CSRF attacks can pose a threat to the security of a user's account and the overall integrity of a WordPress site. Here's how a CSRF attack might work in the context of WordPress:

- 1. User Authentication:**
 - The victim is authenticated on a WordPress site (e.g., logged in as an administrator).
- 2. Malicious Action Triggered:**
 - The victim unknowingly visits a malicious website or clicks on a link that contains a crafted request targeting a WordPress action.
- 3. Unintended Request Sent:**
 - Because the victim is authenticated on the WordPress site, the browser automatically includes the user's credentials in the request to the targeted action.
- 4. Unauthorized Action Executed:**
 - The targeted action is executed on the WordPress site with the victim's credentials, performing an action that the user did not intend.

To prevent CSRF attacks in WordPress, several security measures can be implemented:

- 1. Use Nonces:**
 - WordPress uses nonces (numbers used once) as a security measure to protect against CSRF attacks. Nonces are unique tokens generated for each user and request, and they are included in forms and URLs to verify the legitimacy of the request.
- 2. Check Referrer Header:**
 - WordPress checks the HTTP Referer header to verify that the request originated from the same site. While this check is not foolproof (as the Referer header can be manipulated), it adds an additional layer of protection.
- 3. CSRF Protection in Forms:**
 - When creating forms in WordPress, ensure that they include nonces and that nonces are properly validated before processing form submissions.
- 4. Implement SameSite Cookies:**
 - Use the SameSite attribute for cookies to control when cookies are sent with cross-site requests. This attribute helps prevent CSRF attacks by ensuring that cookies are only sent in a first-party context.
- 5. Security Headers:**
 - Implement security headers, such as Content Security Policy (CSP), to enhance overall security and mitigate various web vulnerabilities, including CSRF.
- 6. Keep Software Updated:**
 - Regularly update WordPress, themes, and plugins to ensure that you benefit from the latest security enhancements and patches.

Composer in Wordpress

It is often employed in the development and management of WordPress projects, especially those involving custom themes or plugins. Composer is a dependency manager for PHP, and it allows you to declare the libraries your project depends on and manage them for you.

Managing WordPress Core:

- While Composer is not the standard method for managing WordPress core itself, some developers choose to use Composer to manage their project dependencies, including WordPress. The `johnpbloch/wordpress` package is a popular Composer package that allows you to manage the WordPress core files using Composer.

Managing Plugins and Themes:

- Composer can be used to manage third-party plugins and themes. If a plugin or theme is available on Packagist (the main repository for Composer packages), you can include it in your project's `composer.json` file.

Autoloading and Namespace Management:

- Composer provides an autoloader that simplifies class loading in your PHP projects. This can be especially useful in larger projects or when developing custom themes and plugins with a modular structure.

WordPress Starter Projects:

- There are Composer-based starter projects and frameworks designed specifically for WordPress development. These projects often include a structure for managing themes, plugins, and other project-specific files using Composer.

Development Tools:

- Some developers use Composer to manage development tools and libraries, such as PHPUnit for testing, PHP CodeSniffer for coding standards, and other utilities that enhance the development workflow.

Transient in wordpress

1. Transient Basics:

- A transient is a piece of data that is stored in the WordPress database with an expiration time.
- It can be any type of data, such as an array, object, string, or integer.
- Transients are useful for caching data that is computationally expensive to generate or retrieve.

2. Setting Transients:

- To set a transient, you use the `set_transient` function, specifying a unique name for the transient, the data to be stored, and the expiration time (in seconds).

```
php Copy code  
  
set_transient('my_transient_key', $data, 3600); // Expires in 1 hour (3600 seconds)
```

3. Getting Transients:

- To retrieve the stored data from a transient, you use the `get_transient` function.

```
php Copy code  
  
$cached_data = get_transient('my_transient_key');  
if (false === $cached_data) {  
    // Data not in the transient, regenerate or fetch it  
    $cached_data = expensive_function_to_generate_data();  
    set_transient('my_transient_key', $cached_data, 3600); // Cache for 1 hour  
}  
// Use $cached_data...
```



In this example, if the transient is not present or has expired, the data is regenerated, and a new

4. Deleting Transients:

- Transients are automatically deleted from the database when they expire, but you can also delete them manually using the ``delete_transient`` function.

```
php
```

 Copy code

```
delete_transient('my_transient_key');
```

5. Examples of Transient Use Cases:

- **API Requests:** Storing the results of expensive API requests to avoid making the same request on every page load.
- **Complex Queries:** Caching the results of complex database queries to improve performance.
- **Dynamic Content:** Storing dynamic content that doesn't change frequently to reduce server load.

6. Transients vs. Options:

- Transients are similar to options in WordPress, but they have an expiration time, making them suitable for temporary caching. Options, on the other hand, persist until explicitly deleted.

7. Object Caching:

- WordPress also supports object caching mechanisms, which can be used in conjunction with transients to store cached data in memory (e.g., using Memcached or Redis).

In details

1. What Are Transients?

- **Definition:** Transients are a type of caching mechanism in WordPress designed for storing temporary data with an expiration time.
- **Use Case:** They are useful for caching data that is resource-intensive to generate or fetch, such as the result of an API request, complex database query, or expensive computation.

2. Setting Transients:

- To set a transient, you use the `set_transient` function.
- Syntax: `set_transient(\$transient_name, \$transient_value, \$expiration_time);`
- Example:

```
php Copy code
      set_transient('my_transient_key', $data, 3600); // Expires in 1 hour (3600 seconds)
```

3. Getting Transients:

- To retrieve the stored data from a transient, you use the `get_transient` function.
- Syntax: `get_transient(\$transient_name);`

```
php
```

 Copy code

```
$cached_data = get_transient('my_transient_key');
if (false === $cached_data) {
    // Data not in the transient, regenerate or fetch it
    $cached_data = expensive_function_to_generate_data();
    set_transient('my_transient_key', $cached_data, 3600); // Cache for 1 hour
}
```

- Explanation:

- If the transient is not present or has expired (`get_transient` returns `false`), you can regenerate or fetch the data and set a new transient.

4. Deleting Transients:

- Transients are automatically deleted from the database when they expire, but you can also delete them manually using the `delete_transient` function.
- Syntax: `delete_transient($transient_name);`
- Example:

```
php
```

 Copy code

```
delete_transient('my_transient_key');
```

5. Examples of Transient Use Cases:

- **API Requests:** Store API responses to avoid making repetitive requests.
- **Complex Queries:** Cache the results of complex and resource-intensive database queries.
- **Dynamic Content:** Cache dynamic content that doesn't change frequently to reduce server load.

6. Transients vs. Options:

- Transients are similar to options in WordPress, but they have an expiration time.
- Transients are suitable for temporary caching, while options persist until explicitly deleted.
- Options are generally used for more permanent storage of configuration settings or data.

7. Object Caching:

- WordPress supports object caching, which involves storing cached data in memory (e.g., using Memcached or Redis).
- Object caching can work in conjunction with transients to improve performance further.

8. Considerations:

- **Expiration Time:** Choose an appropriate expiration time for your transients based on the nature of the data and how frequently it changes.
- **Judicious Use:** Use transients judiciously, as excessively long expiration times may result in serving outdated or stale data.
- **Performance Benefits:** Transients can significantly improve the performance of your WordPress site

Caching in wordpress

Caching in WordPress is a crucial technique to improve the performance and speed of a website. By caching, you store static copies of dynamically generated content, reducing the need to regenerate the same content repeatedly for each visitor. Here are the key aspects of caching in WordPress:

1. Page Caching:

- **What it Does:** Page caching involves storing entire HTML pages so that subsequent requests can be served without regenerating the entire page.
- **Popular Plugins:** There are several popular caching plugins for WordPress, such as W3 Total Cache, WP Super Cache, and WP Rocket, that implement page caching.

2. Object Caching:

- **What it Does:** Object caching involves storing the results of expensive database queries or complex operations in memory.
- **Implementation:** WordPress supports object caching through plugins or by integrating with external systems like Memcached or Redis.
- **Example Plugin:** The "Redis Object Cache" or "Memcached Object Cache" plugins can be used to enable object caching in WordPress.

3. Opcode Caching:

- **What it Does:** Opcode caching involves caching the compiled PHP code to avoid recompiling it on each request.
- **Implementation:** PHP opcode caches like OPcache can be configured at the server level to store precompiled PHP code in memory.

4. Browser Caching:

- **What it Does:** Browser caching instructs the visitor's browser to store static assets locally, reducing the need to download them on subsequent visits.
- **Implementation:** Achieved through setting appropriate HTTP headers for caching static resources like images, stylesheets, and scripts.

5. Transient Caching:

- **What it Does:** Transients are a form of caching used to store temporary data with an expiration time.
- **Use Cases:** Useful for caching data retrieved from APIs, results of complex calculations, or dynamic content that doesn't change frequently.
- **Implementation:** Utilized through functions like `set_transient`, `get_transient`, and `delete_transient` in WordPress.

6. Full-Page Caching vs. Fragment Caching:

- **Full-Page Caching:** Stores entire HTML pages.
- **Fragment Caching:** Stores specific sections or fragments of a page.

7. Cache Invalidation:

- **What it Does:** Cache invalidation refers to the process of clearing or updating cached content when it becomes outdated or when changes are made.
- **Strategies:** Some caching plugins and systems offer mechanisms for automatic cache purging based on events, while others may require manual intervention.

8. Varnish Cache:

- **What it Does:** Varnish is a reverse proxy server that sits in front of your web server and caches static content.
- **Implementation:** Often used in conjunction with WordPress and caching plugins for additional performance benefits.

9. CDN (Content Delivery Network):

- **What it Does:** A CDN is a network of distributed servers that cache and serve static assets, reducing the load on the origin server and improving page load times.
- **Integration:** Many CDNs seamlessly integrate with WordPress, and popular options include Cloudflare, MaxCDN (now part of StackPath), and Amazon CloudFront.

10. Caching Best Practices:

- **Regular Updates:** Keep WordPress, themes, and plugins updated to benefit from performance improvements and security patches.
- **Testing:** Test your site thoroughly after implementing caching to ensure that dynamic content is not affected.
- **Monitoring:** Monitor server resources and cache hit rates to optimize caching settings.

difference between transient and option

ChatGPT

In WordPress, both transients and options are mechanisms for storing data, but they serve different purposes and have distinct characteristics. Here's a breakdown of the key differences between transients and options:

Transients:

1. Expiration Time:

- **Characteristic:** Transients have an expiration time. They are meant for storing temporary data that is expected to change or expire after a specific period.
- **Use Case:** Suitable for caching data that is only relevant for a limited time, such as the result of an API request or a computationally expensive operation.

2. Storage in Database:

- **Location:** Transients are stored in the WordPress database.
- **Automated Cleanup:** WordPress automatically deletes expired transients from the database, minimizing the need for manual management.

3. Usage Examples:

- Setting a Transient:

```
php
```

 Copy code

```
set_transient('my_transient_key', $data, 3600); // Expires in 1 hour
```

- Getting a Transient:

```
php
```

 Copy code

```
$cached_data = get_transient('my_transient_key');
```

4. Transient Functions:

- `set_transient`: Sets a transient with a specified expiration time.
- `get_transient`: Retrieves the value of a transient.
- `delete_transient`: Deletes a transient.

Options:

1. Persistence:

- **Characteristic:** Options are persistent and remain in the database until explicitly deleted.
- **Use Case:** Suitable for storing configuration settings, site-wide preferences, or data that needs to persist across page loads.

2. Storage in Database:

- **Location:** Options are also stored in the WordPress database.
- **Manual Cleanup:** Options need to be explicitly deleted using the `delete_option` function when no longer needed.

3. Usage Example:

- Setting an Option:

```
php
```

 Copy code

```
update_option('my_option_key', $data);
```

- Getting an Option:

```
php
```

 Copy code

```
$stored_data = get_option('my_option_key');
```

4. Option Functions:

- `update_option`: Adds or updates the value of an option.
- `get_option`: Retrieves the value of an option.
- `delete_option`: Deletes an option.

Use Cases:

- Transients:

- Caching data with an expiration time.
- Storing temporary data that can be regenerated or updated periodically.
- Useful for scenarios where it's acceptable to serve slightly outdated data for a short period.

- Options:

- Storing configuration settings.
- Managing site-wide preferences.
- Storing data that needs to persist consistently across different page loads.

Summary:

- **Expiration:** Transients have an expiration time and are automatically cleaned up by WordPress.
- **Persistence:** Options are persistent and remain until explicitly deleted.
- **Use Cases:** Transients are ideal for temporary data and caching, while options are better suited for long-term storage of configuration settings and preferences.

