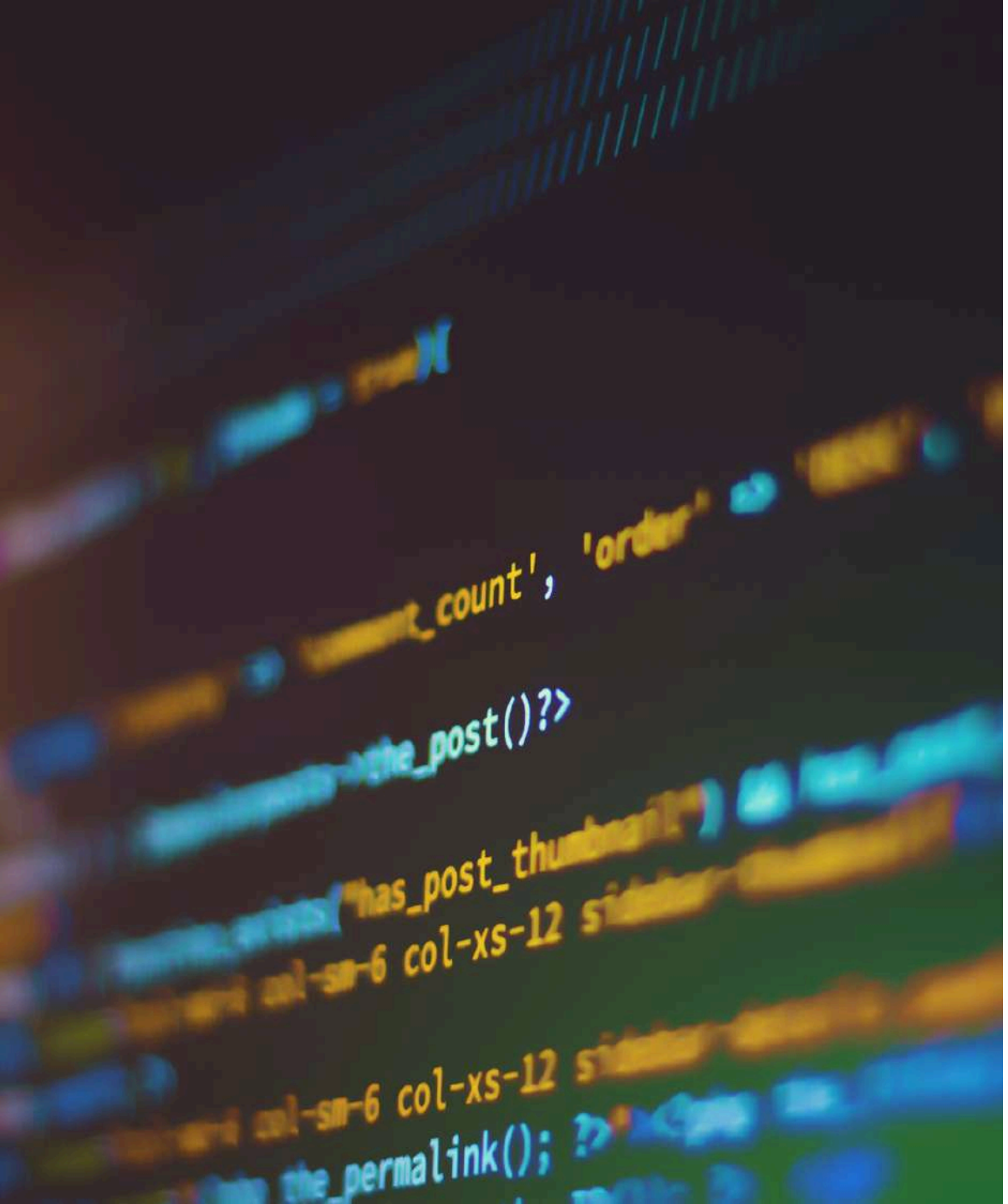


Advanced WordPress

Course Objectives

Understanding Core Concepts

Gain a comprehensive understanding of WordPress architecture, including core **file structures, database management, and the template hierarchy**.

Custom Development

Learn how to manage **custom post types** and **taxonomies** to **organize content** better and enhance user experience.

Theme and Plugin Development

Explore advanced techniques for **developing custom themes and plugins**, including implementing **WordPress hooks, actions, and filters.**

Performance Optimization

Discover methods for optimizing website performance using **caching solutions**, as well as best practices for code efficiency.

