

Unit -1

OOP (Object Oriented Programming)

- **Concept of OOP**

Object oriented programming is nothing but a technique to design your application. Application could be any type like it could be web based application, windows based application. OOP is a design concept. In object oriented programming, everything will be around the objects and class. By using OOP you can create modular web application. By using OOP we can perform any activity in the object model structure. There are many benefit of using oop over the parallel or procedural programming.

- **Object**

Any thing in the world is an object. Look around and you can find lots of object. Your laptop, pc, car every thing is an object. In this world every object has two thing properties and behaviors. Your car has property (color, brand name) and behavior(it can go forward and backward). If you are able to find properties and behaviors of real object. Then it will be very easy for you to work with Object Oriented Programming.

The object in programming is similar to real word object. Every programming object has some properties and behaviors. For example, if you have an object for interest calculator then it has property interest rate and capital and behavior simple interest calculation and compound interest calculation. Interest calculator has some property and method from calculator object like addition, multiplication etc.

- **Class**

The class is something which defines your object. For example, your class is Car. And your Honda car is object of car class. Like object explanation, here we will take an example of the real word and then we will move further in programming definition.

Blueprint of the object is class. The class represents all properties and behaviors of the object. For example, your car class will define that car should have color, the number of doors and your car which is an object will have color green and 2 doors. Your car is an object of a class car. Or in terms of programming, we can say your car object is an instance of the car class. So structural representation (blueprint) of your object is class.

Now let us take an example of the programming. Your interest calculator object is an instance of class interest calculator. Interest calculator class defines properties like capital, rate, and behavior like simple interest calculation and compound interest calculation.

Now in your object when interest calculation behavior will be applied it will take your rate of interest and capital and provide you the result.

- **Constructor**

When you create a new object, it is useful to initialize its properties e.g., for the bank account object you can set its initial balance to a specific amount. PHP provides you with a special method to help initialize object's properties called constructor.

To add a constructor to a class, you simply add a special method with the name `__construct()`. Whenever you create a new object, PHP searches for this method and calls it automatically.

The following example adds a constructor to the BankAccount class that initializes account number and an initial amount of money:

Now you can create a new bank account object with an account number and initial amount as follows:

- **Destructor**

PHP destructor allows you to clean up resources before PHP releases the object from the memory. For example, you may create a file handle in the constructor and you close it in the destructor.

To add a destructor to a class, you just simply add a special method called `__destruct()` as follows:

There are some important notes regarding the destructor:

- Unlike a constructor, a destructor cannot accept any argument.
- Object's destructor is called before the object is deleted. It happens when there is no reference to the object or when the execution of the script is stopped by the `exit()` function.

The following simple FileUtil class demonstrates how to use the destructor to close a file handle.

In this tutorial, we have introduced PHP constructor and destructor to help you initialize object's properties and clean up resources before the object is deleted.

- **Inheritance**

Inheritance is specific to object-oriented programming, where a new class is created from an existing class. Inheritance (often referred to as subclasses) comes from the fact that the subclass (the newly created class) contains the attributes and methods of the parent class.

The main advantage of inheritance is the ability to define new attributes and new methods for the subclass which are then applied to the inherited attributes and methods.

This can be used to create a highly specialized hierarchical class structure. The biggest advantage is that there is no need to start from scratch when wanting to specialize an existing class.

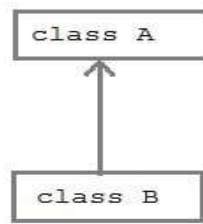
As a result, class libraries can be purchased, providing a base that can then be specialized at will (the company selling these classes tends to protect member data using encapsulation).

inheritance has three types, single, multiple and multi-level inheritance. But, PHP supports single inheritance and multi-level inheritance.

In PHP, inheritance can be done by using `extends` keyword, meaning that, we are extending the derived class with some additional properties and methods of its parent class.

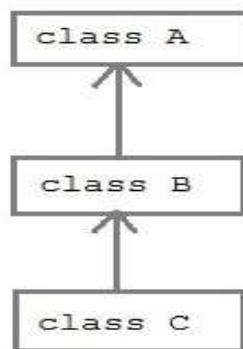
Single Inheritance:

For implementing single inheritance concept in PHP, we require two classes one as a parent and the other as a child.



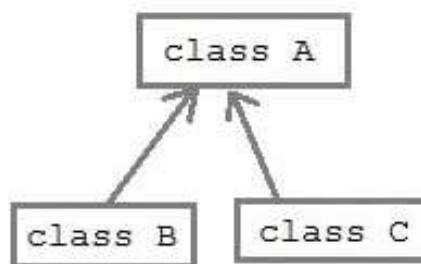
Multilevel Inheritance:

In multilevel, one-to-one ladder increases. Multiple classes are involved in inheritance, but one class extends only one. The lowermost subclass can make use of all its super classes' members. Multilevel inheritance is an indirect way of implementing multiple inheritance.



Hierarchical Inheritance:

In hierarchical type of inheritance, one class is extended by many subclasses. It is one-to-many relationship.



- **Scope Resolution Operator (::)**

The Scope Resolution Operator in simpler terms, the double colon, is a token that allows access to static, constant, and overridden members or methods of a class.

When referencing these items from outside the class definition, use the name of the class.

As of PHP 5.3.0, it's possible to reference the class using a variable. The variable's value cannot be a keyword (e.g. self, parent and static).

When an extending class overrides the parents definition of a method, PHP will not call the parent's method. It's up to the extended class on whether or not the parent's method is called. This also applies to Constructors and Destructors, Overloading, and Magic method definitions.

- **Autoloading Class Files**

how to organize your class files and load them automatically using PHP `__autoload()` function.

It is good practice to keep each PHP class in a separated file and all class files in a folder named classes. In addition, the name of the class should be used to name the file for example if you have a Person class, the file name should be person.php.

Before using a class you need to load it in your script file using PHP include file function such as `require_once()` as follows:

When the number of classes grows, it is quite tedious to call `require_once()` function everywhere you use the classes. Fortunately, PHP provides a mechanism that allows you to load class files automatically whenever you try to create an object from a non-existent class in the context of the current script file.

PHP will call `__autoload()` function automatically whenever you create an object from a non-existent class. To load class files automatically, you need to implement the `__autoload()` function somewhere in your web application as follows:

Whenever you create an object from a non-existent class, PHP looks into the `./classes` folder and load the corresponding class file using the `require_once()` function.

From now on, you don't have to call `require_once()` function every time you use the class, which is very useful.

Notice that if PHP cannot find the `__autoload()` function or if the `__autoload()` function failed to load the class files, you will get an error message.

- **Class constants**

A constant is, just like the name implies, a variable that can never be changed. When you declare a constant, you assign a value to it, and after that, the value will never change. Normally, simple variables are just easier to use, but in certain cases constants are preferable, for instance to signal to other programmers (or your self, in case you forget) that this specific value should not be changed during runtime.

Class constants are just like regular constants, except for the fact that they are declared on a class and therefore also accessed through this specific class. Just like with static members, you use the double-colon operator to access a class constant. Here is a basic example:

As you can see, it's much like declaring variables, except there is no access modifier - a constant is always publically available. As required, we immediately assign a value to the constants, which will then stay the

same all through execution of the script. To use the constant, we write the name of the class, followed by the double-colon operator and then the name of the constant. That's really all there is to it.

GEETANJALI

UNIT – 2

Introduction Installation & Configuration

- **Content Management System (CMS)**

CMS stands for Content Management System.

The definition of a CMS is an application (more likely web-based), that provides capabilities for multiple users with different permission levels to manage (all or a section of) content, data or information of a website project, or internet / intranet application.