Hartbit API

It seems there might be a typo in your query, and you might be referring to "Heartbeat API" in WordPress. The WordPress Heartbeat API is a feature that enables communication between the browser and the server in real-time. It facilitates tasks like autosaving drafts, session management, and other real-time interactions within the WordPress dashboard.

1. Real-Time Communication:

- The Heartbeat API allows the server to send regular requests to the browser, creating a real-time connection between the two.

2. Default Frequency:

- By default, the Heartbeat API sends requests every 15 seconds on post edit pages and every 60 seconds on the dashboard.

3. Use Cases:

- Autosave:** The Heartbeat API is often used for autosaving drafts. It helps prevent data loss by regularly sending updates to the server as the user edits a post or page.
- User Presence:** It can be used to monitor user activity and presence on the site, which can be useful for features like determining if a user is still online.

4. Endpoints:

- The main Heartbeat API endpoint is `/wp-admin/admin-ajax.php?action=heartbeat`.

5. Customization:

- Developers can customize the Heartbeat API behavior, such as adjusting the frequency of requests or extending its functionality.

6. Hooks and Filters:

- WordPress provides hooks and filters for developers to modify or extend the Heartbeat API functionality. For example, `heartbeat_send` and `heartbeat_nopriv_send` are actions that allow developers to hook into the sending of Heartbeat API data.

Here's a simple example of hooking into the Heartbeat API to modify the interval:

```
php Copy code
function custom_heartbeat_settings($settings) {
    $settings['interval'] = 60; // Set the interval to 60 seconds
    return $settings;
}

add_filter('heartbeat_settings', 'custom_heartbeat_settings');
```

Please note that while the Heartbeat API provides valuable real-time communication, its continuous requests can contribute to server load. In some cases, it might be necessary to fine-tune its behavior to avoid unnecessary strain on the server, especially on high-traffic sites.



INDEX Syntax

```
CREATE INDEX index_name  
ON table_name(column1, column2, column3, .... );
```

```
DROP INDEX index_name  
ON table_name;
```



Yahoo Baba

Yahoo

www.yahooomba.net



Guidelines of Index :

- Automatically creates the indexes for PRIMARY KEY and UNIQUE columns.
- Index columns that you frequently use to retrieve the data.
- Index columns that are used for joins to improve join performance.
- Avoid columns that contain too many NULL values.
- Small tables do not require indexes.



Yahoo Baba

Yahoo

www.yahooomba.net

Navigator Schemas Administration Information Schema: student

SQL File 8* × students

1 SELECT * FROM students
2 WHERE dob > "1999-01-01";
3
4 • CREATE INDEX studdob ON students (dob);
5
6 • SHOW INDEX FROM students;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
students	0	id	1	id	A	4
students	1	studdob	1	dob	A	4

Yahoo
Baba

Navigator Schemas Administration Information Schema: student

SQL File 8* × students

1 SELECT * FROM students
2 WHERE dob > "1999-01-01";
3
4 • CREATE INDEX studdob ON students (dob);
5
6 • DROP INDEX studdob ON students;

I

Yahoo
Baba

Migration of wordpress from localhost to another domain

Migrating a WordPress project from localhost to another domain involves several steps. Below is a general guide to help you through the process. Keep in mind that specific steps may vary depending on your setup and hosting environment.

1. Backup Your Local WordPress Site:

- Use a plugin like "Duplicator," "All-in-One WP Migration," or manually export the database and files.

2. Export the Database:

- In phpMyAdmin or a similar tool, export your local WordPress database.
 - Select the database for your WordPress site.
 - Choose the "Export" tab.
 - Select the export method as "Quick" or "Custom" and click "Go" to download the SQL file.

3. Search and Replace URLs:

- Open the SQL file you exported in a text editor.
- Use a search and replace tool to replace all instances of the local URL with the new domain.
 - For example, replace `http://localhost/your-site` with `http://www.newdomain.com`.

4. Create a Database on the New Server:

- In your hosting control panel, create a new MySQL database and a user with the necessary permissions.

5. Import the Database:

- Use phpMyAdmin or a similar tool to import the modified SQL file into the new database.

6. Upload WordPress Files:

- Copy all WordPress files from your local server to the new server using FTP or a file manager.
- Ensure that the WordPress files are in the root directory or the desired subdirectory of the new domain.

7. Update wp-config.php:

- Open the `wp-config.php` file on the new server and update the database connection details (database name, username, password).

8. Update Permalinks (Optional):

- In the WordPress admin dashboard, go to "Settings" -> "Permalinks" and click "Save Changes" to update the permalink structure.

9. Search for Insecure Content (Optional):

- Use a tool like "Better Search Replace" to search for any instances of HTTP URLs and replace them with HTTPS if your new domain uses SSL.

10. Update Theme and Plugin Settings (Optional):

- Some themes and plugins store settings with absolute URLs. Check and update these settings if needed.

11. Test Your Site:

- Visit your new domain to ensure that everything is working as expected.
- Log in to the WordPress admin dashboard and check for any issues.

12. Update DNS Settings (If Changing Domain):

- If you're changing the domain, update the DNS settings to point to the new server.

Important Notes:

- Always make a backup before making changes.
- Ensure that the new server meets WordPress requirements.
- Check for compatibility issues with the new server environment.
- Adjust file and folder permissions on the new server if needed.

Query Monitor Debug Toolbar

WordPress is the **Query Monitor** plugin, which provides detailed information about database queries, hooks, scripts, styles, and much more.

- You'll see a new menu item labeled "QM" (Query Monitor).
- Click on "QM," and it will show you a dropdown with various options, including queries, hooks, conditionals, and more.
- Explore the different sections to view detailed information about your site's performance, such as database queries, executed hooks, HTTP requests, and more.
- The information provided by Query Monitor can be instrumental in identifying and resolving performance issues and bugs in your WordPress site.

features of the Query Monitor plugin:

1. Database Query Monitoring:
<ul style="list-style-type: none"> Query Monitor provides detailed information about all database queries executed during the loading of a WordPress page. This includes the time it takes for each query to execute and the specific queries themselves.
2. Performance Profiling:
<ul style="list-style-type: none"> It helps identify performance bottlenecks by providing information on the time it takes for different components (queries, scripts, styles, etc.) to load. This is crucial for optimizing the speed and responsiveness of your website.
3. PHP Error Tracking:
<ul style="list-style-type: none"> Query Monitor displays PHP errors and warnings, making it easier to identify and resolve issues related to PHP code on your site.
4. HTTP Requests and REST API Calls:
<ul style="list-style-type: none"> It logs and displays information about HTTP requests made during the page load, including REST API calls. This can be helpful when working with external APIs or third-party services.
5. Theme and Plugin Information:
<ul style="list-style-type: none"> It shows information about the active theme and plugins, including the number of database queries they generate and the time it takes for them to load.
6. Hooks and Actions Monitoring:
<ul style="list-style-type: none"> Query Monitor provides insights into the hooks and actions that are executed during the page load. This is useful for understanding the sequence in which different functions are called.
7. Conditionals and Environment Details:
<ul style="list-style-type: none"> It displays information about the conditionals that are true or false during the page load. This helps you understand which parts of your code are being executed based on different conditions. It also provides details about the server environment.
8. Debugging AJAX Requests:
<ul style="list-style-type: none"> Query Monitor can be useful for debugging AJAX requests by providing information about the requests made and the associated callbacks.
9. Customization and Filtering:
<ul style="list-style-type: none"> The plugin allows you to customize the debugging output and filter the information based on your specific needs. You can enable or disable specific features to focus on particular aspects of your site.
10. Compatibility Testing:
<ul style="list-style-type: none"> Query Monitor can be used to check the compatibility of your theme and plugins with the current version of WordPress, helping you identify potential issues when updating.

Core Web Vitals

Core Web Vitals are a set of specific factors that Google considers important in assessing the user experience of a web page. These metrics are designed to measure various aspects of page performance, loading, and interactivity. The Core Web Vitals consist of three key metrics:

1. Largest Contentful Paint (LCP):
<ul style="list-style-type: none"> LCP measures the loading performance of a web page. It specifically looks at the time it takes for the largest piece of content (such as an image or text block) to become visible to the user.
2. First Input Delay (FID):
<ul style="list-style-type: none"> FID measures the interactivity of a web page. It quantifies the time it takes for a user to interact with the page, such as clicking a button or a link, after the page has started loading.
3. Cumulative Layout Shift (CLS):
<ul style="list-style-type: none"> CLS measures the visual stability of a web page. It assesses the extent to which page elements shift around during loading, which can be disruptive to the user experience.

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. Firstly you should stay calm,
 2. Search for an optimal solution,
 3. Locate the hack,
 4. Contact your hosting company,
 5. Hire IT professional (Expertise of InfoSec),
 6. Restore WordPress Versions,
 7. Scan, Identify and reduce the malware,
 8. Check access permission,
 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 oftenly 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 database -> 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 `<?php get_header(); ?>` 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()

This ensures that the query has been reset to the originally created main query.

wp_reset_postdata()

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_itemColon'       => '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` retrieves 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 Vissal programming

```
a.php [x] form_data.sql [x] contact_us.php [x]
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);
}
?>
```

```
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 }
?>
```

```
42     add_shortcode('form_data_list_shortcode','form_data_list');
43
44     function form_data_list(){
45         include('form_data_list.php');
46     }

```

```
1  <?php
2      global $wpdb;
3      global $table_prefix;
4      $table=$table_prefix.'form_data';
5      $sql="select * from $table";
6      $result=$wpdb->get_results($sql);
7      ?>
8      <table>
9          <tr>
10             <td>ID</td>
11             <td>Name</td>
12         </tr>
13         <?php
14         foreach($result as $list){
15             ?>
16             <tr>
17                 <td><?php echo $list->id?></td>
18                 <td><?php echo $list->name?></td>
19             </tr>
20         <?php
21         }
22         ?>
23     </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(); ?&gt;" &lt;?php post_class(); ?&gt;&gt;
    &lt;?php
        if ( has_post_thumbnail() ) :
            the_post_thumbnail();
        endif;
    ?&gt;
    &lt;header class="entry-header"&gt;
        &lt;h1 class="entry-title"&gt;&lt;?php the_title(); ?&gt;&lt;/h1&gt;
    &lt;/header&gt;
    &lt;div class="entry-content"&gt;
        &lt;?php the_excerpt(); ?&gt;
        &lt;a href="<?php the_permalink(); ?&gt;">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:

```
$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
/**
 * 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

'MultiPHP INI Editor'. Click on it.



On the next screen, you will find a provision to set values for `post_max_size` and `upload_max_filesize`. For them add the values 72M and 64M respectively. Don't forget to click on the 'Apply' button.

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 '';
```

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 '';

        }

        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; ?>
```

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 ?>
```

```
<div class="course-box">
<?php
$wpnew=array(
'post_type'=>'news',
'post_status'=>'publish'
);
$newsquery=new Wp_Query($wpnew);
while($newsquery->have_posts()) {
$newsquery->the_post();
?>
<div class="blog-item">=|</div>
```

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

Reference

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.

```
$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',
        ),
    ) );
)
```

```

$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.

```

$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

```
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);?&gt;"&gt;
    &lt;/input&gt;
}
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');</pre
```

```
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');
```

plugin in WordPress to add custom fields for storing image URL

```
<?php  
/*  
Plugin Name: Image Custom Fields Plugin  
Description: Adds custom fields for storing image URLs.  
  
// Function to add custom field for image URL  
function add_custom_image_field() {  
    // Replace 'post' with the post type where you want to add the custom f  
    $post_type = 'post';  
  
    // Adding the custom field for image URL  
    add_meta_box(  
        'custom_image_field',  
        'Custom Image Field',  
        'display_custom_image_field',  
        $post_type,  
        'normal',  
        'default'  
    );  
}  
}
```

```
// Function to display the custom field input
function display_custom_image_field($post) {
    $value = get_post_meta($post->ID, 'custom_image_field', true);
    echo '<input type="text" id="custom_image_field" name="custom_image_field" value="' . esc_attr($value) . '" />';
}

// Function to save the custom field value
function save_custom_image_field($post_id) {
    if (array_key_exists('custom_image_field', $_POST)) {
        update_post_meta(
            $post_id,
            'custom_image_field',
            sanitize_text_field($_POST['custom_image_field'])
        );
    }
}

// Hooks to execute the functions at appropriate times
add_action('add_meta_boxes', 'add_custom_image_field');
add_action('save_post', 'save_custom_image_field');

stom_image_field', true);
image_field" name="custom_image_field" value="' . esc_attr($value) . '" />';
```

Plugin Name: File & Checkbox Custom Fields Plugin

Description: Adds custom fields for file upload and multiple checkboxes.