Security Practices

Understand the importance of security in WordPress and learn advanced **security measures** to protect websites from vulnerabilities.

Headless CMS Approach

Explore the concept of using WordPress as a headless CMS, **integrating it with modern JavaScript** frameworks like React, Vue, or Angular.

Debugging and Troubleshooting

Learn how to **enable debugging** features in WordPress, identify common issues, and employ **effective troubleshooting** techniques.

WordPress

What is WordPress?

WordPress is an open-source **content management system** (CMS) that allows users to **create, manage, and customize websites** and blogs easily.It's built with PHP and uses a MySQL database.

When Was WordPress Invented and By Whom?

WordPress was created in **2003** by **Matt Mullenweg** and **Mike Little**.It started as a simple blogging platform but has since evolved into a powerful website-building tool.

Purpose of WordPress

To make website creation **accessible for everyone**, from **personal blogs** to **business websites** and **eCommerce** stores.It's designed to be user-friendly, customizable, and extendable, supporting various types of content through themes and plugins.

The Evolution of WordPress

2003

WordPress was created by Matt Mullenweg and Mike Little as a fork of the blogging platform b2/cafelog.

2004

WordPress 1.0 was released, introducing the core features that would shape the future of the platform.

2005

WordPress 1.5 introduced the concept of custom post types, expanding the platform's capabilities beyond just blogging.

2008

WordPress 2.7 introduced the revolutionary automatic updates feature, making it easier for users to keep their sites up to date.

2010

WordPress 3.0 introduced the Customizer, allowing users to preview and customize their site in real-time.

2015

WordPress 4.4 introduced the responsive images feature, improving the platform's performance on mobile devices.

2020

WordPress 5.5 introduced the block editor, revolutionizing the way users create and manage content on their sites.

Tools to Setup Wordpress locally

Docker

A platform that enables developers to automate the deployment of applications in lightweight, portable containers, providing isolated environments and easy replication.

Local by Flywheel

A user-friendly local development tool specifically designed for WordPress, with an intuitive interface for setting up local sites, built-in tools for managing SSL, backups, and migrations.

WP-CLI

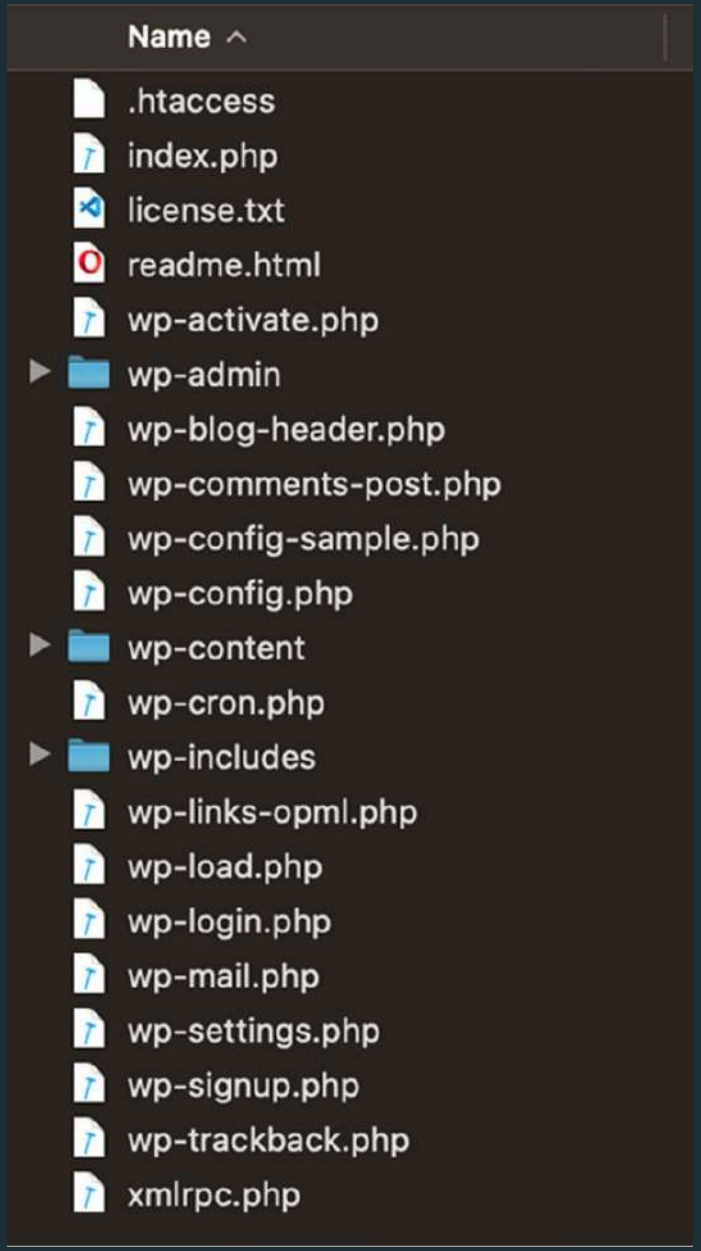
A command-line interface for managing WordPress installations, allowing developers to perform various tasks such as updating plugins, managing users, and importing/exporting data with speed and efficiency.