Managing content refers to creating, editing, archiving, publishing, collaborating on, reporting, distributing website content, data and information.

An example of a CMS application is a Web Application that provides the following administration, control panel or website management functionalities:

Administration or Control Panel:

- Create, Edit, Publish, Archive web pages
- Create, Edit, Publish, Archive articles
- Create, Edit, Publish, Archive press releases

Introduction of Wordpress

WordPress is web publishing software you can use to create your own website or blog. Since it was released in 2003, WordPress has become one of the most popular web publishing platforms. Today it powers more than 70 million websites!

WordPress is open source software you can use to create a beautiful website or blog. It just may be the easiest and most flexible blogging and website content management system (CMS)

WordPress enables website owners to update page content and operate a blog page through a friendly interface.

But many people don't realize that WordPress is much more just a blogging tool. It's also a highly flexible content management system (CMS) that enables you to build and manage your own full-featured website using just your web browser. Best of all, it's completely FREE.

That's because WordPress is an 'Open Source' project. That means that hundreds of volunteers from all around the world are constantly creating and improving the code for the WordPress software. And, there are thousands of plugins, widgets, and themes that enable you to build just about any type of website you can imagine.

There are several reasons WordPress is a great choice for building your blog or business website.

1. Open Source

First of all, WordPress is Open Source. That means that there are hundreds of people all around the world working on improving WordPress. And because WordPress is an open source project it's also completely FREE.

2. User-Friendly

Second, it's user-friendly. Rather than having to hire a web designer or contact a "webmaster" every time you want to make a small change to your website, you can easily manage and update your own content — without having to learn HTML. In fact, if you know how to use the basic formatting tools in a program like Microsoft Word, you can edit your site.

3. Flexible and Extensible

Third, it's flexible and extensible. There are thousands of plugins and themes that enable you to easily change the entire look of your website, or even add new features like polls or contact forms, with just a few clicks.

4. Easy to find support

Next, if you run into problems, or you want to add custom features, it's easy to find support or hire someone to help you. In addition to the WordPress tutorials on this site, there are also thousands of WordPress developers and designers who can help you. The official WordPress Forum is a great place to get answers to your questions.

5. SEO-friendly

WordPress is SEO-friendly. Right out of the box, WordPress includes everything you need to ensure that your content is optimized for search engines. This is critical to your site's visibility and online success. Matt Cutts of Google says, "WordPress is made to do SEO well."

6. Control of your own content

Last, you're in control of your own content. Other publishing platforms limit what you can and can't do on your own website. And, you're locked in to that service. If it should ever shut down, your content could simply disappear. With WordPress, you can import your data from other systems like Blogger or Tumblr. And you can also easily export your data to move away from WordPress, should you choose. You're in control of your site... and your content.

- **WordPress Advantages and Key Features**

WordPress is the most desired website development platform today and its popularity can indeed be evaluated with the fact that it powers over 23 percent of the websites. Well, the figure isn't constant but increases every single day.

Although WordPress has considerably high competition in the website development market, this powerful CMS (i.e. Content Management System) combines a group of features, viz. Simplicity and User-friendliness.

Its under-the-hood complexity for developers cannot be overlooked though. However, the following list of standard features and advantages, powers WordPress stand out in the competition:

- **Quick Installation and Upgrade**

WordPress has simple and quick processes of installation and upgrade. Simply create your web pages online and upload the database. In case of using the FTP program, simply create a database and upload WordPress, and subsequently install it to continue.

- **User Management**

A website has several different roles associated with it. For instance, the administrator to manage the web pages, writers and editors to manage content, users or subscribers to create and manage their profiles. WordPress makes the management quick and easy.

- **Simplicity of Operations**

The simplicity and ease of operations make the process efficient and the productivity level enhances as a result. With WordPress, you can anytime create a new web page and publish content with quickness, thereby following the standard Internet culture.

- **Inbuilt Themes**

WordPress has multiple inbuilt themes and allows you create more as per your personal or business requirements. WordPress API powers you create themes that can be very simple as well as complex in design.

- **Inbuilt Comments**

You write and publish a blog of your WordPress website. Your friends and followers can put their comments, as WordPress' built-in comments feature provides them with a space or forum for discussion. You can manage or moderate those comments.

- **Extensions & Plug-ins**

WordPress is a feature-rich website development platform that meets your different needs. A plug-in directory full of plug-ins extend its features. Use these plug-ins enhance the features and functionality of your website.

- **Flexibility**

WordPress is an Open-source platform for website development and you can proceed with any type of theme, i.e. a personal blog to a full-fledged professional website. Choose any of the existing designs or create a new one.

- **Media Management**

You can add images, videos, and other media items to your WordPress website and enhance the richness of content. Simply drag the media content from your computer and drop it into the uploader to get it

uploaded. Use the image editing tools if required.

Not only is WordPress available free of cost, but also the open source to be used in any manner.

Advantages & Disadvantages of Wordpress

- **Advantages:**

1. **Easy to Use**

While most products advertise themselves as being “easy to use” WordPress doesn’t advertise this, this is the reputation it has earned. Unlike Drupal and Joomla, WordPress doesn’t require knowledge of PHP or HTML coding. WordPress comes with rich preinstalled features that are literally point and click, installing plugins is simple, and their templates make designing easy.

2. **Menus**

WP 3.0 has some great menu options, making it prettier and simpler than Joomla due to its custom menus that can be rearranged to include categories, pages, etc.

3. **Community**

Having a large community to draw on for support and tricks is important to many people and WP has one of the largest online communities around.

4. **Themes**

There are thousands of free themes available online to use to help you choose a design for your site. There are also premium themes available for purchase for those who want an added touch of professionalism to their site.

5. **Plugins**

There are almost 13,000 plugins available for the WP platform. It is through these plugins that WP gained its CMS title as plugins opened up WP to a world of possibilities. Most of these plugins are free which is cool.

6. **Custom Fields**

Through the use of custom fields you can turn WP into a CMS by going beyond the typical blogging activities like posting, categorizing and tagging. They have made this process easy for beginners as well with the addition of custom field plugins like ‘Custom Fields Template,’ ‘Flutter’, and ‘More Fields’.

7. **Pods CMS**

This plugin allows you to create and manage different content types without using custom fields.

- **Disadvantages:**

1. **WP Scripting**

For advanced users who are familiar with more advanced techniques like scripting, WP uses its own script. And knowledge of the WP script is necessary for things like adding or removing post dates.

2. **Design Knowledge Required**

While WP does have plenty of nice-looking templates many people want to design a unique website as WP themes often look too similar to one another. To make unique designs in WP you need knowledge of the CSS style sheet language, and odds are that if you have this sort of designing knowledge you will want a more powerful toolset like those available in Drupal-like CMSs.

3. **Too Many Plugins**

WP needs a lot of plugins to be able to do the things that Joomla and Drupal can do out of the box. While the plugins are available they can slow your page down if you install too many of them. The double-edged sword on WP plugins is that by the time your page has been slowed because of the

plugins you have already put a significant amount of man-hours into the project and would be reluctant to migrate to another CMS. Plan for this ahead of time by looking at how many plugins you would need to create the functionality you want.

4. PHP Security

Unfortunately the PHP security for WP has a shoddy track record and security isn't as strong as some of the other platforms. This will only be a factor depending on how sensitive the content of your site is.

5. Documentation

While there is a large online community of users to help and support you there is little to no offline documentation. WP has been criticized for not creating a MS SDK document, especially now that it calls itself a CMS.

6. Tables and Graphics

Modifying and formatting tables and graphic images is more difficult in WP than in its competitors.