```
// Function to add custom fields for file upload and multiple checkboxes
function add_custom_file_checkbox_fields() {
    // Replace 'post' with the post type where you want to add the custom fields.
    $post_type = 'post';

    // Adding the custom field for file upload
    add_meta_box(
        'custom_file_field',
        'Custom File Field',
        'display_custom_file_field',
        $post_type,
        'normal',
        'default'
    );

    // Adding the custom field for multiple checkboxes
    add_meta_box(
        'custom_checkbox_field',
        'Custom Checkbox Field',
        'display_custom_checkbox_field',
        $post_type,
        'normal',
        'default'
    );
}

// Function to display the file upload custom field
function display_custom_file_field($post) {
    $value = get_post_meta($post->ID, 'custom_file_field', true);
    echo '<input type="file" id="custom_file_field" name="custom_file_field" />';
}

// Function to display the multiple checkboxes custom field
function display_custom_checkbox_field($post) {
    $value = get_post_meta($post->ID, 'custom_checkbox_field', true);
    $options = array('Option 1', 'Option 2', 'Option 3'); // Add your checkbox options
    foreach ($options as $option) {
        $checked = in_array($option, $value) ? 'checked' : '';
        echo '<input type="checkbox" value="' . $option . '" ' . $checked . ' />';
    }
}
```

```
        echo '<label><input type="checkbox" name="custom_checkbox_field[]" value="" . esc_attr($option) . "" ' . $checked . ' />' . esc_html($option) . '</label><br />';
    }
}

// Function to save the custom field values
function save_custom_file_checkbox_fields($post_id) {
    if (array_key_exists('custom_file_field', $_FILES)) {
        // Handle file upload here
        // Example: move_uploaded_file($_FILES['custom_file_field']['tmp_name'], $upload_dir . '/' . $_FILES['custom_file_field']['name']);
    }

    if (array_key_exists('custom_checkbox_field', $_POST)) {
        update_post_meta($post_id, 'custom_checkbox_field', $_POST['custom_checkbox_field']);
    }
}

// Hooks to execute the functions at appropriate times
add_action('add_meta_boxes', 'add_custom_file_checkbox_fields');
add_action('save_post', 'save_custom_file_checkbox_fields');
```

```
// Function to display the select dropdown for gender
function display_custom_gender_field($post) {
    $value = get_post_meta($post->ID, 'custom_gender_field', true);
    $options = array('Male', 'Female'); // Options for gender select
    echo '<select id="custom_gender_field" name="custom_gender_field">';
    foreach ($options as $option) {
        $selected = ($value === $option) ? 'selected' : '';
        echo '<option value="' . esc_attr($option) . '" ' . $selected . '>';
    }
    echo '</select>';
}

// Function to save the custom field value
function save_custom_gender_field($post_id) {
    if (array_key_exists('custom_gender_field', $_POST)) {
        update_post_meta($post_id, 'custom_gender_field', $_POST['custom_ge
    }
}

// Hooks to execute the functions at appropriate times
add_action('add_meta_boxes', 'add_custom_gender_field');
add_action('save_post', 'save_custom_gender_field');

c_attr($option) . '" ' . $selected . '>' . esc_html($option) . '</option>';

nder_field', $_POST)) {
    'custom_gender_field', $_POST['custom_gender_field']);
}
```

Opps Method custom field

Refference

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

```
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' );

```

```

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' );

```

content > plugins > my-plugin > public > register.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();
        }
    }

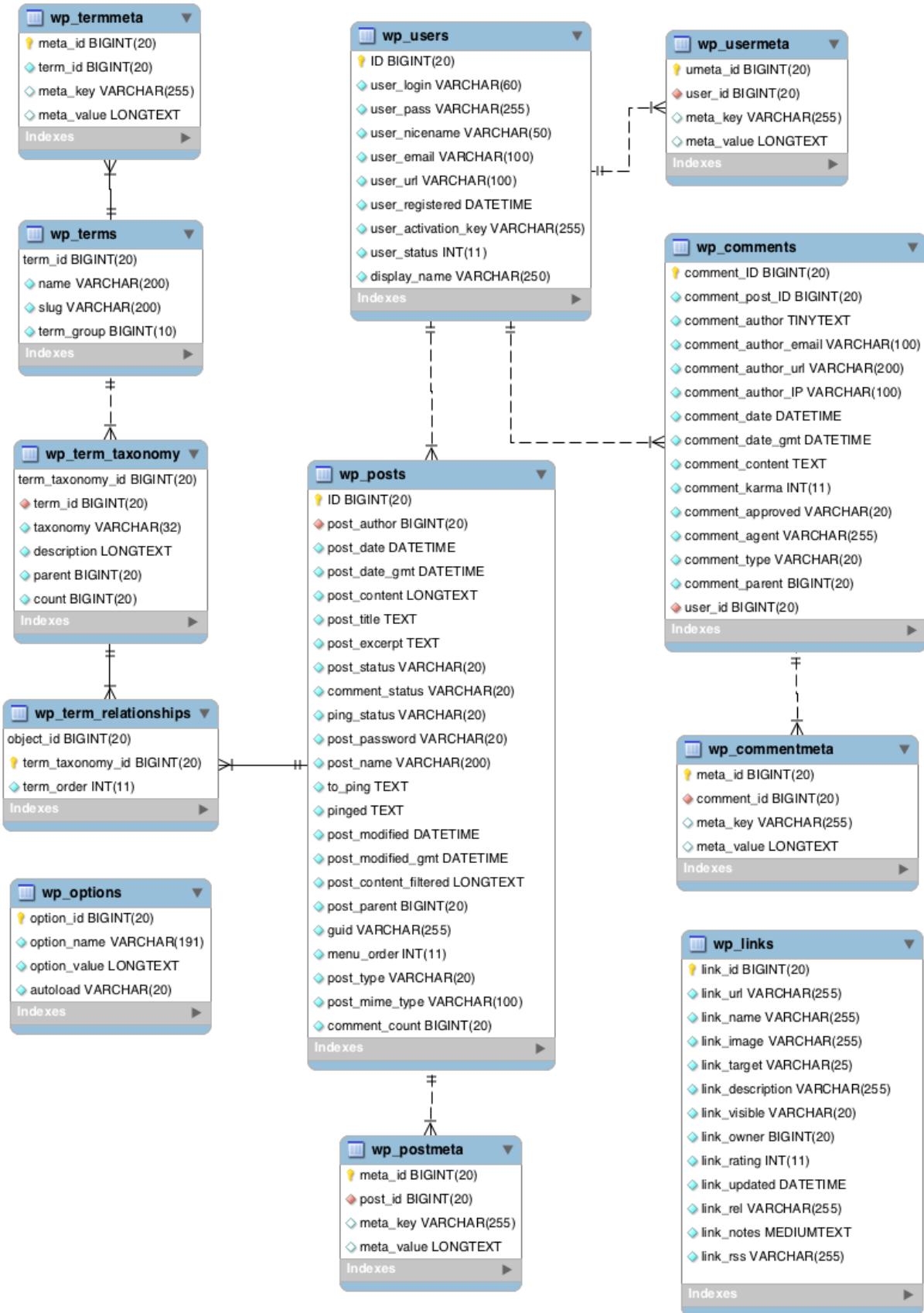
}else{
    echo 'Password must match!';
}
```

```
$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!';
}
}
```



How to create a custom plugin for your shortcode:

```
<?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"]
```

<p>This is some content before the custom shortcode.</p>

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

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

```
$post_data = array(
    'title' => 'New Post Title',
    'content' => 'Content of the new post.',
    'status' => 'publish' // Set the status as needed (draft, pending, published)
);

$response = wp_remote_post(
    'https://your-wordpress-site.com/wp-json/wp/v2/posts',
    array(
        'headers' => array(
            'Content-Type' => 'application/json',
        ),
        'body' => wp_json_encode($post_data),
    )
);

if ( is_wp_error( $response ) ) {
    echo 'Error occurred: ' . $response->get_error_message();
} else {
    $new_post = json_decode( wp_remote_retrieve_body( $response ), true );
    // Handle the newly created post
    print_r($new_post);
}
```

To add a new post in WordPress using WordPress functions and methods, you can use the `wp_insert_post` function. Here's an example:

```
php Copy code

// Create a new post object
$new_post = array(
    'post_title'      => 'New Post Title',
    'post_content'   => 'Content of the new post.',
    'post_status'     => 'publish',
    'post_author'    => 1, // Replace with the author's user ID
    'post_type'       => 'post', // Replace with the post type you want
);

// Insert the post into the database
$post_id = wp_insert_post($new_post);

if (is_wp_error($post_id)) {
    echo 'Error occurred: ' . $post_id->get_error_message();
} else {
    echo 'New post added with ID: ' . $post_id;
}
```

how to create Wordpress custom registration form

Custom Registration Form HTML:

```
<form id="custom-registration-form" action="<?php echo esc_url( admin_url('admin-post.php') ); ?>" method="post">
    <input type="text" name="username" placeholder="Username">
    <input type="email" name="email" placeholder="Email">
    <input type="password" name="password" placeholder="Password">
    <input type="text" name="mobile" placeholder="Mobile Number">
    <input type="text" name="address" placeholder="Address">
    <input type="date" name="dob" placeholder="Date of Birth">
    <input type="submit" name="submit" value="Register">
    <input type="hidden" name="action" value="custom_register">
    <?php wp_nonce_field( 'custom-register-nonce', 'custom-register-nonce' ); ?>
</form>
```

Form Processing in functions.php:

```
function custom_registration() {
    if (isset($_POST['submit'])) {
        if( wp_verify_nonce( $_POST['custom-register-nonce'], 'custom-register-nonce' ) ) {
            $username = sanitize_user( $_POST['username'] );
            $email = sanitize_email( $_POST['email'] );
            $password = $_POST['password'];
            $mobile = sanitize_text_field( $_POST['mobile'] );
            $address = sanitize_text_field( $_POST['address'] );
            $dob = sanitize_text_field( $_POST['dob'] );

            $userdata = array(
                'user_login' => $username,
                'user_email' => $email,
                'user_pass' => $password,
                'user_url' => $mobile, // Storing mobile number in the user's URL field
            );

            $user_id = wp_insert_user( $userdata );

            if ( is_wp_error($user_id) ) {
                echo $user_id->get_error_message();
            } else {
```

```

        // Additional meta fields for user data
        update_user_meta($user_id, 'user_address', $address);
        update_user_meta($user_id, 'user_dob', $dob);

        echo 'Registration successful!';
    }
}

}

add_action('admin_post_custom_register', 'custom_registration');
add_action('admin_post_nopriv_custom_register', 'custom_registration');

```

how to display custom this field on user profile page

```

// Function to add custom fields to user profile
function custom_user_profile_fields($user) {
    ?>
    <h3>Custom Profile Fields</h3>
    <table class="form-table">
        <tr>
            <th><label for="mobile">Mobile Number</label></th>
            <td><input type="text" name="mobile" id="mobile" value=<?php echo
esc_attr(get_the_author_meta('user_url', $user->ID)); ?>" class="regular-
text"></td>
        </tr>
        <tr>
            <th><label for="address">Address</label></th>
            <td><input type="text" name="address" id="address" value=<?php echo
esc_attr(get_user_meta($user->ID, 'user_address', true)); ?>" class="regular-
text"></td>
        </tr>
        <tr>
            <th><label for="dob">Date of Birth</label></th>
            <td><input type="date" name="dob" id="dob" value=<?php echo
esc_attr(get_user_meta($user->ID, 'user_dob', true)); ?>" class="regular-
text"></td>
        </tr>
    </table>
    <?php
}
add_action('show_user_profile', 'custom_user_profile_fields');
add_action('edit_user_profile', 'custom_user_profile_fields');

```

```
// Function to save custom fields on user profile update
function save_custom_user_profile_fields($user_id) {
    if (!current_user_can('edit_user', $user_id)) {
        return false;
    }

    if (isset($_POST['mobile'])) {
        update_user_meta($user_id, 'user_url', $_POST['mobile']); // Save mobile
        to user_url field
    }
    if (isset($_POST['address'])) {
        update_user_meta($user_id, 'user_address', $_POST['address']); // Save
        address to user meta
    }
    if (isset($_POST['dob'])) {
        update_user_meta($user_id, 'user_dob', $_POST['dob']); // Save date of
        birth to user meta
    }
}
add_action('edit_user_profile_update', 'save_custom_user_profile_fields');
```