XAMMP

Is a popular local server environment that provides developers with the tools needed to run WordPress on their own computers. It includes Apache, MySQL (or MariaDB), PHP, and Perl, enabling developers to easily set up a local WordPress

WordPress File and Architecture

The WordPress architecture encompasses the core file structure, database structure, and template hierarchy that work together to power the content management system.



wp-content Directory

Purpose

The wp-content directory is where all **user-generated content** and **custom files are stored**. It's the most frequently accessed directory when customizing WordPress.

The wp-content directory is the heart of WordPress customization, allowing you to manage themes, plugins, and media files to enhance your website's functionality and appearance.

Themes

The themes subdirectory contains all theme files. Each theme is stored in its own folder, allowing you to modify existing themes or create new ones.

Plugins

The plugins subdirectory holds all plugins that extend WordPress functionality. Like themes, each plugin resides in its own folder within this directory.

Uploads

The uploads subdirectory stores media files, such as images, videos, and documents. By default, it organizes files by year and month.

wp-includes Directory

Purpose

The wp-includes directory contains the **core WordPress code**, providing the underlying framework that powers the CMS.

Understanding the key files and functions in the wp-includes directory is crucial for debugging and creating advanced WordPress customizations.

functions.php

This file defines essential functions used throughout WordPress, such as those responsible for database interaction, HTTP requests, and more.

class-*.php Files

These files define the various classes used by WordPress, including the WP_Query class and others that handle core functionality.

default-widgets.php

This file defines the default WordPress widgets, like the text widget and recent posts widget.

wp-admin Directory

Purpose

The wp-admin directory powers the **WordPress admin dashboard**, containing all the files necessary for running the backend interface where you manage your site's content, settings, and appearance.

Understanding the purpose and key components of the wp-admin directory is crucial for customizing the WordPress admin experience and creating advanced dashboard features.

To access the dashboard: https://www.your-domain/wp-admin

admin.php

This is the main file for **loading and routing admin** requests. When you access the dashboard, this file processes requests and loads appropriate pages.

menu.php

This file defines the admin menu items, such as Dashboard, Posts, and Settings, which are displayed in the WordPress admin interface.

CSS and JS Folders

These folders contain the CSS and JavaScript files used by the WordPress admin interface, managing the look and feel of the dashboard.

Other Key PHP Files

- ## wp-config.php

  This is one of the **most critical files in a WordPress** installation. It contains **database connection information, site settings, and configuration options**. It's essential to secure this file, as it holds **sensitive information**.

- ## wp-load.php

  This file loads the WordPress environment and core files, setting up the necessary framework for executing WordPress functions.

- ## wp-blog-header.php

  This file is responsible for loading the WordPress environment for the front-end of the site.

- ## index.php

  The main template file that serves as the fallback for WordPress. If no other template files are found for rendering a page, WordPress uses this file.

- ## .htaccess

  This is not a PHP file but a configuration file for the Apache web server, used to manage redirects, permalinks, and other server settings.

Database Structure and Table Relationships

DB files	Purpose
wp_options	Stores site-wide settings and options
wp_posts	Contains posts, pages, and Custom post type
wp_postmeta	Stores metadata for posts (custom fields)
wp_users	Holds user information and credentials
wp_usermeta	Stores metadata for users
wp_comments	Holds comments made on posts and pages
wp_commentmeta	Stores metadata for comments.
wp_terms	Stores categories, tags, and custom taxonomies
wp_termmeta	Holds metadata for terms in taxonomies
wp_term_relationships	Links terms to posts
wp_term_taxonomy	Defines types of terms (category, tag)
wp_links	Contains links created by the user (mostly legacy)