7. Needs Regular Updates

WP is constantly changing and growing and it needs regular updates. This is not a big deal unless you are looking for a set-it-and-forget-it solution, in which case this might not be the solution for you as it needs periodic updates.

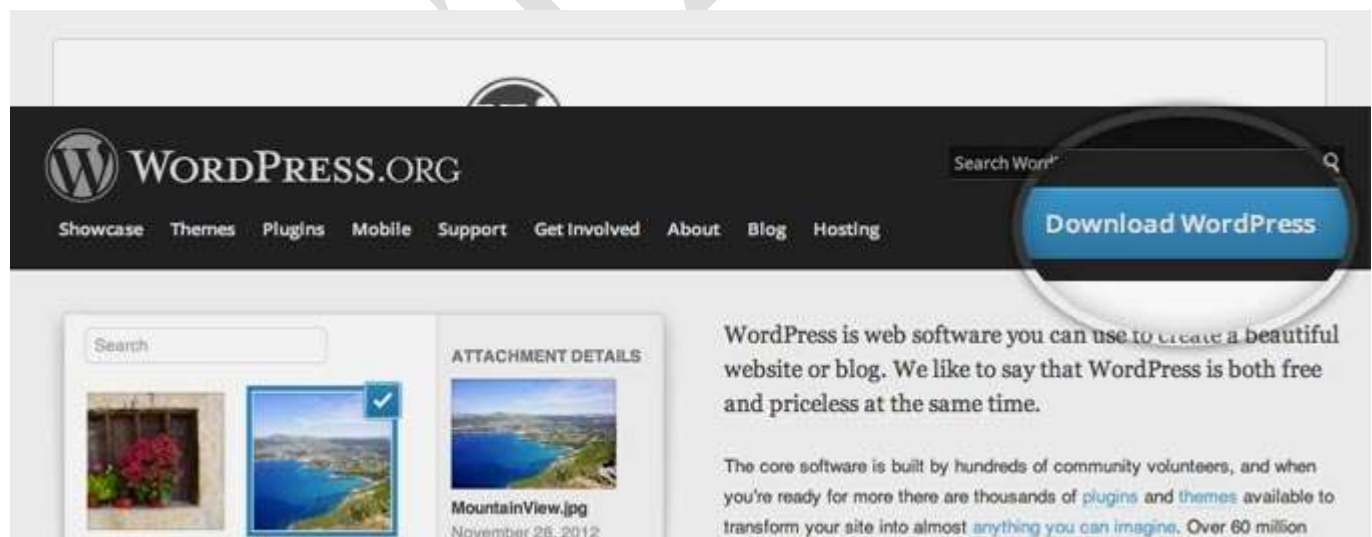
8. Queries

Queries can be a difficult and convoluted process because of all the custom fields you need to use to go take WP beyond typical blogging actions. Often you need a Custom Select Query, which can be a royal pain in the butt.

- **Installation of wordpress**

Step -1 : Download the WordPress installation package

To start the installation process, first you need to download WordPress from its [official download page](#). We recommend that you always download and install the latest stable version of WordPress.



Once you click on the Download button for the latest WordPress version, the installation package will be saved to your hard disk. Locate the installation package that you've just downloaded and extract it to a new folder.

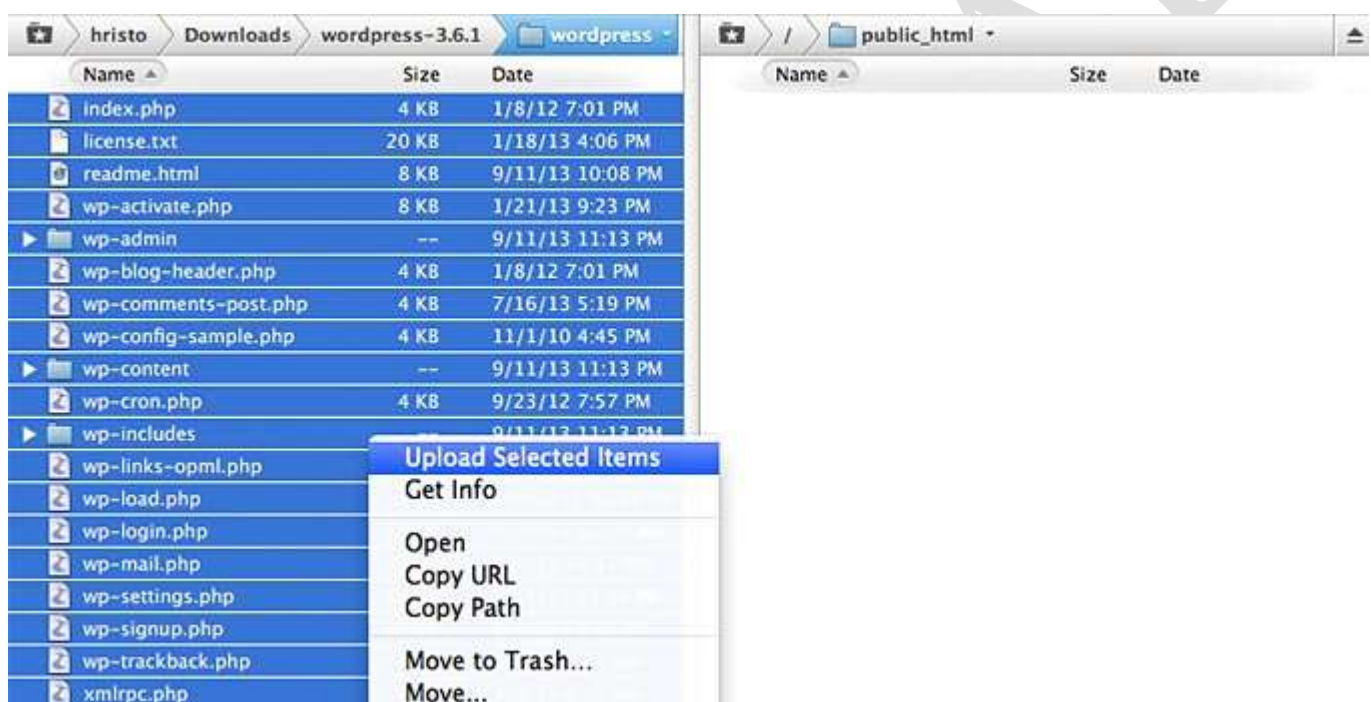
Step 2 : Upload the WordPress Files to Your Server

Now, you need to upload the extracted files and folders to your web server. The easiest way to upload the installation files is via FTP.

If you want your WordPress to be the main installation on your account and to be accessible through your main domain (i.e. www.mydomain.com), you need to upload the extracted files to your `public_html` folder.

Once the download is complete, extract the archive and upload it to your web hosting account. You can do that via FTP using a client application like Filezilla or via cPanel -> File Manager -> Upload file(s). If you want this WordPress installation to be main for your website, the files should reside in the `public_html` folder of your account.

However, you can always make a subfolder (i.e. `public_html/blog`) if you want to run only part of your website on WordPress.



Step 3 : Create a MySQL Database for WordPress to use

Now, you need to create a MySQL database and assign a user to it with full permissions. Once you create your MySQL Database and User, make sure you write down the database name, database username and password you've just created. You will need those for the installation process.

MySQL Users

Add New User

[↑ Jump to MySQL Databases](#)

Username: user_wp

Password:

Password (Again):

Strength (why?): Very Strong (96/100) [Password Generator](#)

Add User To Database


User:

Database:

Step 4: Go through the installation process

Now it's time to navigate to your website to start with the installation process. If you have uploaded WordPress in your public_html directory you'll need to go to <http://yourdomain.com> in your preferred browser. The first thing you will notice is a message, telling you that you don't have a wp-config.php file and you should create one. Just click on the Create a Configuration File button to proceed.