Jquery

How to get field data from form by jquery

```
<form id="frontEndPostForm">
  <input type="text" name="title" placeholder="Title"><br>
  <textarea name="content" placeholder="Content"></textarea><br>
  <button type="submit" id="submitPostBtn">Submit Post</button>
</form>
<div id="responseMessage"></div>
```

```
$(document).ready(function() {
  $('#frontEndPostForm').submit(function(event) {
    event.preventDefault();

    var newPost = {
      title: $('#frontEndPostForm input[name="title"]').val(),
      content: $('#frontEndPostForm textarea[name="content"]').val(),
      status: 'publish'
    };

    $.ajax({
      type: 'POST',
      url: 'https://your-wordpress-site.com/wp-json/wp/v2/posts',
      data: JSON.stringify(newPost),
      contentType: 'application/json',
      success: function(response) {
        $('#responseMessage').text('Post submitted successfully. Pos
          // Handle the response after post submission
      },
      error: function(error) {
```

how to submit a form with field name, email, mobile, message through with submit button by jquery

```
<form id="contactForm">
    <input type="text" name="name" placeholder="Name"><br>
    <input type="email" name="email" placeholder="Email"><br>
    <input type="tel" name="mobile" placeholder="Mobile"><br>
    <textarea name="message" placeholder="Message"></textarea><br>
    <button type="submit">Submit</button>
</form>
```

```
$(document).ready(function() {
    $('#contactForm').submit(function(event) {
        event.preventDefault(); // Prevent the default form submission

        var formData = $(this).serialize(); // Serialize form data

        $.ajax({
            type: 'POST',
            url: 'YOUR_SERVER_ENDPOINT_URL', // Replace with your server endpoint
            data: formData,
            success: function(response) {
                console.log('Form submitted successfully:', response);
                // Handle the response after form submission
            },
            error: function(error) {
                console.log('Error submitting form:', error);
                // Handle errors during form submission
            }
        });
    });
});
```

how to submit a form with field name, email, mobile, message through with submit button by jquery but without reloading page

```
<form id="contactForm">
  <input type="text" name="name" placeholder="Name"><br>
  <input type="email" name="email" placeholder="Email"><br>
  <input type="tel" name="mobile" placeholder="Mobile"><br>
  <textarea name="message" placeholder="Message"></textarea><br>
  <button type="submit" id="submitBtn">Submit</button>
</form>
<div id="responseMessage"></div>
```

```
document).ready(function() {
  $('#contactForm').submit(function(event) {
    event.preventDefault(); // Prevent the default form submission

    var formData = $(this).serialize(); // Serialize form data

    $.ajax({
      type: 'POST',
      url: 'YOUR_SERVER_ENDPOINT_URL', // Replace with your server endpoint
      data: formData,
      success: function(response) {
        $('#responseMessage').text('Form submitted successfully: ' + response);
        // Handle the response after form submission
      },
      error: function(error) {
        $('#responseMessage').text('Error submitting form: ' + error);
        // Handle errors during form submission
      }
    });
});
```

Form validation

```
$(document).ready(function() {
  $('#contactForm').submit(function(event) {
    event.preventDefault(); // Prevent the default form submission

    // Basic form validation
    var name = $('#name').val();
    var email = $('#email').val();
    var mobile = $('#mobile').val();
    var message = $('#message').val();

    // Regular expressions for email and mobile number validation
    var emailRegex = /^[\w\.-]+@[^\w\.-]+\.\w+$/;
    var mobileRegex = /^\d{10}$/;

    if (name === '' || message === '') {
      alert('Please fill in name and message fields.');
      return;
    }

    if (!emailRegex.test(email)) {
      alert('Please enter a valid email address.');
      return;
    }

    if (!mobileRegex.test(mobile)) {
      alert('Please enter a valid 10-digit mobile number.');
      return;
    }

    var formData = $(this).serialize(); // Serialize form data

    $.ajax({
      type: 'POST',
      url: 'YOUR_SERVER_ENDPOINT_URL', // Replace with your server end
    });
  });
});
```

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

$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;

// 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\(\)](#).

```
1 | function wporg_simple_role() {
2 |     add_role(
3 |         'simple_role',
4 |         'Simple Role',
5 |         array(
6 |             'read'          => true,
7 |             'edit_posts'   => true,
8 |             'upload_files' => true,
9 |         ),
10 |     );
11 |
12 |
13 // Add the simple_role.
14 add_action( 'init', 'wporg_simple_role' );
```

Removing Roles

Remove roles with [remove_role\(\)](#).

```
1 | function wporg_simple_role_remove() {
2 |     remove_role( 'simple_role' );
3 |
4 |
5 // Remove the simple_role.
6 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.

```
1 | function wporg_simple_role_caps() {
2 |     // Gets the simple_role role object.
3 |     $role = get_role( 'simple_role' );
4 |
5 |     // Add a new capability.
6 |     $role->add_cap( 'edit_others_posts', true );
7 |
8 |
9 // Add simple_role capabilities, priority must be after the initial role definit:
10 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

The screenshot shows a code editor interface with two tabs: "index.php" and "custom-registration.php". The "custom-registration.php" tab is active and displays the following PHP code:

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

A yellow circle highlights the condition in line 13: `if(strpos($username, ' ') !== FALSE)`. A yellow arrow points from this highlighted code back to the first step in the validation list above.

custom-login.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

```
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();
    ?>
```

The screenshot shows a code editor with two tabs open: "custom-login.php" and "functions.php". The "functions.php" tab is active and displays the following PHP code:

```
634 // ends here
635
636 function redirect_to_custom_login_page() {
637     wp_redirect(site_url() . "/login");
638     exit();
639 }
640
641 add_action("wp_logout", "redirect_to_custom_login_page");
642
643
644 add_action("init", "fn_redirect_wp_admin");
645
646 function fn_redirect_wp_admin() {
647     global $pagenow;
648     if ($pagenow == "wp-login.php") {
649         wp_redirect(site_url() . "/login");
650         exit();
651     }
652 }
```

The code implements a custom login page. It defines a function `redirect_to_custom_login_page()` that uses `wp_redirect()` to redirect users to the custom login URL (`site_url() . "/login"`) and exits. This function is hooked into the `wp_logout` action. It also defines a function `fn_redirect_wp_admin()` that checks if the current page is the WordPress login page (`"wp-login.php"`). If so, it redirects to the custom login page and exits. This function is hooked into the `init` action.

```
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')

| 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";
}
```

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

function da_custom_menu(){

    add_menu_page( page_title: 'Records', menu_title: 'Record List', capability: 'manage_options' );
    add_submenu_page( parent_slug: 'da-record', page_title: 'Record List Sub Menu', menu_title: 'Sub Menu' );
}

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
wpacademy-like-dislike.php

45 add_action('admin_menu', 'wpac_register_menu_page');
46
47 //Sub-Level Administration Menu
48 /* function wpac_register_menu_page() {
49     add_theme_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html', 30 );
50 }
51 add_action('admin_menu', 'wpac_register_menu_page'); */
52
53 function wpac_plugin_settings(){
54
55     register_setting( option_group, option_name, sanitize_callback );
56
57     add_settings_section( id, title, callback, page );
58
59     add_settings_field( id, title, callback, page, section, args );
60 }
61 add_action('admin_init', 'wpac_plugin_settings');
62 ?>
```

```

27 //Include Scripts & Styles
28 if( !function_exists('wpac_plugin_scripts')) {
29     function wpac_plugin_scripts() {
30         wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
31         wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true );
32     }
33     add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
34 }
35
36 add_menu_page( page_title, menu_title, capability, menu_slug, function, icon_url, position );
37 ?>

```

pwspk-plugin-dev.php — wpdev

```

pwspk-plugin-dev.php ✘ template.php ✘ option.php ✘ plugin.php ✘
add_submenu_page('pwspk-options', 'PWS PK Layout', 'PWS PK 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>PWS PK 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 ↑](#)

```
//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>
    ?>
}

}
```

```
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(){
}

?>
```



```
//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');
}

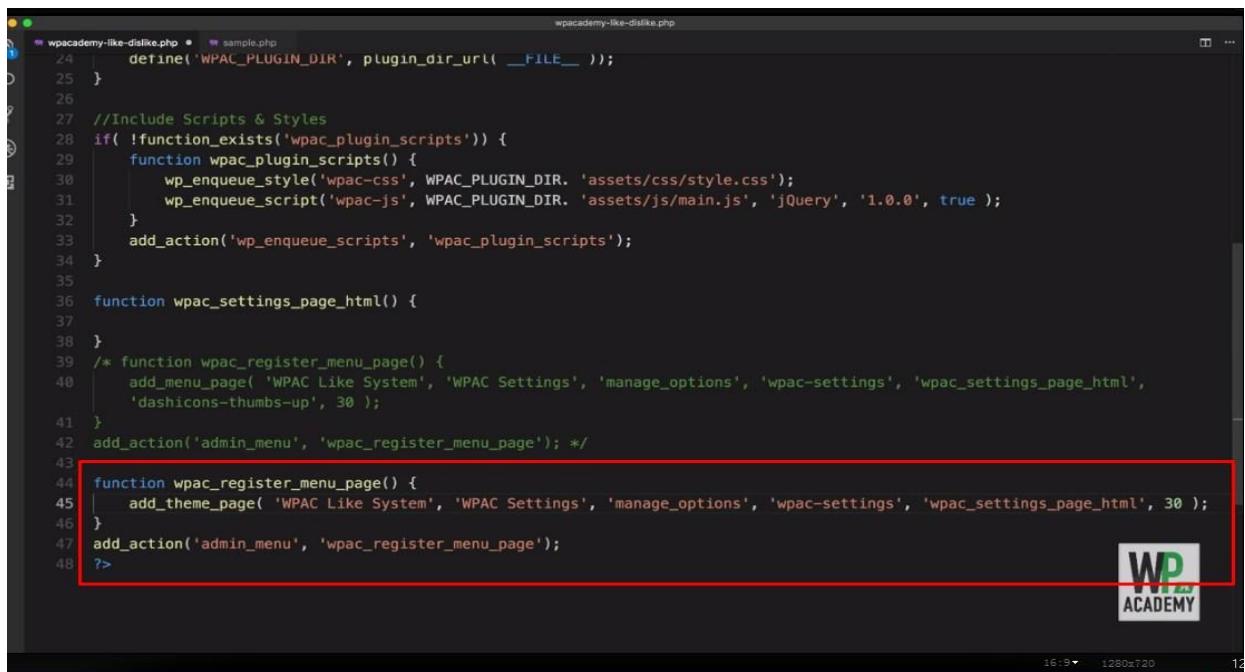
function wpac_settings_page_html() {

}

/* 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');
?>
```

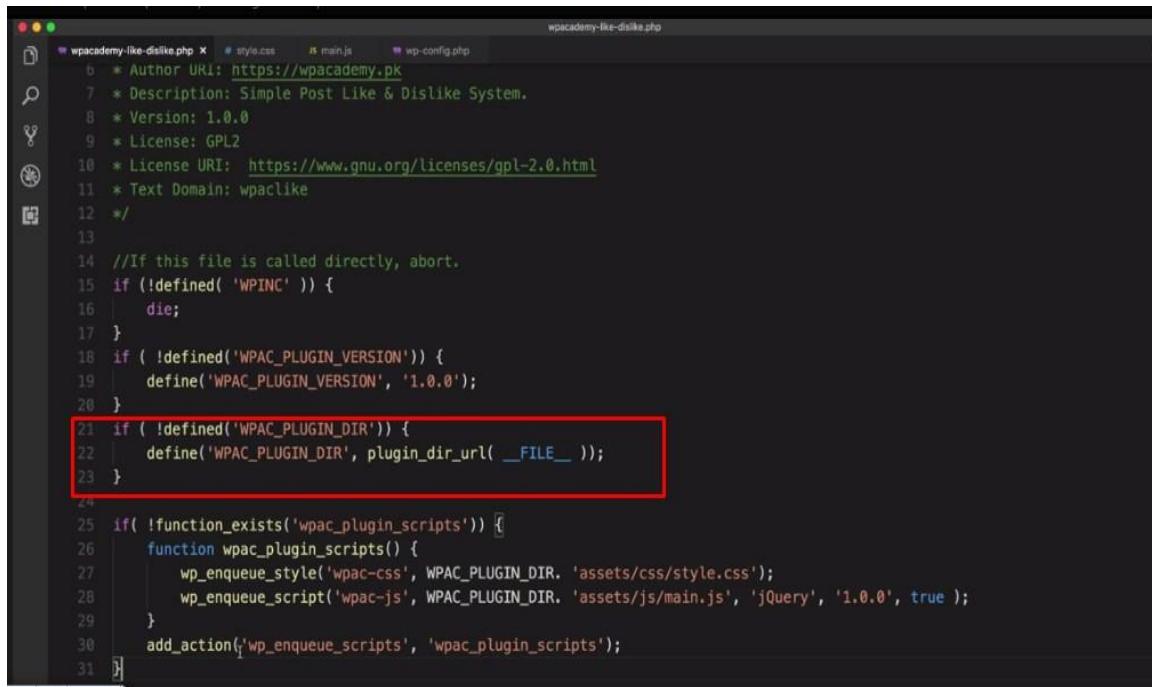




```
wpacademy-like-dislike.php sample.php wpacademy-like-dislike.php
24 |     define('WPAC_PLUGIN_DIR', plugin_dir_url( __FILE__ ));
25 |
26 |
27 //Include Scripts & Styles
28 if( !function_exists('wpac_plugin_scripts') ) {
29     function wpac_plugin_scripts() {
30         wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
31         wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true );
32     }
33     add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
34 }
35
36 function wpac_settings_page_html() {
37 }
38
39 /* function wpac_register_menu_page() {
40     add_menu_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html',
41     'dashicons-thumbs-up', 30 );
42 }
43 add_action('admin_menu', 'wpac_register_menu_page'); */
44
45 function wpac_register_menu_page() {
46     add_theme_page( 'WPAC Like System', 'WPAC Settings', 'manage_options', 'wpac-settings', 'wpac_settings_page_html', 30 );
47 }
48 add_action('admin_menu', 'wpac_register_menu_page');
?>
```