WordPress Theme

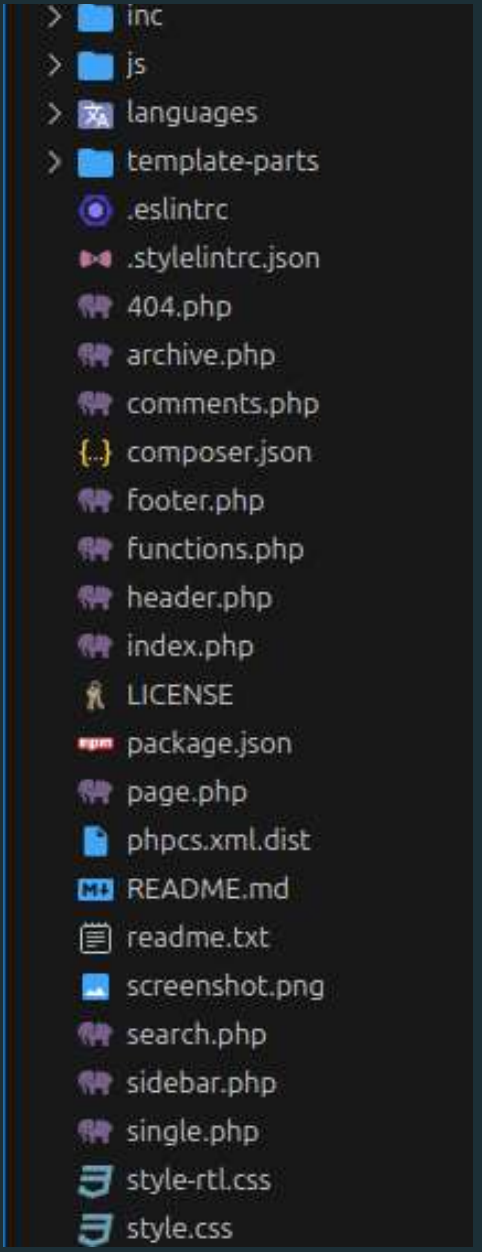
Developing WordPress themes can be approached in various ways, depending on your skills, project requirements, and preferred methodologies. Here are some popular methods to consider:

From Scratch
Create a theme from scratch using HTML, CSS, and PHP. This method provides complete control over design and functionality but requires a solid understanding of WordPress architecture and coding standards.

Theme Frameworks
Utilize a theme framework (like Genesis or Underscores) that provides a foundation of pre-built functionality.

Child Themes
Create a child theme based on an existing theme (like a parent theme) to modify its appearance or functionality without altering the original code. This approach allows for easier updates to the parent theme while retaining custom changes.

Page Builders
Use page builder plugins (like Elementor, Beaver Builder, or WPBakery) that allow for drag-and-drop theme design. This method is user-friendly and suitable for those with limited coding skills.



The WordPress Theme Hierarchy

Overview

The WordPress template hierarchy is a system that **determines which template file(s) to use when displaying a particular type of page**. This hierarchy allows themes to load **different** templates for various content types, such as single posts, pages, category archives, and the homepage.

Order of Templates

When a page is requested, WordPress follows a specific hierarchy to find the most appropriate template file to load. This includes templates for **single posts, pages, category archives, tag archives, author archives, search results, 404 error pages, and homepages**.

Single Post Templates

WordPress will look for the most specific single post template, starting with **single-{post_type}-{slug}.php**, then **single-{post_type}.php**, **single.php, singular.php**, and finally **index.php**.

Page Templates

Page templates follow a similar order, starting with **page-{slug}.php, page-{id}.php, page.php, singular.php**, and **index.php**.

Archive Templates

For category, tag, and author archives, WordPress will look for templates in the order of **{archive_type}-{slug}.php, {archive_type}-{id}.php, {archive_type}**.php, archive.php, and index.php.

Other Templates

WordPress also has specific templates for search results (**search.php, index.php**) and the 404 error page (**404.php, index.php**), as well as the homepage (**home.php or front-page.php, index.php**).