On this page you will see a message, asking you to prepare the necessary information for the installation. Since we already have this information, simply press the Go! button.



The image shows the WordPress installation screen. At the top is the WordPress logo. Below it, a message says: "Welcome to WordPress. Before getting started, we need some information on the database. You will need to know the following items before proceeding." This is followed by a numbered list of five items: 1. Database name, 2. Database username, 3. Database password, 4. Database host, and 5. Table prefix (if you want to run more than one WordPress in a single database). Below the list, a paragraph states: "If for any reason this automatic file creation doesn't work, don't worry. All this does is fill in the database information to a configuration file. You may also simply open wp-config-sample.php in a text editor, fill in your information, and save it as wp-config.php." Another paragraph follows: "In all likelihood, these items were supplied to you by your Web Host. If you do not have this information, then you will need to contact them before you can continue. If you're all ready..." At the bottom left, there is a circular button with the text "Let's go!" inside it.

Enter the details for your newly created MySQL database and press the Submit button



Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	<input type="text" value="user_dbname"/>	The name of the database you want to run WP in.
User Name	<input type="text" value="user_username"/>	Your MySQL username
Password	<input type="text" value="yourpassword"/>	...and your MySQL password.
Database Host	<input type="text" value="localhost"/>	You should be able to get this info from your web host, if localhost does not work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

Submit

WordPress will now check if your settings are correct. If you have entered all the necessary information, you will see a confirmation screen. Press the Run the Install button to proceed.

On the next screen you will have to enter the information about your administrative username and the title of your new site. In addition, you can specify whether you'd want search engines to index your site or not. Once you fill in that information, press the Install WordPress button. Bear in mind, however, that you should specify a real email address. It can be later used in case you forget your password.

Your new WordPress application is installed. You can use the Login In button to access your administrative backend and start posting in your new site.

Wordpress Directory & file structure

The directory structure is the organization of files into a hierarchy of folders. It describes how files are arranged for an application. A hierarchy is similar to a tree structure.

The WordPress Directory Structure

The core WordPress files and directories are listed below.

wp-admin

wp-content

wp-includes

index.php

license.txt

readme.html

wp-activate.php

wp-blog-header.php

wp-comments-post.php

wp-config-sample.php

wp-cron.php

wp-links-opml.php

wp-load.php

wp-login.php

wp-mail.php

wp-settings.php

wp-signup.php

wp-trackback.php

xmlrpc.php

.htaccess

wp-config.php

These are the core WordPress directories and files. Now, let's see some of the important files and folders in detail. Keep in mind that the first three are folders and rest are files.

- **wp-admin**

The admin tools are powered by this folder. As its name indicates, this deals with the administrator. The main file inside this directory is the admin.php. It enables the connection to the database, displays the WordPress dashboard, and performs any other number of key functions, such as checking if any given user is in fact the admin in question.

- **wp-content**

The next folder we are going to see is the wp-content. The Themes and Plugins are familiar to every WordPress user. These are stored inside this directory.

- **Plugin**

The plugins are used to add more functionality to the WordPress site. Plugins can offer custom setup to the WordPress installation while the default WordPress installation is designed to be light weight.

- **Themes**

The WordPress themes provide the graphical interface to the website. There are many files that work together to achieve this.

The themes and plugins are the major parts in the wp-content directory.

- **wp-includes**

The wp-includes is the final top-level folder and is large in size. As we have seen earlier, the wp-admin includes all the files necessary to power said admin functions, wp-content stores all your themes and plugins, and wp-includes is what enables the site to run.

This folder is where most of the WordPress core files are stored. A fresh WordPress install will include over 140 different files in the main directory, and fourteen different folders including certificates, fonts, js, theme-compact, and widgets.

These subfolders aren't important as the files included in the main folder, such as functions.php. This file is part of WordPress' core, and it comes with a lot of the functions that enable the WordPress installation to work. As an example, some lines of code will be seen when you open the file on a text editor, and they're just a regular function meant to transform dates into different formats.

- **index.php**

The index file loads and initializes all your WordPress files when a page is requested by a user.

- **license.txt**

This is WordPress license file. The WordPress is a free software and is licensed under the GNU General Public License as published by the Free Software Foundation.

- **readme.html**

This core file contains the instructions to the user as its name indicates.

- **wp-activate.php**

This contains the following:

- **do_activate_header()**

Function: Adds an action hook specific to this page that fires on wp_head.

- **activate_wp_head**

Fires before the Site Activation page is loaded, but on the wp_head action.

- **wpmu_activate_stylesheet()**

Function: Loads styles specific to this page.

- **activate_header**

Action Hook: Fires before the Site Activation page is loaded.

- **wp-blog-header.php**

This folder contains the http headers.

- **xmlrpc.php**

WordPress uses an XML-RPC interface. WordPress has its own implementation for WordPress-specific functionality in an API called the WordPress API. This should be used when possible, and your client should use the API variants beginning with the wp prefix.

XML-RPC functionality is turned on by default since WordPress 3.5.

In previous versions of WordPress, XML-RPC was user enabled. To enable, go to Settings > Writing > Remote Publishing and check the box.

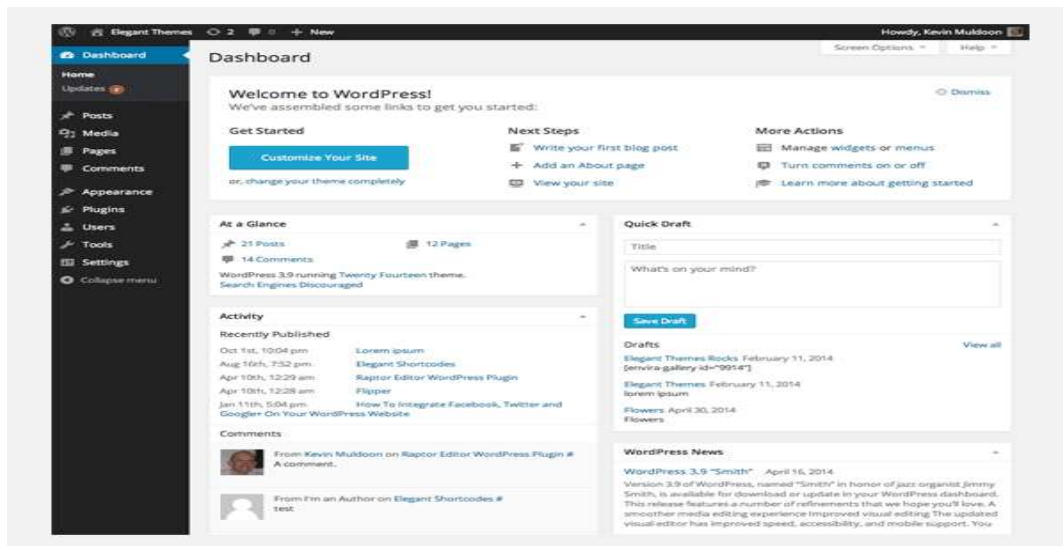
- **wp-config.php**

It is one of the core WordPress files which contains information about the database, including the name, host (typically localhost), username, and password.

There are many other folders and files, but these are the most important folders and files in the WordPress directory structure.

- **Dashboard overview**

The WordPress dashboard area gives you a general overview of your website. It also displays many useful quick links for performing common tasks such as writing a quick draft or replying to the latest comment.