16:9 1280x720 12



```
wpacademy-like-dislike.php style.css main.js wp-config.php wpacademy-like-dislike.php
6 * Author URI: https://wpacademy.pk
7 * Description: Simple Post Like & Dislike System.
8 * Version: 1.0.0
9 * License: GPL2
10 * License URI: https://www.gnu.org/licenses/gpl-2.0.html
11 * Text Domain: wpaclike
12 */
13
14 //If this file is called directly, abort.
15 if (!defined( 'WPINC' )) {
16     die;
17 }
18 if ( !defined('WPAC_PLUGIN_VERSION') ) {
19     define('WPAC_PLUGIN_VERSION', '1.0.0');
20 }
21 if ( !defined('WPAC_PLUGIN_DIR') ) {
22     define('WPAC_PLUGIN_DIR', plugin_dir_url( __FILE__ ));
23 }
24
25 if( !function_exists('wpac_plugin_scripts') ) {
26     function wpac_plugin_scripts() {
27         wp_enqueue_style('wpac-css', WPAC_PLUGIN_DIR. 'assets/css/style.css');
28         wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js', 'jQuery', '1.0.0', true );
29     }
30     add_action('wp_enqueue_scripts', 'wpac_plugin_scripts');
31 }
```

```
wpacademy-like-dislike.php * main.js wp-config.php
12 */
13
14 //If this file is called directly, abort.
15 if (!defined( 'WPINC' )) {
16     die;
17 }
18 if ( !defined('WPAC_PLUGIN_VERSION')) {
19     define('WPAC_PLUGIN_VERSION', '1.0.0');
20 }
21 if ( !defined('WPAC_PLUGIN_DIR')) {
22     define('WPAC_PLUGIN_DIR', plugin_dir_path( __FILE__ ));
23 }
24
25 if( !function_exists('wpac_plugin_scripts')) {
26     function wpac_plugin_scripts() {
27         wp_enqueue_script('wpac-js', WPAC_PLUGIN_DIR. 'assets/js/main.js' );
28     }
29 }
30
31
32
```

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(); ?&gt;"&gt;&lt;?php the_post_thumbnail('medium'); ?&gt;
                &lt;/div&gt;
                &lt;div class="product-title"&gt;
                    &lt;h3&gt;&lt;a href="<?php the_permalink(); ?&gt;"&gt;&lt;?php the_title(); ?&gt;&lt;/a&gt;&lt;/h3&gt;
                &lt;/div&gt;
                &lt;div class="product-price"&gt;
                    &lt;?php echo $product-&gt;get_price_html() ?&gt;
                &lt;/div&gt;
                &lt;div class="product-buttons"&gt;

                &lt;/div&gt;
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;?php
    endwhile;

else :</pre
```

```
//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>
    <?
}

}
```

```
//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



```
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' );
```

```
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');
```

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

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

```
32
33     add_theme_support('menus');
34
35     register_nav_menus(
36         array('top_menu'=>'Top Menu', 'theme')
37     );
```

```
* 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');
```

```
<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
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 scrollly">Learn More</a>
    </div>
</section>
```

```
<?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 scrollly"><?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.</p>
    </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' );
```

```
1 welcome      # style.css      index.php      single.php      page.php      archives.php
2
3 <?php
4 /*
5  * This template is used display single pages
6  */
7 get_header();
8 ?>
9
10
11     <?php if ( have_posts() ) : ?>
12         <?php while ( have_posts() ) : the_post(); ?>
13             <h1><?php the_title() ?></h1>
14             <p><?php the_content();?></p>
15             <?php endwhile; ?>
16         <?php endif; ?>
17
18
19 <?php
20 get_footer();
21 ?>
```

```
# style.css      index.php      home-featured.php      single.php      page.php      wp-config.php      archives.php
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();
```



```
<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>
    </div>
</body>
```

```
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
*/
?>
<!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>
    </div>

```

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

```
index.php
1  index.php ✘ footer.php header.php functions.php page.php single.php wp-config.php archives.php 404.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
1  <?php
2  /*
3  * This template is used display sidebar
4  */
5  ?>
6  <div class="main-sidebar-inner">
7      <?php
8          dynamic_sidebar('main-sidebar');
9      ?>
10 </div>
11
```

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

}

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' );
```

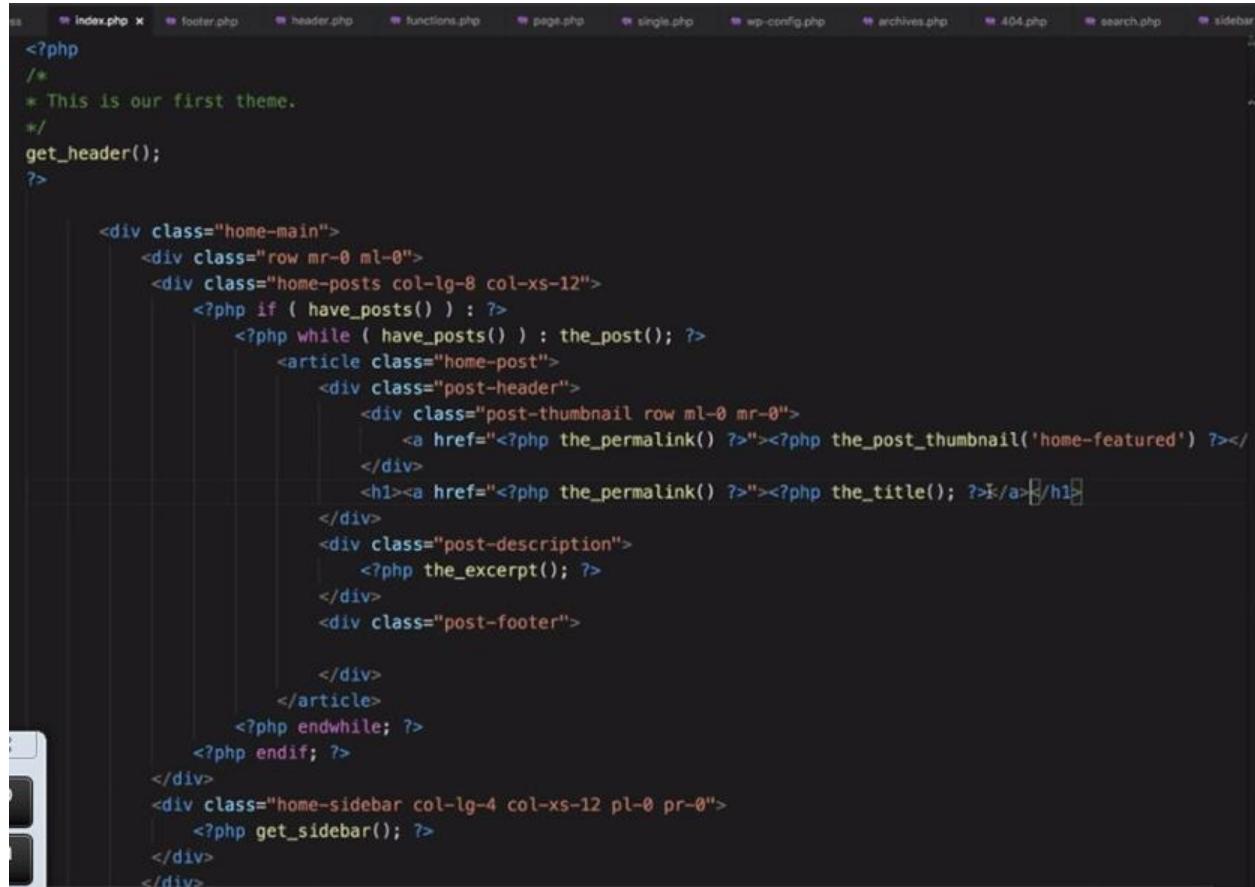
```
css index.php footer.php header.php functions.php page.php single.php wp-config.php archives.php

<?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();
?>
```

```
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');
```



The screenshot shows the code editor interface with the file 'index.php' open. The tab bar at the top lists other files: footer.php, header.php, functions.php, page.php, single.php, wp-config.php, archives.php, 404.php, search.php, and sidebar. The code itself is a template for the homepage, featuring a main content area with a grid of posts and a sidebar.

```
<?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() ?&gt;"&gt;&lt;?php the_post_thumbnail('home-featured') ?&gt;&lt;/a&gt;<br/                        </div>  
                        <h1><a href="php the_permalink() ?&gt;"&gt;&lt;?php the_title(); ?&gt;&lt;/a&gt;&lt;/h1&gt;<br/                    </div>  
                    <div class="post-description">  
                        <?php the_excerpt(); ?>  
                    </div>  
                    <div class="post-footer">  
                        <?php the_post_thumbnail('post-thumbnail') ?>  
                    </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') ?></a>
                            </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">
        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>
```

```

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 x footer.php header.php functions.php page.php single.php wp-config.php archives

<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>
<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">
    <?phg 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 x single.php archives.php 404.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') ?>
    
<div class="page-custom-header">
    <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
    
<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>
```

```
* 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
1 <?php
2 /*
3 * This template is used to display about us page
4 * Template Name: Homepage Template
5 */
6 get_header();
7 ?>
8     <div class="page-custom-header">
9         <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
10        " class="img-fluid">
11    </div>
12    <div class="flex-row ml-0 mr-0 mt-3 text-center">
13        <?php if ( have_posts() ) : ?>
14            <?php while ( have_posts() ) : the_post(); ?>
15                <p><?php the_content(); ?></p>
16            <?php endwhile; ?>
17        <?php endif; ?>
18    </div>
19    <div class="flex-row ml-0 mr-0 mt-3">
20        <div class="col">
21            <div class="section-head"><h3>Latest from Technology</h3></div>
22            <div class="section-content"></div>
23        </div>
24        <div class="col">
25            <div class="section-head"><h3>Latest from Technology</h3></div>
26            <div class="section-content"></div>
27        </div>
28        <?php if ( have_posts() ) : ?>
29            <?php while ( have_posts() ) : the_post(); ?>
30                <p><?php the_content(); ?></p>
31            <?php endwhile; ?>
32        <?php endif; ?>
33    </div>
34
```

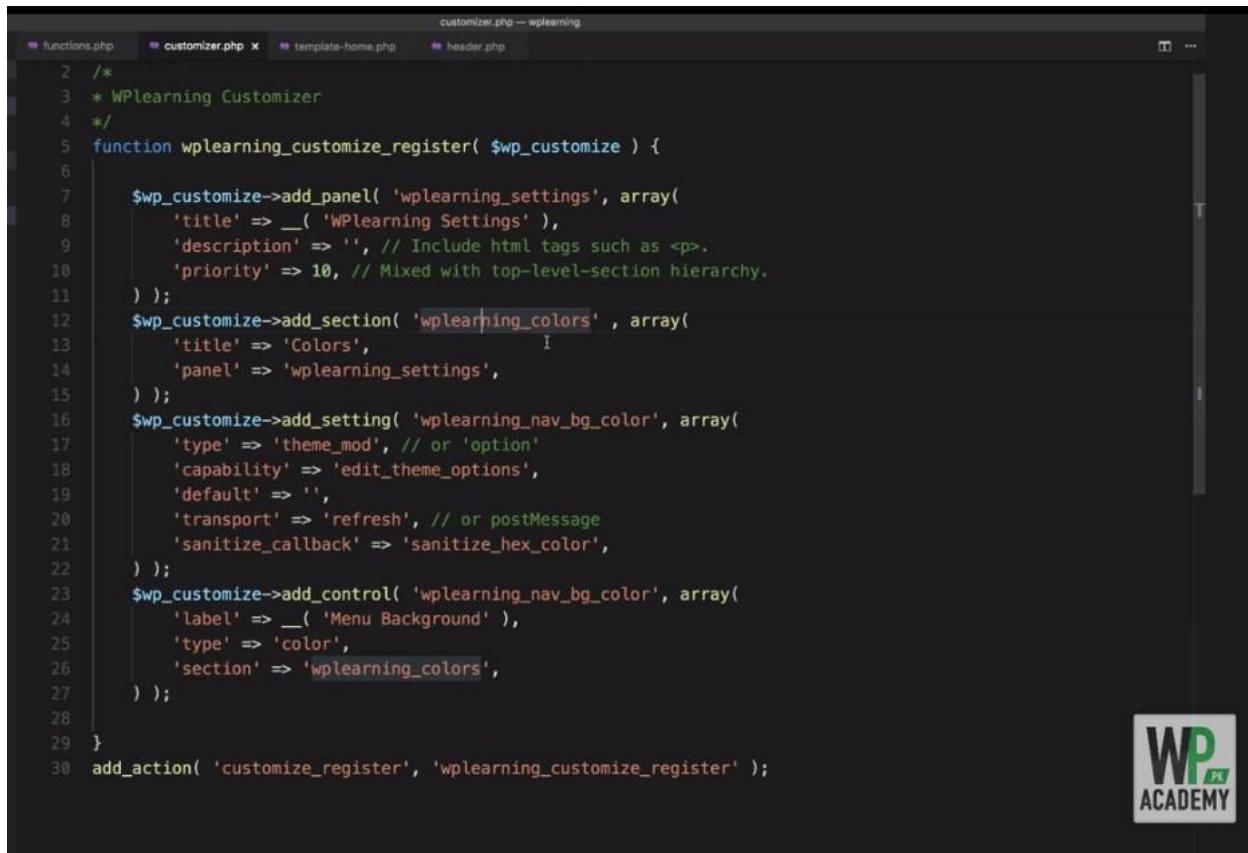