Archive Page
Author Archive
author-$nicename.php
author-$id.php
author.php
Category Archive
category-$slug.php
category-$id.php
category.php
Custom Post Type Archive
archive-$posttype.php
Custom Taxonomy Archive
taxonomy-$taxonomy-$term.php
taxonomy-$taxonomy.php
taxonomy.php
Date Archive
Year Archive
Month Archive
Day Archive
date.php
Tag Archive
tag-$slug.php
tag-$id.php
tag.php
archive.php
index.php
Singular Page
Single Post Page
Attachment Post
$mimetype-$subtype.php
$subtype.php
$mimetype.php
attachment.php
Custom Post
If selected: $custom.php
single-$posttype-$slug.php
single-$posttype.php
Blog Post
If selected: $custom.php
single-post.php
single.php
singular.php
Static Page
Page Template
Custom Template
$custom.php
Default Template
page-$slug.php
page-$id.php
page.php
Site Front Page
front-page.php
Page Shown On Front
Posts Shown On Front
home.php
Blog Posts Index page
Error 404 Page
404.php
Search Result Page
search.php
For oEmbeds:
embed-{post-type}-{post_format}.php
embed-{post-type}.php
embed.php
wp-includes/theme-compat/embed.php
Primary Template
Secondary Template
Variable Template
Page Type

Creating and Managing Custom Post Types and Taxonomies

Post Types

WordPress comes with built-in post types like **Posts and Pages**. Developers can create custom post types to handle unique content that doesn't fit within the default structures.

Taxonomies

Taxonomies are a way to **group and categorize** content in WordPress. The default taxonomies are Categories and Tags. Developers can create custom taxonomies to further organize content types.

Creating Custom Post Types

The **register_post_type()** function is used to create custom post types, allowing developers to specify various arguments to customize the behavior and features of the post type.

Creating Custom Taxonomies

The **register_taxonomy()** function is used to create custom taxonomies, enabling developers to associate taxonomies with specific custom post types.

```php
<?php // Register Events CPT
Tabnine | Edit | Test | Fix | Explain | Document | Ask | 0 references
function create_events_cpt(): void {
    $args = array(
        'labels' => array(
            'name' => 'Events',
            'singular_name' => 'Event',
            'add_new_item' => 'Add New Event',
            'edit_item' => 'Edit Event',
            'all_items' => 'All Events'
        ),
        'public' => true,
        'has_archive' => true,
        'rewrite' => array('slug' => 'event'),
        'supports' => array('title', 'editor', 'thumbnail', 'ex
    );
    register_post_type('event', $args);
}
add_action('init', 'create_events_cpt');

// Register Event Categories & Tags
Tabnine | Edit | Test | Fix | Explain | Document | Ask | 0 references
function create_event_taxonomies(): void {
    register_taxonomy('event_category', 'event', array(
        'label' => 'Event Categories',
        'rewrite' => array('slug' => 'event-category'),
        'hierarchical' => true,
    ));
    register_taxonomy('event_tag', 'event', array(
        'label' => 'Event Tags',
        'rewrite' => array('slug' => 'event-tag'),
        'hierarchical' => false,
    ));
}
add_action('init', 'create_event_taxonomies');
```

WordPress Hooks, Filters, and Actions

Hooks: Customization Points

Actions: Executing Custom Code

Filters: Modifying Data

add_action() and add_filter(): Extending Core Behavior

```
0 references
function add_custom_message(): void {
    echo '<p>Hello,Team!</p>';
}
add_action('the_content', 'add_custom_message');

0 references
function modify_post_title($title): string {
    return $title . ' - Special Event';
}
add_filter('the_title', 'modify_post_title');
```

What is a Plugin?

Extends Functionality

Plugins add powerful new features and capabilities to the host application, significantly expanding its functionality and versatility. By integrating seamlessly, plugins enable developers to enhance the application's core features, customize it to specific needs, and provide users with a richer, more tailored experience.

Modular Design

Plugins are designed as modular components that can be **easily integrated and removed without affecting the core application**.

Customization

Plugins allow users to customize the application to their unique needs and preferences, tailoring the experience to fit specific requirements. They expand the core functionality, transforming the application into a more personalized and versatile tool.

Ecosystem

Many applications have a thriving plugin ecosystem, where developers create and share a wide variety of plugins to extend the functionality of the application.

In summary, plugins are valuable software components that enhance the **functionality** and flexibility of applications, allowing users to personalize and extend the capabilities of the host software to meet their specific needs.

Common WordPress Plugins

Jetpack

Jetpack is a powerful plugin that offers a wide range of features, from **security and performance** enhancements to social sharing and analytics.

WooCommerce

WooCommerce is a popular e-commerce plugin that allows you to easily set up an online store and manage your products, orders, and payments.

Yoast SEO

Yoast SEO is a comprehensive SEO plugin that helps you optimize your content for search engines, including features for meta tags, sitemaps, and more.

Contact Form 7

Contact Form 7 is a simple and versatile form builder plugin that allows you to create and customize contact forms for your website.