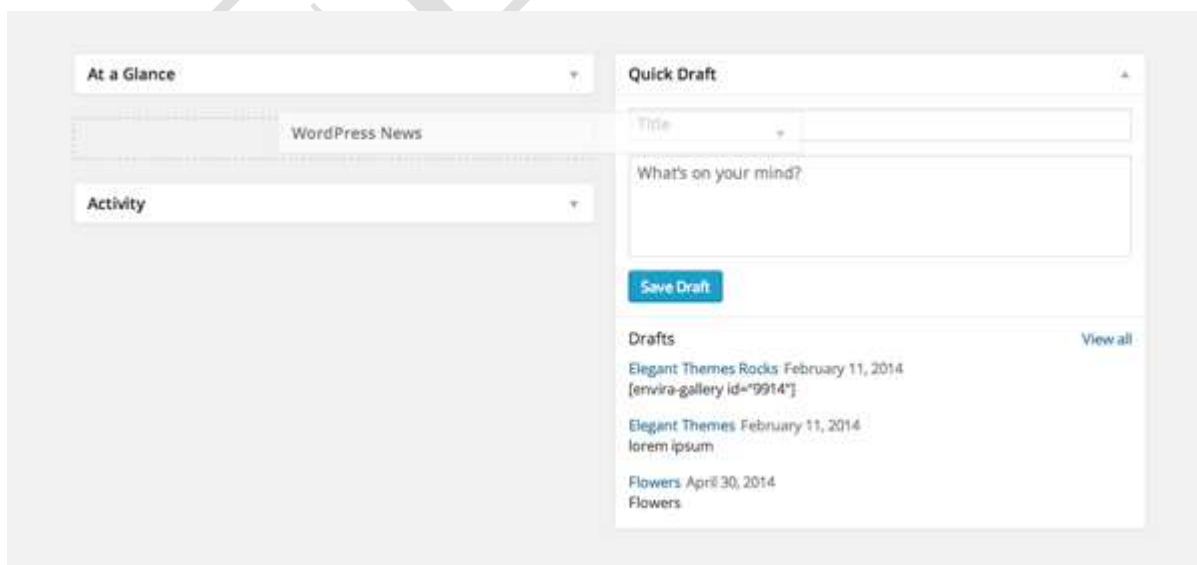


The dashboard area is built up of many different widgets. Each widget can be enabled or disabled.

To do this, click on the “Screen Options” drop down menu at the top of the page. This will show you a list of default widgets and widgets that have been added by plugins. Simply uncheck the widgets that you want to remove and they will automatically be hidden.



Many beginners do not alter the default dashboard that WordPress initially offers. However, WordPress allows you to minimize widgets. It also allows you to drag widgets to a new location. This lets you place the widgets you use more frequently in a more prominent position.

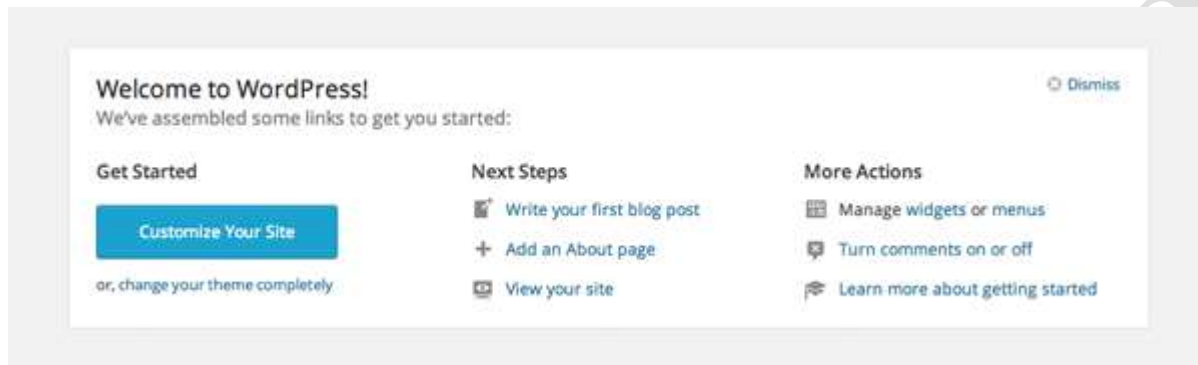


Let us take a closer look at the default widgets that populate the WordPress dashboard.

Welcome to WordPress!

The welcome widget is displayed at the top of the dashboard. On the left hand side is a large “Customize” button that takes you to the WordPress theme customizer and a link underneath to the main themes area.

The center column contains useful links to create a blog post, create a page, and view the front end of your website.



The last column contains links to the widgets page, menu page and comment settings. A link to the “First Steps With WordPress” page in the WordPress codex is also displayed.

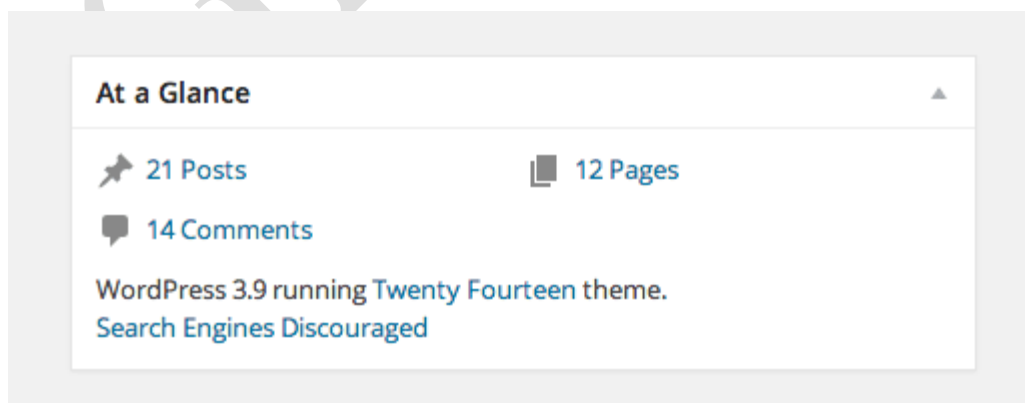
Unlike other widgets, there is no option to minimize the “Welcome to WordPress” box. Instead, there is an option to remove it. I assume that this is because the widget is aimed at beginners.

You may have dismissed the widget already and removed it from your dashboard, however if you find its quick links useful, you can re-enable the “Welcome to WordPress” box via the screen options drop down menu.

At a Glance

The “At a Glance” widget gives you a general overview of your website including the number of published posts and published pages. The figure for the total number of comments include spam comments.

The theme you are currently using is also noted and a reminder is printed if you have blocked search engines from indexing your website.