```
template-home.php
14        <?php while ( have_posts() ) : the_post(); ?>
15            <p><?php the_content(); ?></p>
16        <?php endwhile; ?>
17    </div>
18    <div class="home-posts row ml-0 mr-0 mt-5">
19        <div class="col">
20            <div class="section-head"><h3>Latest from Technology</h3></div>
21            <div class="section-content">
22                <?php
23                    $args = array (
24                        'cat'    => 4
25                    );
26                    $tech_posts = new WP_Query( $args );
27                    if ( $tech_posts->have_posts() ) : ?>
28                        <?php while ( $tech_posts->have_posts() ) : $tech_posts->the_post(); ?>
29                        <div class="home-post-row row ml-0 mr-0 mb-3">
30                            <div class="post-thumb col-4 pl-0">
31                                <?php the_post_thumbnail('thumbnail', array('class' => 'img-fluid')) ?>
32                            </div>
33                            <div class="post-title col-8">
34                                <h3><a href="<?php echo get_permalink(get_the_ID()) ?>"><?php the_title() ?></a></h3>
35                                <p><?php the_excerpt() ?></p>
36                            </div>
37
38                            <?php endwhile; ?>
39                            <?php endif; ?>
40                            <?php wp_reset_postdata(); ?>
41                        </div>
42                    <div class="col">
```

```
3 }  
4 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.

```
1 add_action('customize_register','my_customize_register');  
2 function my_customize_register( $wp_customize ) {  
3     $wp_customize->add_panel();  
4     $wp_customize->get_panel();  
5     $wp_customize->remove_panel();  
6  
7     $wp_customize->add_section();  
8     $wp_customize->get_section();  
9     $wp_customize->remove_section();  
10  
11    $wp_customize->add_setting();  
12    $wp_customize->get_setting();  
13    $wp_customize->remove_setting();  
14  
15    $wp_customize->add_control();  
16    $wp_customize->get_control();  
17    $wp_customize->remove_control();  
18 }
```



The screenshot shows a code editor with several tabs at the top: functions.php, customizer.php (which is the active tab), template-home.php, and header.php. The code in the editor is as follows:

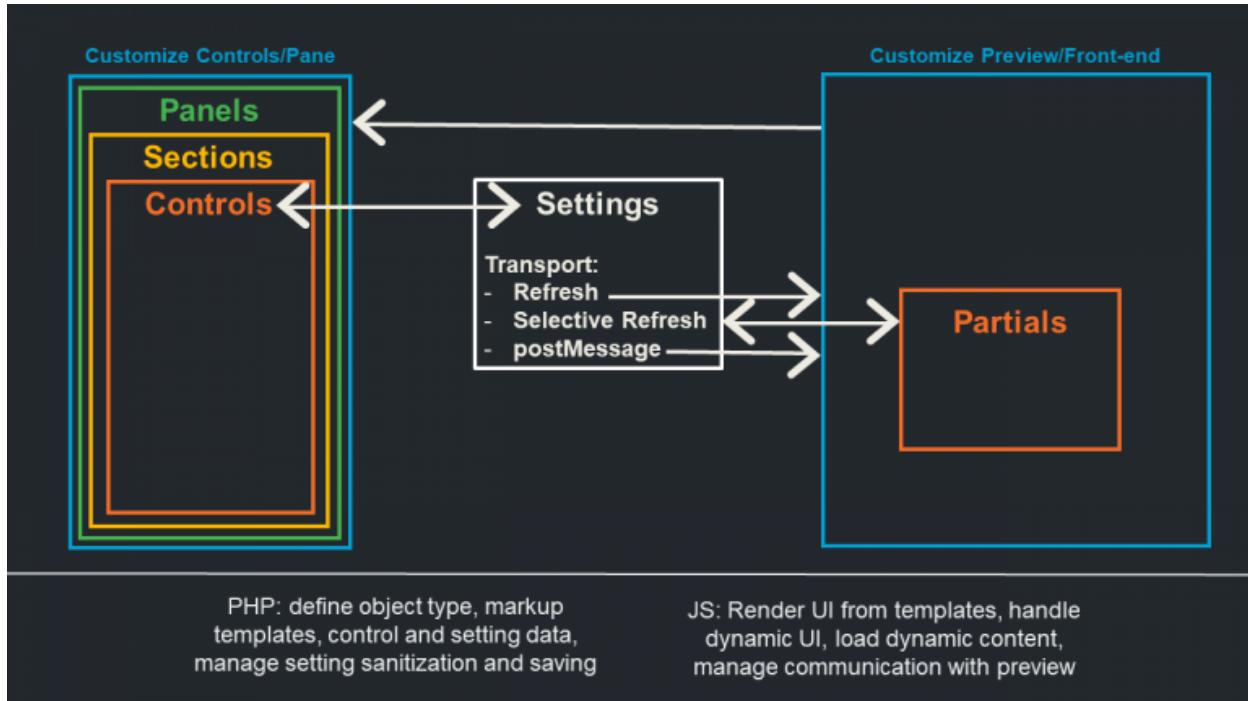
```
2 /*  
3  * WP Learning Customizer  
4 */  
5 function wplearning_customize_register( $wp_customize ) {  
6  
7     $wp_customize->add_panel( 'wplearning_settings', array(  
8         'title' => __( 'WP Learning Settings' ),  
9         'description' => '', // Include html tags such as <p>.  
10        'priority' => 10, // Mixed with top-level-section hierarchy.  
11    ) );  
12    $wp_customize->add_section( 'wplearning_colors' , array(  
13        'title' => 'Colors',  
14        'panel' => 'wplearning_settings',  
15    ) );  
16    $wp_customize->add_setting( 'wplearning_nav_bg_color', array(  
17        'type' => 'theme_mod', // or 'option'  
18        'capability' => 'edit_theme_options',  
19        'default' => '',  
20        'transport' => 'refresh', // or postMessage  
21        'sanitize_callback' => 'sanitize_hex_color',  
22    ) );  
23    $wp_customize->add_control( 'wplearning_nav_bg_color', array(  
24        'label' => __( 'Menu Background' ),  
25        'type' => 'color',  
26        'section' => 'wplearning_colors',  
27    ) );  
28  
29 }  
30 add_action( 'customize_register', 'wplearning_customize_register' );
```

In the bottom right corner of the editor window, there is a logo for "WP ACADEMY".

```

functions.php(customizer.php) wplearning
59     'section' => 'wplearning_labels',
60   ) );
61
62 $wp_customize->add_setting( 'wplearning_home_banner', array(
63   'type' => 'theme_mod', // or 'option'
64   'capability' => 'edit_theme_options',
65   'default' => 'Latest From Technology',
66   'transport' => 'refresh', // or postMessage
67   'sanitize_callback' => 'esc_attr',
68 ) );
69 $wp_customize->add_control( 'wplearning_home_banner', array(
70   'label' => __( 'Show/Hide Home Banner' ),
71   'type' => 'radio',
72   'section' => 'wplearning_misc',
73   'choices' => array (
74     'yes' => 'Yes',
75     'no' => 'No'
76   )
77 );
78 }
79 add_action( 'customize_register', 'wplearning_customize_register' );

```



```
functions.php  customizer.php  style.css  template-home.php  header.php
1 <?php
2 /*
3 * This template is used to display about us page
4 * Template Name: Homepage Template
5 */
6 get_header();
7 ?>
8     <div class="page-custom-header">
9         <?php
10             $banner_status = get_theme_mod('wplearning_home_banner', 'yes');
11         ?>
12         <?php if($banner_status == "yes") : ?> I
13             <?php $img_url = get_the_post_thumbnail_url(get_the_ID(), 'full') ?>
14             <img src=<?php echo $img_url ?>" alt=<?php echo get_the_title() ?>" class="img-fluid">
15         <?php endif; ?>
16     </div>
17     <div class="flex-row ml-0 mr-0 mt-3 text-center">
18         <div class="home-main-content">
19             <h1>WHO WE ARE</h1>
20             <p>
21                 <?php if ( have_posts() ) : ?>
22                     <?php while ( have_posts() ) : the_post(); ?>
23                         <p><?php the_content(); ?></p>
24                     <?php endwhile; ?>
25                 <?php endif; ?>
26             </p>

```

```
style.css  template-home.php  functions.php  services.php
20             <?php endif; ?>
21         </p>
22
23         </div>
24     </div>
25     <div class="home-services row ml-0 mr-0 mt-5">
26         <?php
27             $args = array (
28                 'post_type'    => 'service',
29                 'posts_per_page' => 3,
30                 'order'        => 'ASC'
31             );
32             $tech_posts = new WP_Query( $args );
33             if ( $tech_posts->have_posts() ) : ?>
34                 <?php while ( $tech_posts->have_posts() ) : $tech_posts->the_post(); ?>
35                 <div class="home-service-col col-4">
36                     <div class="service-thumb col-12">
37                         <?php the_post_thumbnail('medium', array('class' => 'rounded')) ?>
38                     </div>
39                     <div class="service-title col-12">
40                         <h3><a href=<?php echo get_permalink(get_the_ID()) ?>"<?php the_title() ?></a></h3>
41                         <p><?php the_excerpt() ?></p>
42                     </div>
43
44                     <?php endwhile; ?>
45                 <?php endif; ?>
46                 <?php wp_reset_postdata(); ?>
47             </div>

```



```
template-home.php
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```



```
<?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>
</div>
<div class="home-carousel row ml-0 mr-0 mt-5">
<div class="owl-carousel">
<?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(); ?>
<div>
    <a href=<?php the_permalink() ?>><?php the_post_thumbnail( 'medium' ) ?></a>
</div>

    <?php endwhile; ?>
<?php endif; ?>
<?php wp_reset_postdata(); ?>
</div>
</div>
<div class="home-posts row ml-0 mr-0 mt-5 pr-0">
<div class="col pl-0">
    <div class="section-head"><h3><?php echo get_theme_mod('wplearning_featured_block_1', 'Latest Project Technology') ?></h3></div>
```

```
url-metabox.php
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```



```
<?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 ) {
```

```
template-home.php  projects.php  single-service.php  functions.php  url-metabox.php  services.php  my-js.js
<?php wp_reset_postdata(); ?>
</div>
<div class="home-carousel row ml-0 mr-0 mt-5">
    <div class="owl-carousel">
        <?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(); ?>
                <div>
                    <?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' ) ?></a>
                </div>

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