Akismet

Akismet is an anti-spam plugin that helps protect your website from spam comments and other unwanted content.

Working with Advanced Custom Fields (ACF)

Advanced Custom Fields (ACF) is a powerful WordPress plugin that allows developers to **create custom fields** and field groups for their WordPress websites.

ACF provides a user-friendly interface for **adding, organizing, and managing** these custom fields, making it easier to extend the functionality of a WordPress site beyond the default content types.

Optimizing WordPress for Performance

Compress and optimize all images on your WordPress site to reduce file size and improve loading times.

Configure your WordPress website to leverage browser caching, allowing visitors' browsers to **store static resources** and reduce repeat requests.

Minify your site's CSS, JavaScript, and HTML files to reduce file size and improve page load speeds.

Implement a CDN to serve static assets, such as images, CSS, and JavaScript, from a network of servers closer to your visitors, reducing latency.

Regularly optimize your WordPress database by removing unused data, such as **post revisions**, **spam comments**, and transient options.

Select a WordPress hosting provider that offers fast, reliable, and optimized hosting to ensure your website loads quickly.

Optimizing Images

Smush

A popular image optimization plugin that compresses and resizes images without sacrificing quality. It also allows for lazy loading of images.

Imagify

Another effective image optimization plugin that compresses images and converts them to WebP format for better performance. It also offers features like bulk optimization and automatic resizing.

Leveraging advanced image optimization tools like Smush and ShortPixel can significantly enhance your website's performance by reducing image file sizes without compromising quality.

Performance Optimization

Browser Caching

Stores static files (like images, CSS, and JavaScript) in a user's browser so they don't need to be reloaded on subsequent visits. Implemented using .htaccess or a caching plugin.

Page Caching

Saves the rendered HTML of pages and posts, reducing the load on the server and speeding up delivery to users. Implemented using plugins like W3 Total Cache or WP Super Cache.