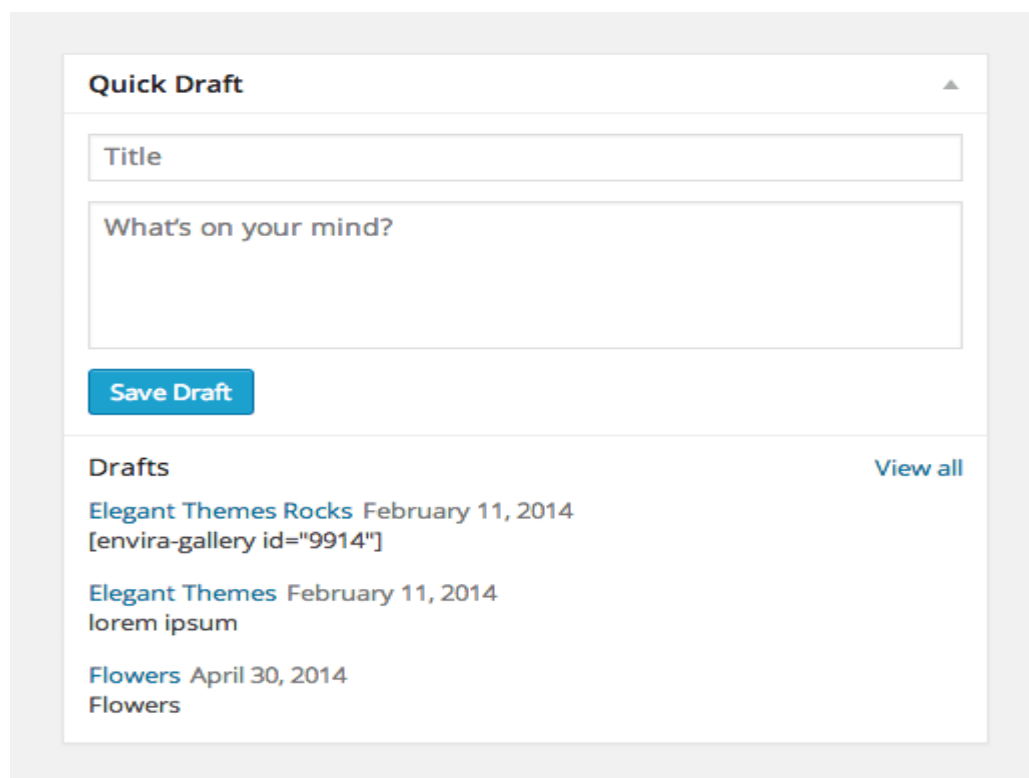


Some plugins add additional information to this widget. For example, if Akismet is activated, it displays how many spam comments it has protected you from and how many spam comments are queued.

- **Quick Draft**

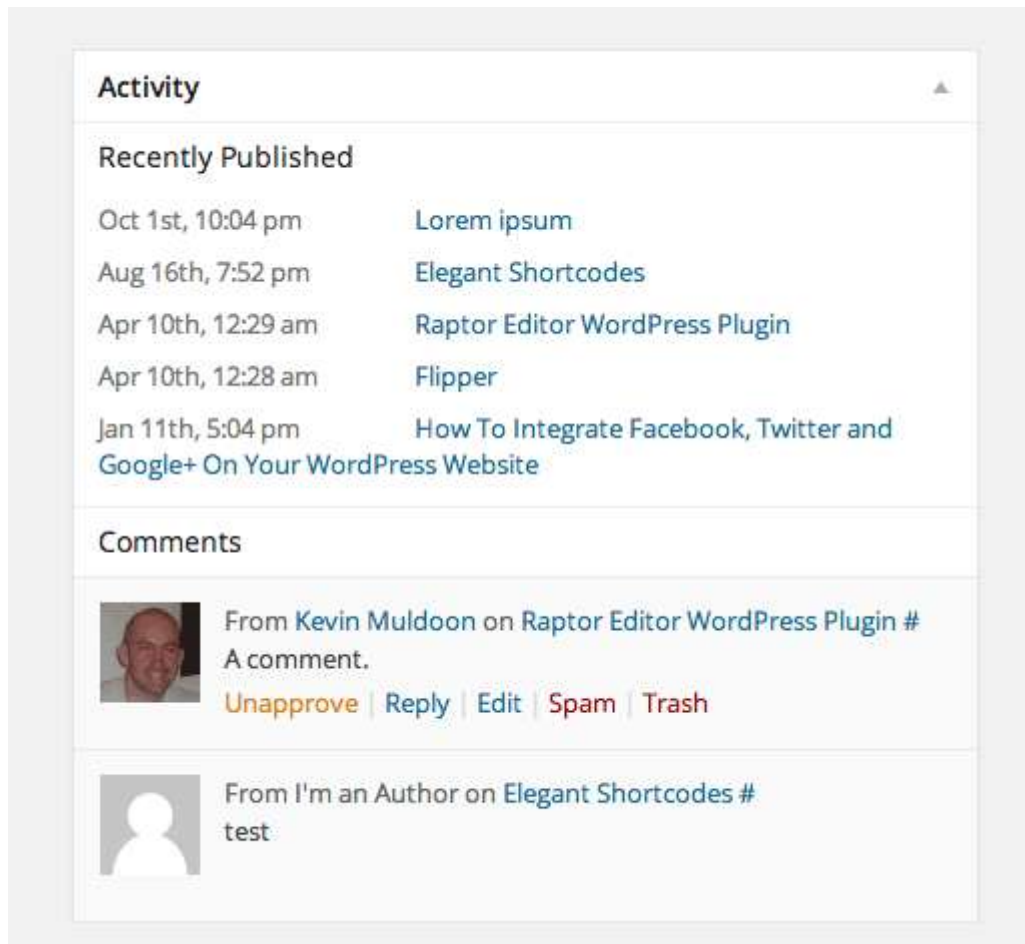
The “Quick Draft” widget is handy for jotting down an idea for a future blog post. There is no visual editor available, therefore you cannot make text bold or upload an image.

The concept of quick draft is simple. If you have an idea for a post, write down a title and some notes about your idea and then save it as a draft. You can then complete the post at a later date.

A screenshot of the 'Quick Draft' widget. It features a title input field, a larger text area with the placeholder 'What's on your mind?', and a blue 'Save Draft' button. Below the input fields is a 'Drafts' section with a 'View all' link. The drafts list includes: 'Elegant Themes Rocks February 11, 2014 [envira-gallery id="9914"]', 'Elegant Themes February 11, 2014 lorem ipsum', and 'Flowers April 30, 2014 Flowers'.

- **Activity**

The “Activity” widget displays your last few published posts. It also displays the latest comments that have been submitted. You can respond to the comment directly through the widget. You can also trash the comment or move it to spam.



- **WordPress News**

The “WordPress News” widget displays the latest blog posts from official WordPress blogs such as WP Tavern and the WordPress.org blog.

In the past, I would check this widget to see if there was any WordPress update. Since WordPress now advises you whenever a new update is live, I tend to hide this widget.

WordPress News

WordPress 3.9 "Smith" April 16, 2014

Version 3.9 of WordPress, named "Smith" in honor of jazz organist Jimmy Smith, is available for download or update in your WordPress dashboard. This release features a number of refinements that we hope you'll love. A smoother media editing experience Improved visual editing The updated visual editor has improved speed, accessibility, and mobile support. You

WPTavern: Would Anyone Be Interested in a WordCamp Badges Plugin?