```
+2 template-home.php
50
51     <div class="course-box">
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             <div class="blog-item">
65                 </div>
66
67             <?php } ?>
```

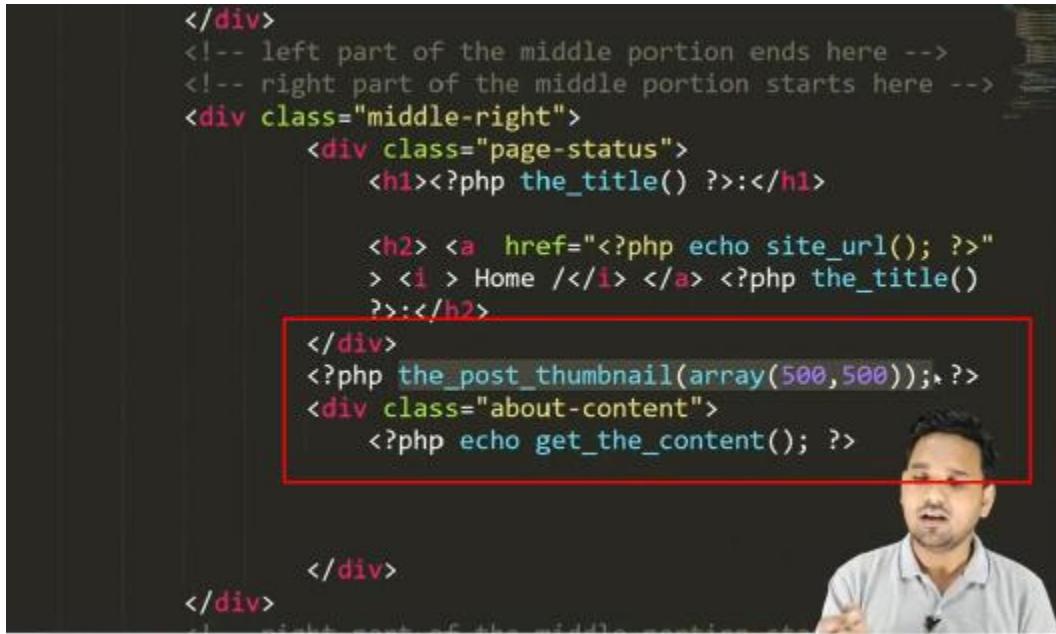
```
<!DOCTYPE html>
<html>
<head>
<title><?php bloginfo('name'); ?> <?php
    wp_title() ?> <?php if (is_front_page()) {
        echo "|"; bloginfo('description'); } ?> </
    title>

<link rel="stylesheet" type="text/css" href=
<?php echo get_template_directory_uri(); ?> /css/
style.css"/>
<?php wp_head(); ?>
</head>
```



Thumbnail Image on WordPress Theme | WordPress Theme Development

```
Thumbnail Image on WordPress Theme | WordPress Theme Development
1  page.php
2
3  <?php get_header(); ?>
4
5  <?php $imagepath= wp_get_attachment_image_src
6  (get_post_thumbnail_id(),'large');
7
8  <?php if($imagepath){ ?>
9      
11
12  <?php } ?>
13
14  <?php get_footer(); ?>
```



```
</div>
<!-- left part of the middle portion ends here --&gt;
<!-- right part of the middle portion starts here --&gt;
&lt;div class="middle-right"&gt;
    &lt;div class="page-status"&gt;
        &lt;h1&gt;&lt;?php the_title() ?&gt;&lt;/h1&gt;

        &lt;h2&gt; &lt;a href="&lt;?php echo site_url(); ?&gt;"&gt; &lt;i&gt; Home /&lt;/i&gt; &lt;/a&gt; &lt;?php the_title() ?&gt;&lt;/h2&gt;
    &lt;/div&gt;
    &lt;?php the_post_thumbnail(array(500,500)); ?&gt;
    &lt;div class="about-content"&gt;
        &lt;?php echo get_the_content(); ?&gt;
    &lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;</pre>
```

Thumbnail Image with tag

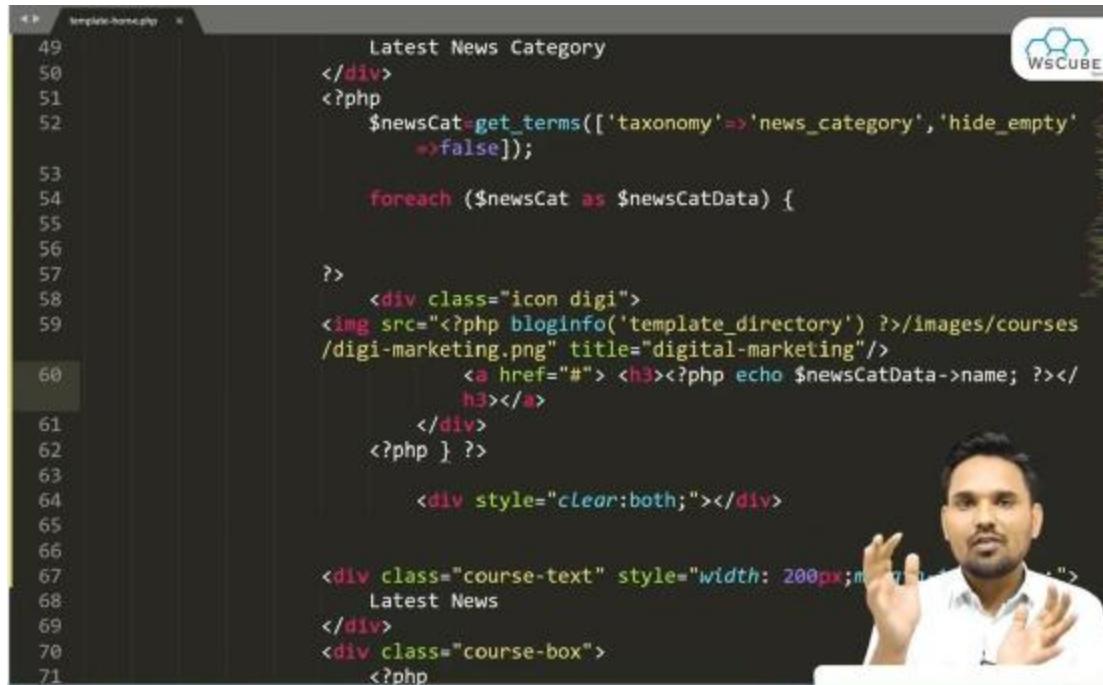
```
the_post_thumbnail(); // Without parameter -> Thumbnail
the_post_thumbnail( 'thumbnail' ); // Thumbnail (default 150px x 150px max)
the_post_thumbnail( 'medium' ); // Medium resolution (default 300px x 300px max)
the_post_thumbnail( 'large' ); // Large resolution (default 1024px x 1024px max)
the_post_thumbnail( 'full' ); // Original image resolution (unmodified)
the_post_thumbnail( array( 100, 100 ) ); // Other resolutions (height, width)
```

Get Image Path

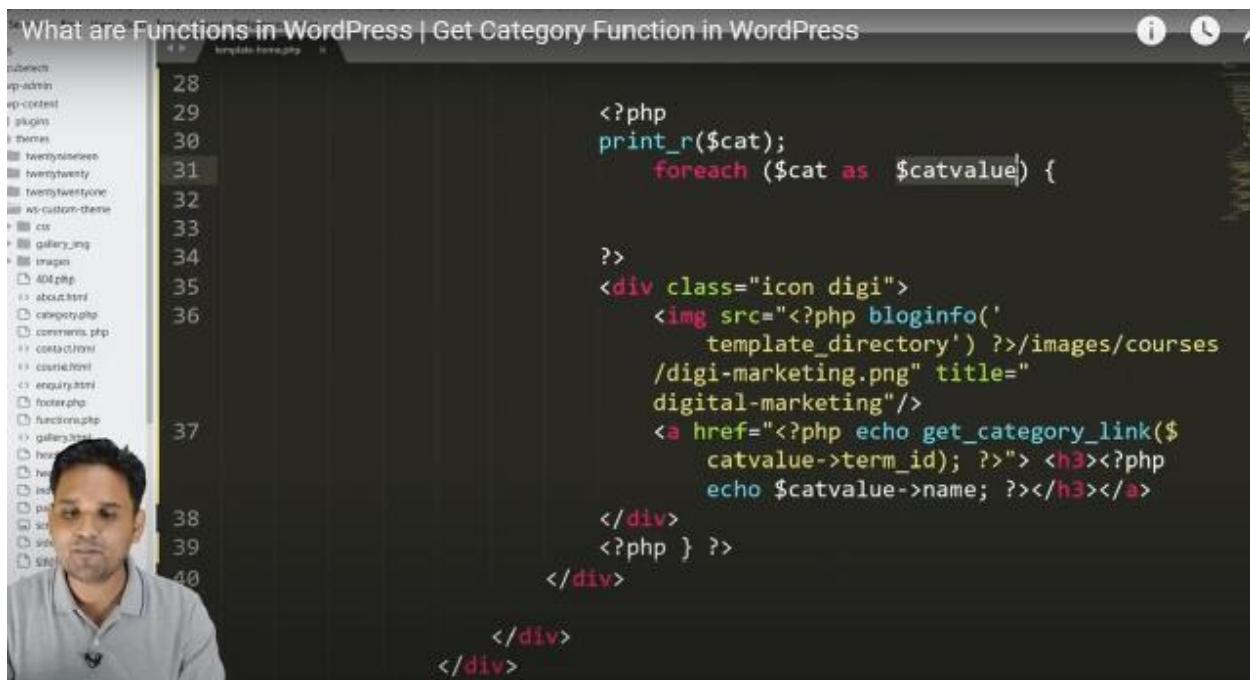
```
wp_get_attachment_image_src( get_post_thumbnail_id(), 'large');
```

```
1 <div class="container mt-5 py-5">
2   <div class="row">
3     <?php
4       $args = array(
5         'post_type' => 'post',
6         'posts_per_page' => 6,
7         'tax_query' => array(
8           array(
9             'taxonomy' => 'category',
10            'field' => 'term_id',
11            'terms' => 4
12          )
13        ),
14        'meta_query' => array(
15          array(
16            'key' => 'country',
17            'value' => 'India',
18            'compare' => 'LIKE'
19          )
20        )
21      );
22      query_posts($args);
23      while(have_posts()) : the_post();
```