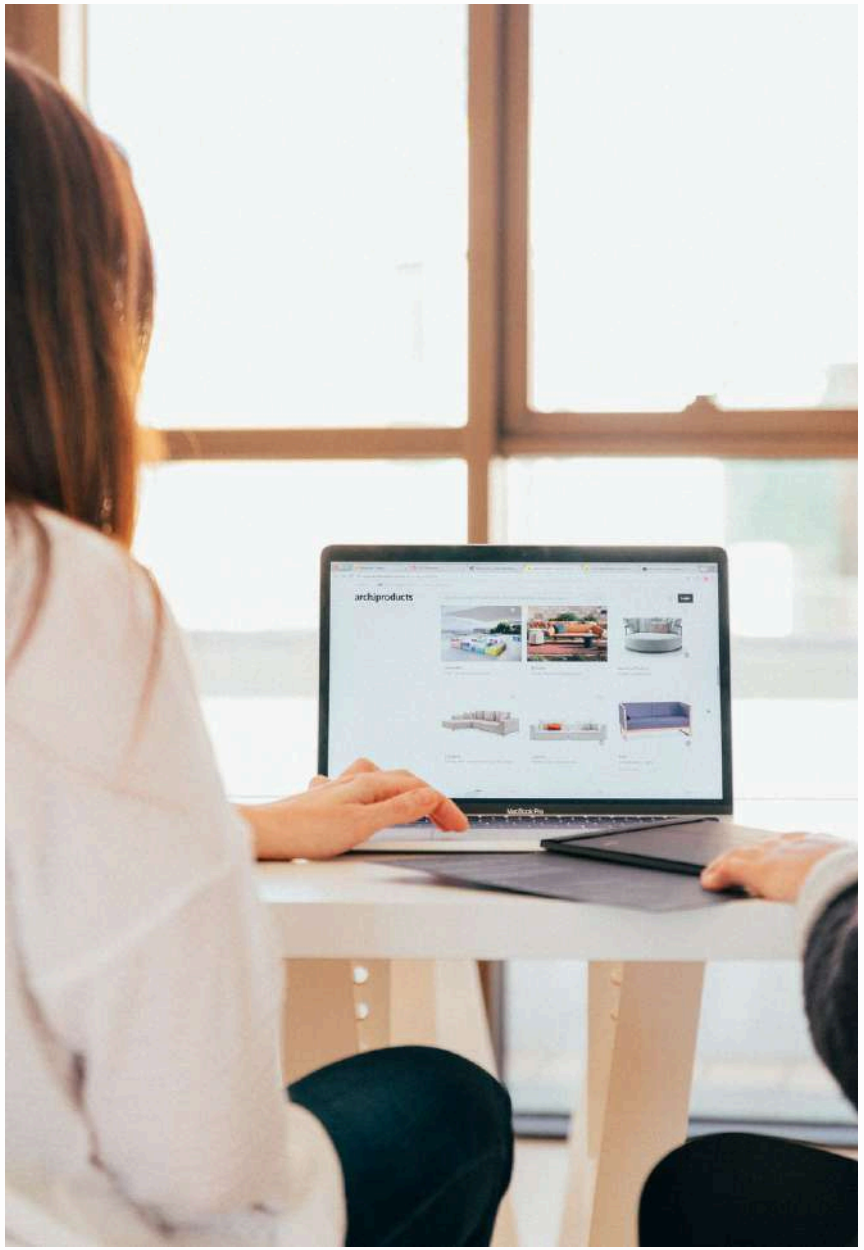
Object Caching

Stores database query results **in memory** to speed up repeated queries, useful for dynamic sites. Implemented using solutions like Redis or Memcached, with plugins like Redis Object Cache or W3 Total Cache.

Leveraging these caching layers can significantly improve the performance and responsiveness of your WordPress site.

Reducing HTTP Requests and Using a CDN

Minimizing the number of HTTP requests and optimizing asset delivery can significantly enhance website performance. This slide covers strategies for reducing HTTP requests and leveraging a Content Delivery Network (CDN) to improve overall page load times.



WordPress Security Best Practices

- ## Keep WordPress Updated
  Regularly update your WordPress core, themes, and plugins to the latest version to ensure you have the latest security patches and bug fixes.

- ## Use Strong Passwords
  Create strong, unique passwords for your WordPress admin account and any other user accounts to prevent brute-force attacks.

- ## Enable Two-Factor Authentication
  Require two-factor authentication for all user accounts to add an extra layer of security and prevent unauthorized access.

- ## Limit Login Attempts
  Implement a limit on the number of login attempts allowed to prevent brute-force attacks and protect your website from unauthorized access.

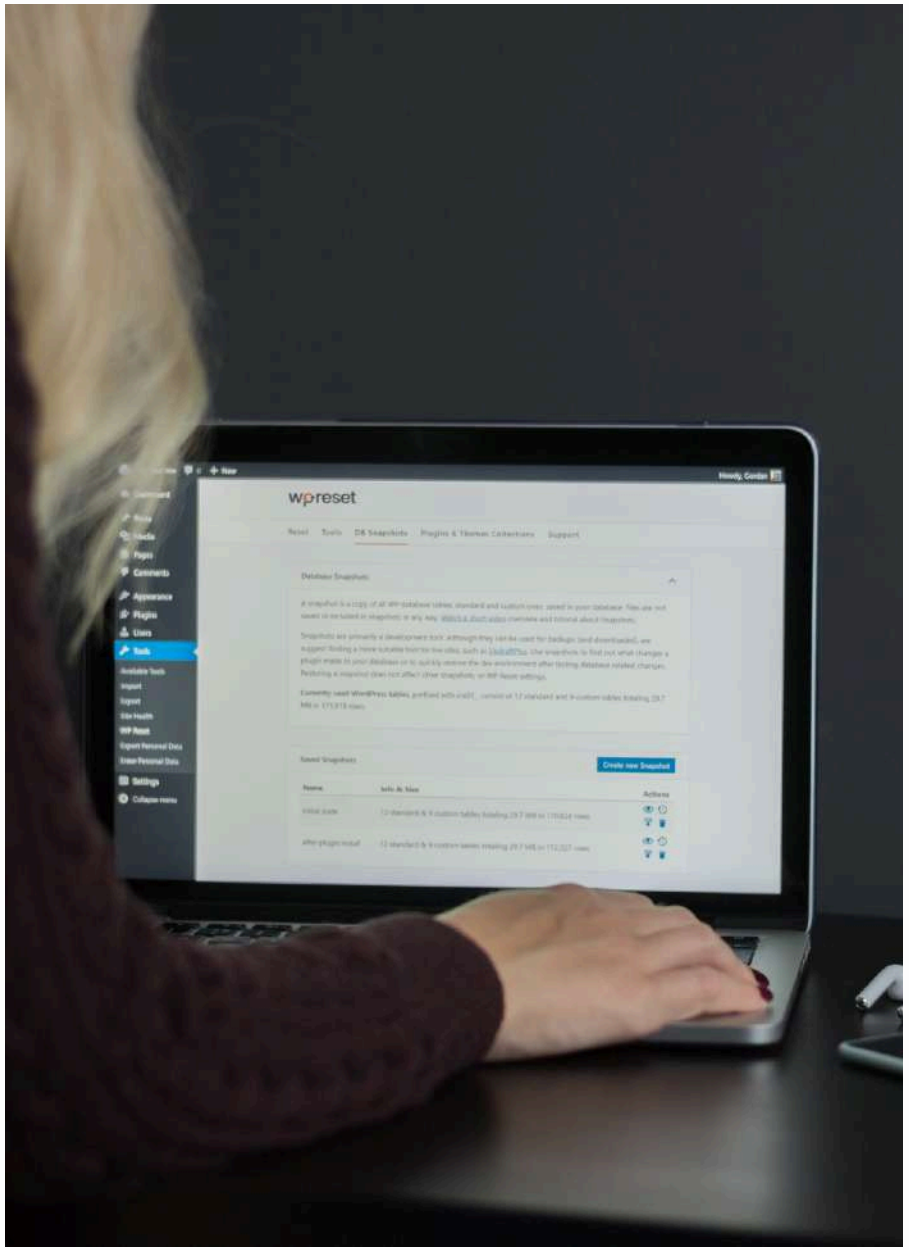
- ## Disable File Editing
  Disable the ability to edit files directly within the WordPress admin panel to prevent potential security vulnerabilities.