WPTavern: How To Archive A Site You Don't Have Access To

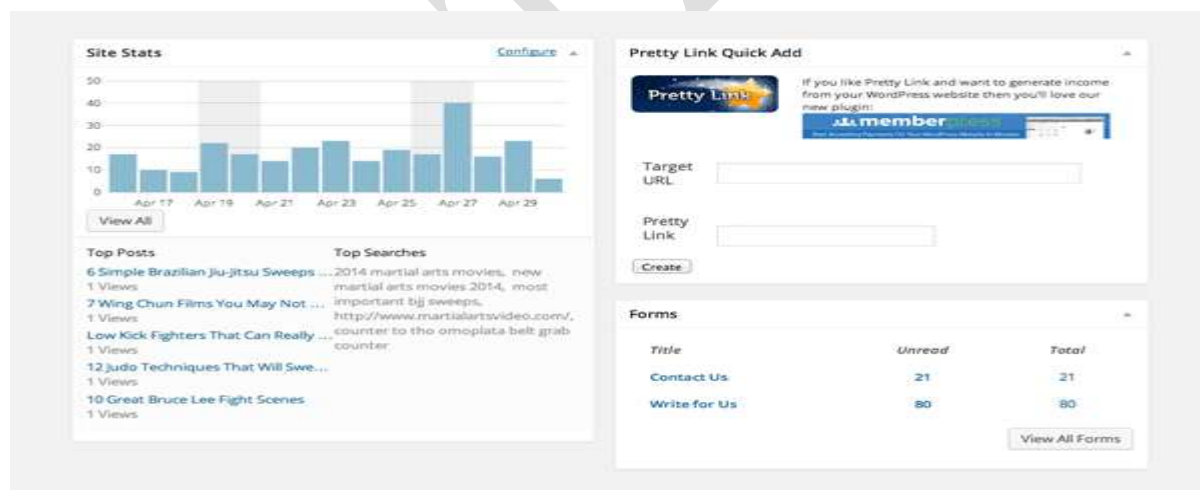
WPTavern: How to Share Beer Recipes in WordPress

Popular Plugin: WooCommerce - excelling eCommerce (Install)

- **Custom Dashboard Widgets**

The WordPress dashboard is not restricted to the five widgets that come packaged with WordPress. Many plugins add a widget to your dashboard after activating it.

For example, if you have the WordPress.com Stats module from [Jetpack](#) activated, you can see a summary of your traffic stats directly from your dashboard.



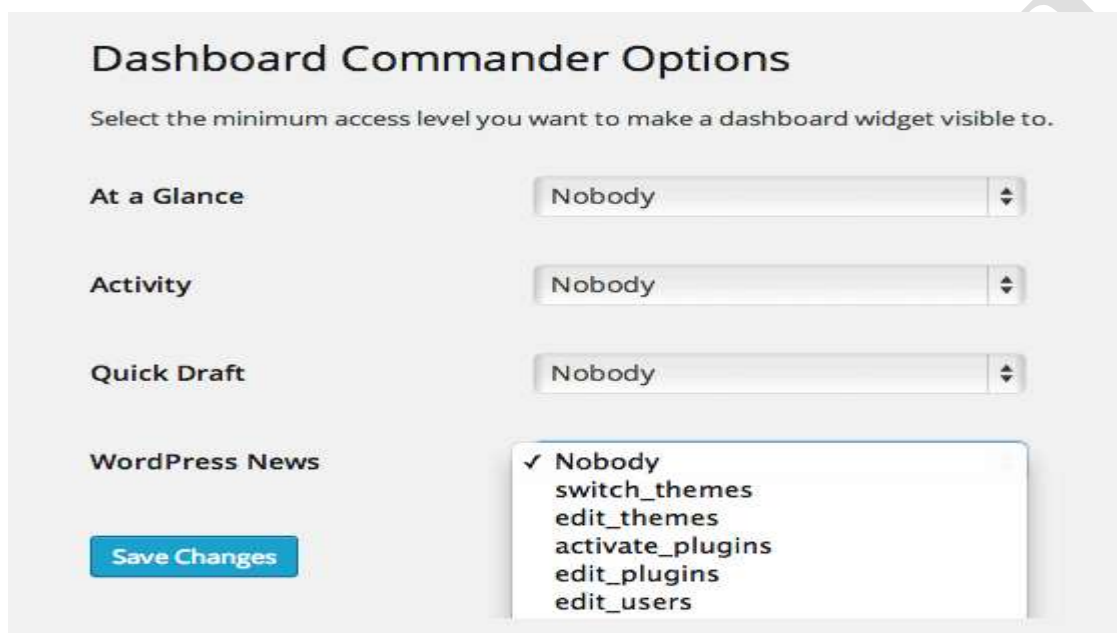
Other plugins, such as Dashboard Notes or Advanced Drafts & Reviews Dashboard Widget, are specifically designed to enhance your WordPress dashboard. This allows you to truly customize your dashboard with the information you need. For example, on my personal blog I have installed Google Analytics Dashboard for WP so that I can have a quick glance at my recent traffic stats every time I log in.

You can also add your own custom widgets to your dashboard by adding functions to a plugin or to your theme's functions.php template. This does, however, require a little technical knowledge to execute.

If you have a lot of plugins installed on your website, your dashboard may start getting a little overcrowded. Do not be alarmed if this occurs as you can disable any unwanted widgets through the screen options menu. Alternatively, you could use a plugin such as Nova Dashboard Cleanup to remove unwanted widgets.

The default widgets, such as activity and quick draft, are automatically displayed on the dashboard of other users such as contributors and authors. Plugins that add widgets to your dashboard may also show them to unauthorized users. This can be a big problem as users may see sensitive information that they were not supposed to (e.g. traffic, important notes etc).

To control what widgets other user groups can see, I recommend using a plugin such as Dashboard Commander. It allows you to restrict access of widgets to users who have a certain permission level.



For many people, the WordPress dashboard is simply the page they see before they click on the admin menu and go to a specific page. For others, the dashboard has become a nuisance as it has been filled with useless widgets from plugin developers.

Yet, if you take control of your dashboard, it can be a fantastic tool. It can be a place to save ideas, reply to comments, and view your current traffic statistics. WordPress gives you the tools to control your dashboard; however it is your job to use them.

- **Adding a New Web Page**

To add a new web page:

1. Click on the Pages menu in the sidebar and select "Add New."
2. Enter your page title in the top box
3. Add the page content into the WordPress Editor.
4. In the right sidebar, you will see a box labelled "Page Attributes." Here you can specify a parent page and a template.

- **Parent Page:** If your new page belongs on a sub-menu beneath another page, you should specify that page as it's Parent.
- **Template:** You may also choose a template if your theme has multiple templates; however, most pages will use the default template.



5. Make sure you click the Publish button to save your changes and make them live. If you are not ready to go live, you can also choose to save your page as a draft.

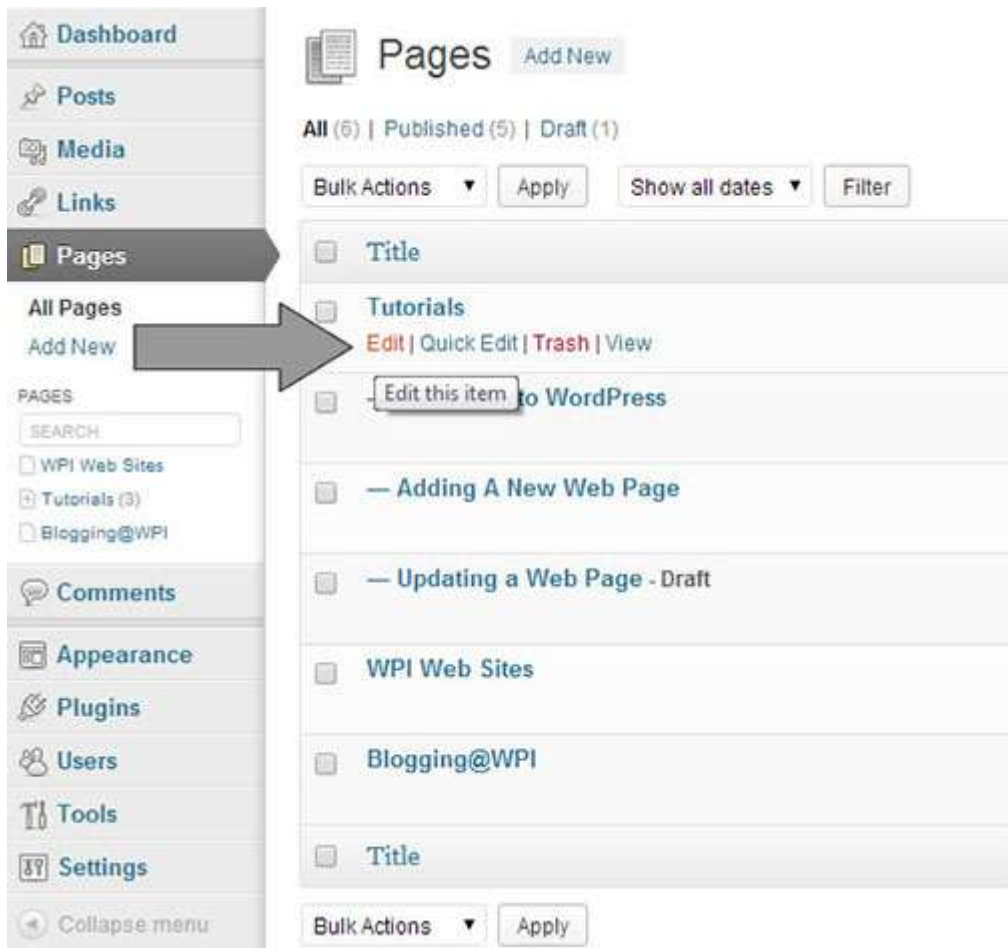
Important Note: Some themes are set up to automatically add your page to the menu and others require that an Administrative user manually add the page to the menu.