```
16      $args2 = array();
17      $args3 = array();
18      $value = apply_filters('my_filter_hook', $args1, $args2, $args3);
19      print_r($value) /* ?>
20
21      <div class="container mt-5 py-5">
22        <div class="row">
23          <?php
24            $args = array(
25              'post_type' => 'post',
26              'posts_per_page' => 6,
27              'tax_query' => array(
28                array(
29                  'taxonomy' => 'category',
30                  'field' => 'term_id',
31                  'terms' => 4
32                )
33              )
34            );
35            query_posts($args);
36            while(have_posts()) : the_post();
37            ?>
38            <div class="col-md-4">
39              <?php the_post(); ?>
```



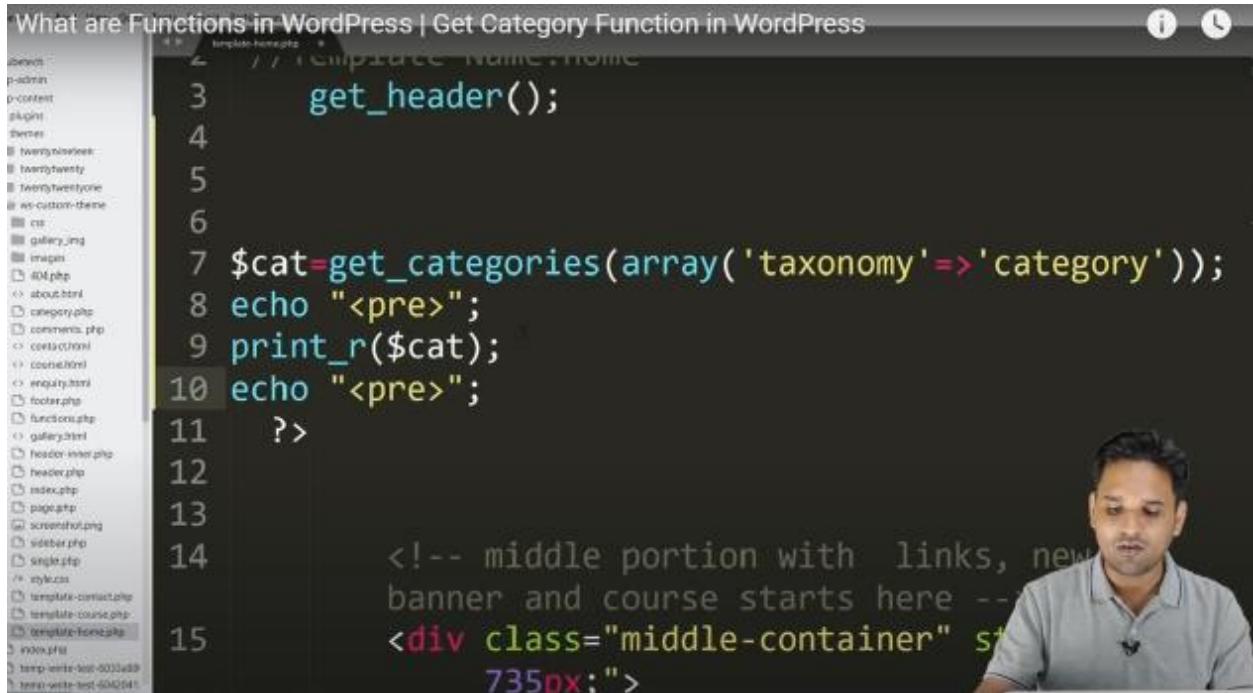
```
49             Latest News Category
50         </div>
51         <?php
52             $newsCat=get_terms(['taxonomy'=>'news_category','hide_empty'
53             =>false]);
54
55             foreach ($newsCat as $newsCatData) {
56
57                 ?>             <div class="icon digi">
58                 
60                     <a href="#"> <h3><?php echo $newsCatData->name; ?><
61                         h3></a>
62                     </div>
63             <?php } ?>
64
65             <div style="clear:both;"></div>
66
67             <div class="course-text" style="width: 200px; margin-left: 10px;">
68                 Latest News
69             </div>
70             <div class="course-box">
71                 <?php
```



What are Functions in WordPress | Get Category Function in WordPress

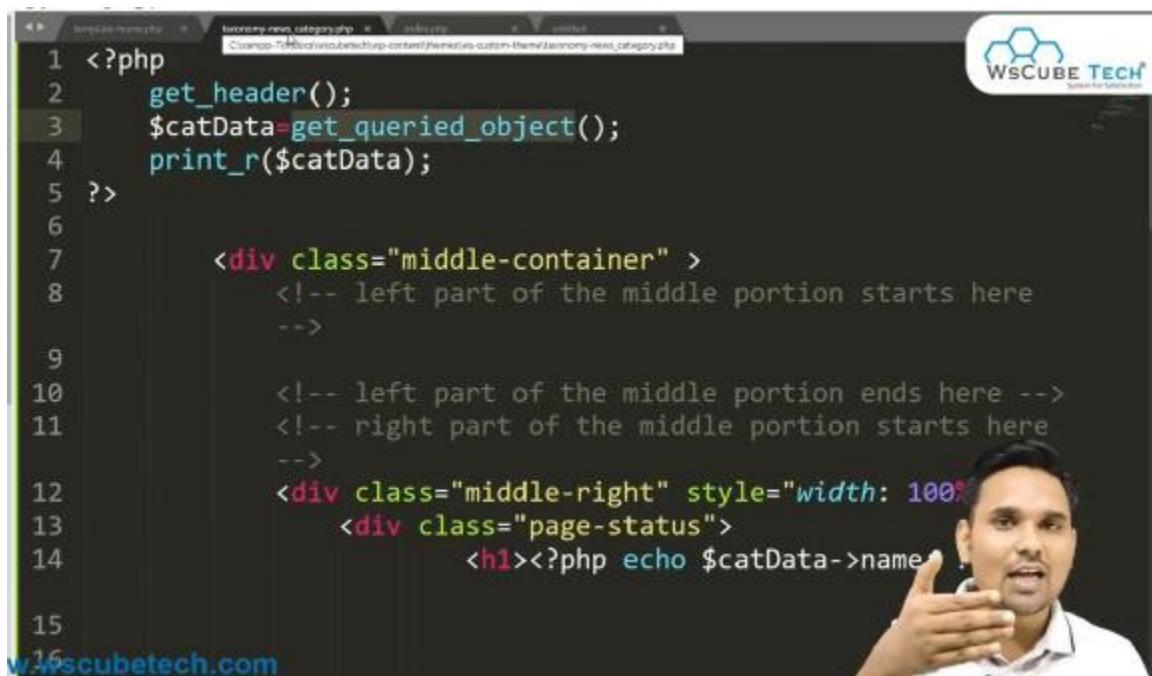
```
28
29
30
31             <?php
32                 print_r($cat);
33
34                 foreach ($cat as $catvalue) {
35
36
37             ?>             <div class="icon digi">
38                 
42                     <a href="<?php echo get_category_link($
43                         catvalue->term_id); ?>"> <h3><?php
44                         echo $catvalue->name; ?></h3></a>
45                     </div>
46                 <?php } ?>
47             </div>
48
49         </div>
50     </div>
```

What are Functions in WordPress | Get Category Function in WordPress



```
template-home.php
get_header();
$cat=get_categories(array('taxonomy'=>'category'));
echo "<pre>";
print_r($cat);
echo "<pre>";
?>

<div class="middle-container" style="width: 735px;">
```



```
taxonomy-news-category.php
<?php
get_header();
$catData=get_queried_object();
print_r($catData);
?>
<div class="middle-container" >
<!-- left part of the middle portion starts here
--&gt;
<!-- left part of the middle portion ends here --&gt;
<!-- right part of the middle portion starts here
--&gt;
&lt;div class="middle-right" style="width: 100%"&gt;
&lt;div class="page-status"&gt;
&lt;h1&gt;&lt;?php echo $catData-&gt;name?&gt;</pre>

WScubeTech.com


```

How to Use Custom Taxonomy in get_terms Function | WordPress Theme Development



```

49             Latest News Category
50         </div>
51     <?php
52         $newsCat = get_terms(['taxonomy' => 'news_category', 'hide_empty'
53                               => false]);
54
55         foreach ($newsCat as $newsCatData) {
56
57             ?>
58                 <div class="icon digi">
59                     
61                         <a href="#"><?php echo $newsCatData->name; ?><
62                           h3></a>
63                     </div>
64             <?php } ?>
65
66             <div style="clear:both;"></div>
67
68             <div class="course-text" style="width: 200px; margin-right: 20px;">
69                 Latest News
70             </div>
71             <div class="course-box">
72                 <?php

```



template-category.php

```

5     get_header(); ?>
6
7     <style type="text/css">
8         .wp-pagenavi {
9             width: 50%;
10            margin: 0 auto;
11            margin-top: 20px;
12        }
13     </style>
14
15     <div id="primary" class="content-area">
16         <main id="main" class="site-main" role="main">
17
18             <?php
19             $paged = (get_query_var( 'paged' )) ? get_query_var( 'paged' ) : 1;
20             $args = array(
21                 'post_type' => 'post',
22                 'post_status' => 'publish',
23                 'category_name' => 'wordpress',
24                 //'cat' => '2,3',
25                 'posts_per_page' => 5,
26                 'paged' => $paged,
27                 );
28             $arr_posts = new WP_Query( $args );
29
30             if ( $arr_posts->have_posts() ) :
31
32                 while ( $arr_posts->have_posts() ) :
33                     $arr_posts->the_post();
34                     ?>
35                     <article id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
36                         <?php
37

```

```
my-plugin.php X
wp-content > plugins > my-plugin > my-plugin.php
306     add_action('wp_logout', 'redirect_after_logout');
307
308
309     function my_meta_fields(){
310         ?>
311         <label for="my-meta-field1">My Meta Field 1</label>
312         <input type="text" name="my-meta-field1" id="my-meta-field1" />
313         <?php
314     }
315
316     function add_my_meta_box(){
317         add_meta_box('my-meta-box', 'My Meta Box', 'my_meta_fields', 'cars');
318     }
319     add_action('add_meta_boxes', 'add_my_meta_box');
```

Checklist of Technical SEO

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



Looking for Digital Marketing Training? Call us at : +91 9269698122 or visit www.wscubetech.com

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



woocommerce.php — wpdev

```

4
5  * @package Understrap
6  */
7
8 // Exit if accessed directly.
9 defined( 'ABSPATH' ) || exit;
10
11 add_action( 'after_setup_theme', 'understrap_woocommerce_support' );
12 if ( ! function_exists( 'understrap_woocommerce_support' ) ) {
13 /**
14 * Declares WooCommerce theme support.
15 */
16 function understrap_woocommerce_support() {
17     add_theme_support( 'woocommerce' );
18
19     // Add Product Gallery support.
20     add_theme_support( 'wc-product-gallery-lightbox' );
21     add_theme_support( 'wc-product-gallery-zoom' );
22     add_theme_support( 'wc-product-gallery-slider' );
23
24     // Add Bootstrap classes to form fields.
25     add_filter( 'woocommerce_form_field_args', 'understrap_wc_form_field_args', 10, 3
        ).
```

Line 1, Column 1 Tab Size: 4 PHP

1) Overriding Template File • woocommerce.php • archive-product.php •

```

<?php
function pws_pk_woocommerce_product_loop_title_classes($classes) {
    return $classes . ' text-center';
}
add_action('init', 'pws_pk_woocommerce_features');
function pws_pk_woocommerce_features(){
    add_filter('woocommerce_product_loop_title_classes',
        'pws_pk_woocommerce_product_loop_title_classes' );
    remove_action('woocommerce_before_shop_loop','woocommerce_result_count',20);
    remove_action('woocommerce_before_shop_loop','woocommerce_catalog_ordering',20);
}
```

```
<?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
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">
            <?php the_content(); ?>
```

A screenshot of a code editor window titled "page-how-to-use-nonce.php". The code is a PHP script for a WordPress page template. It includes a security check for a nonce, loops through posts, and displays the title and a link to a settings page.

```
1 <?php
2 if ( isset($_POST['submit']) ) {
3     if ( wp_verify_nonce($_POST['_nonce'], 'submit-user') ) {
4         //safe to submit data
5     } else {
6         die('security check!');
7     }
8 }
9
10 get_header(); ?>
11
12 <div id="primary" class="content-area">
13     <main id="main" class="site-main" role="main">
14
15     <?php
16     // Start the loop.
17     while ( have_posts() ) : the_post();
18     ?>
19         <header class="entry-header">
20             <?php the_title( '<h1 class="entry-title">', '</h1>' ); ?>
21         </header><!-- .entry-header -->
22         <div class="entry-content">
23             <a href="?php echo wp_nonce_url(get_bloginfo('url'), '/settings', 'page-settings', 'my-nonce') ?>">Go to settings page</a>
24         </div>
25     <?php
26     // End the loop.
27 endwhile;
28 ?>
```

```
<?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:

```
1 | register_uninstall_hook(__FILE__, 'pluginprefix_function_to_run');
```

```
my-plugin.php      uninstall.php ●      index.php
wp-content > plugins > my-plugin > uninstall.php
1  <?php
2
3  if(!defined('WP_UNINSTALL_PLUGIN')){
4      header("Location: /youtube");
5      die();
6  }
7
8  /
```

```
'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');
```

The screenshot shows a code editor with two tabs open: "my-plugin.php" and "uninstall.php". The "my-plugin.php" tab is active, displaying the following PHP code:

```
wp-content > plugins > my-plugin > my-plugin.php X
1  <?php
2
3  if(!defined('WP_UNINSTALL_PLUGIN')){
4      header("Location: /youtube");
5      die();
6  }
7
8  global $wpdb, $table_prefix;
9
10 $wp_emp = $table_prefix.'emp';
11
12 $q = "DROP TABLE `{$wp_emp}`;";
13 $wpdb->query($q);
```

The screenshot shows a code editor displaying the "my_shortcode" function definition:

```
44 function my_shortcode($atts){
45
46     return 'This is from Shortcode'. $atts['msg'];
47 }
48 add_shortcode('youtube', 'my_shortcode');
49
```

The screenshot shows a code editor displaying the "my_shortcode" function definition with a modification:

```
function my_shortcode($atts){
    $atts = shortcode_atts(array(
        'msg'    => 'I am Good'
    ), $atts);

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

```
14 function my_shortcode($atts){  
15     $atts = array_change_key_case($atts, CASE_LOWER);  
16  
17     $atts = shortcode_atts(array(  
18         'msg'    => 'I am Good',  
19         'note'   => 'default'  
20     ), $atts);  
21  
22     ob_start();  
23     ?>  
24     <h1>Youtube</h1>  
25     <?php  
26     $html = ob_get_clean();  
27  
28     return $html;  
29 }  
30 add_shortcode('youtube', 'my_shortcode');  
31
```

```
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');
```

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

```
$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 current 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_shortcode('my-register-form', 'my_register_form');
```