- ## Restrict Access to the WordPress Admin Area
  Limit access to the WordPress admin area by restricting IP addresses or requiring a specific URL path.

- ## Regularly Back Up Your Website
  Implement a reliable backup solution to ensure you can restore your website in the event of a security breach or other issues.

WordPress as a Headless CMS

WordPress has traditionally been known for its full-stack approach, where the front end and back end are tightly coupled. However, in recent years, WordPress has evolved to offer a headless CMS approach, allowing developers to decouple the frontend and backend, and leverage its powerful REST API to build custom, dynamic web applications.



Introduction to the WordPress REST API

Overview of the REST API

The WordPress REST API provides a standardized, programmatic interface for interacting with WordPress sites, allowing developers to build decoupled applications that consume WordPress content and functionality.

Available Core Endpoints

The REST API exposes a wide range of core endpoints, such as
/wp-json/wp/v2/posts,
/wp-json/wp/v2/pages,
/wp-json/wp/v2/users,

Benefits of Decoupled Architecture

Using the WordPress REST API enables a decoupled architecture, where the frontend and backend are separated, allowing for more **flexibility, scalability**, and the ability to **build custom user experiences** across various platforms and devices.

Common Issues

White Screen of Death

Can occur due to PHP errors or memory limit exhaustion. Solution: Enable WP_DEBUG to see errors, increase the PHP memory limit by adding define('WP_MEMORY_LIMIT', '256M') to wp-config.php.

Internal Server Error (500)

Indicates a server misconfiguration or a corrupted .htaccess file. Solution: Rename the .htaccess file to .htaccess_old and refresh the site. If it resolves the issue, regenerate the .htaccess file by going to Settings > Permalinks and saving changes.

Broken Links or 404 Errors

Can occur when permalinks are not set correctly or a post/page has been moved. Solution: Update permalinks in Settings > Permalinks to refresh the rewrite rules.

Plugin/Theme Conflicts

Issues can arise after installing new plugins or themes. Solution: Deactivate all plugins and switch to a default theme (like Twenty Twenty-Three) to see if the issue persists. Reactivate them one by one to identify the culprit.

Database Connection Errors

This error typically means the database credentials in wp-config.php are incorrect. Solution: Verify database settings (DB_NAME, DB_USER, DB_PASSWORD, DB_HOST) in wp-config.php.

Troubleshooting and Debugging

- ## Enabling WP_DEBUG

  Understand what WP_DEBUG is and how to enable it in the wp-config.php file to display and log errors.

- ## Reading Error Logs

  Learn where to find the debug.log file and how to interpret common error messages, such as fatal errors and warnings.

- ## Using Query Monitor

  Explore the powerful Query Monitor plugin to analyze database queries, PHP errors, HTTP API calls, and hook/action execution.

- ## Quick Troubleshooting Tips

  Apply quick fixes like **clearing cache**, **checking server logs**, **restoring backups**, and **updating software**.

THE END.