• **Updating a Web Page**

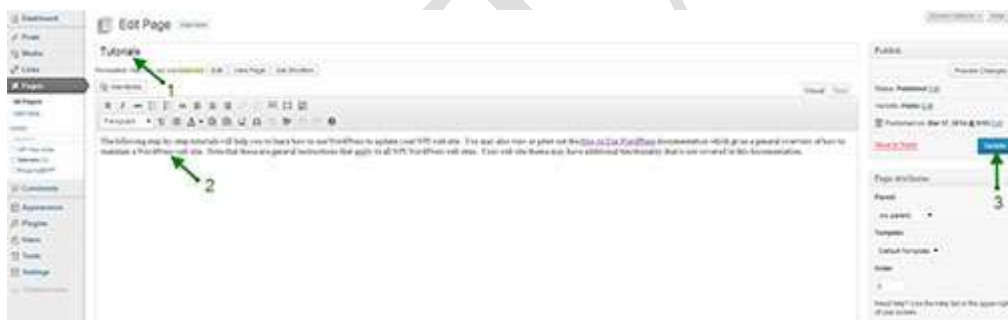
The process for updating an existing web page is basically the same as adding a new page.

Click on the Pages menu to bring up a list of your pages.

When you hover over the page that you want to edit, a menu will appear below that page as shown in the screen-shot below. Click on the Edit link to edit your web page.



When the web page opens, you can edit your title(1) and your page content(2) as desired. Make sure you click the Update button (3) to save your changes and make them live.



• Adding a Post

One of the great things about WordPress is that its interface is very consistent. So if you already know how to add/edit web pages, then this tutorial will seem like a review. There are however a few “extra” things that you need to do when adding a blog post.

To add a new blog article, click on the Posts menu in the sidebar and select “Add New.”

1. Enter your post title in the top box
2. Add the content into the WordPress Editor
3. In the sidebar, you need to choose a category for the article to appear in.

4. You should also specify tags for your article (Make sure you check out the tutorial on Organizing your Blog with Categories & Tags)
5. If your theme uses Excerpts, you can specify a shortened version of the article in the Excerpts box.
6. Make sure you press the Publish button to save your changes and make them live!



• Updating a Blog Post

The process for updating a blog post is basically the same as adding a new post.

Click on the Post menu to bring up a list of your articles.

When you hover over the article that you want to edit, a menu will appear below that page as shown in the screen-shot below. Click on the Edit link to edit your post.



When the post opens, you can edit your title(1) and your page content(2) as desired.

You can also change the categories and tags in side bar if necessary.

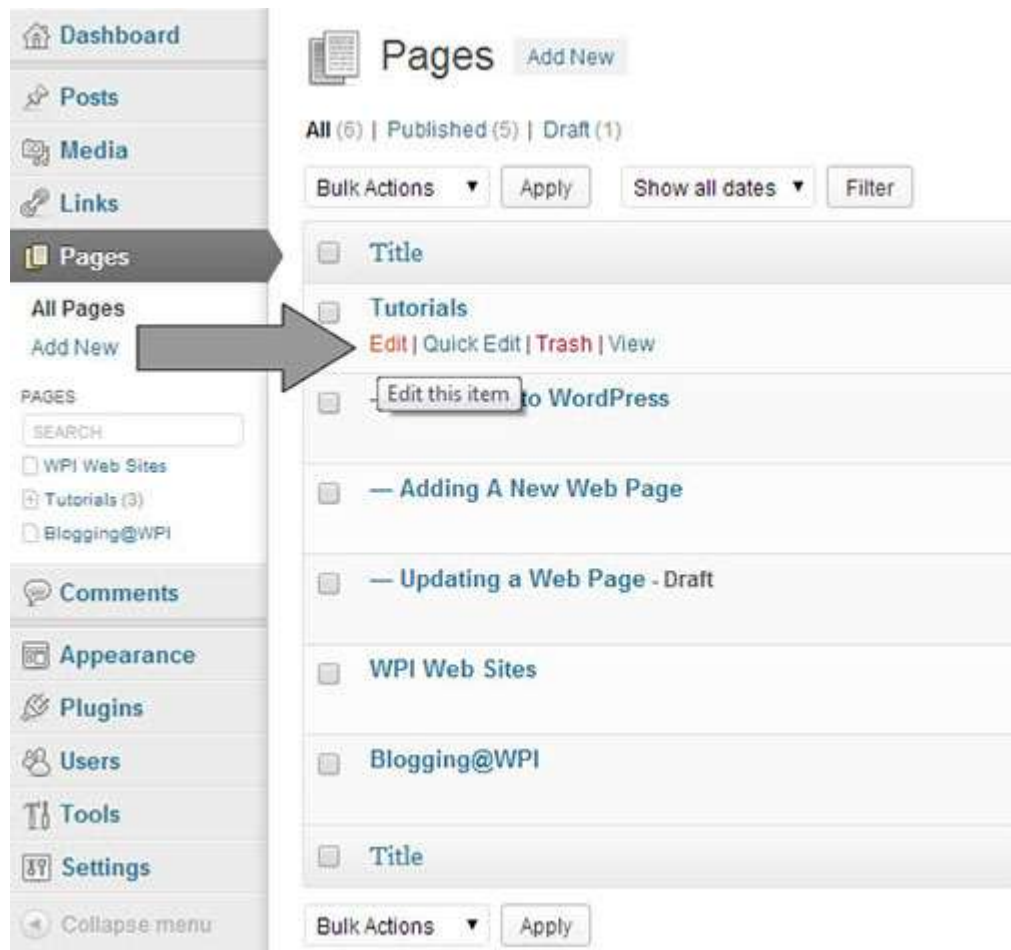
Make sure you click the Update button (3) to save your changes and make them live.



- **Deleting a Page or Post**

To delete a web page or blog post, click on the appropriate menu (Pages or Posts) and hover over the one you wish to delete..

On the hover menu, you will see a Trash option. Click this to move the page to the trash.



Note that the file is just moved to the trash and is not actually deleted. You can go into the trash folder to restore a page if you accidentally delete a page.



To get to the trash folder go to either the Pages section or the Posts, depending upon what you want to delete. At the top of the page, there is a horizontal menu (1) that allows you to filter the page to view: All Pages, Published Pages, Draft Pages or Trash. Click on the Trash link.

To RESTORE(2): Hover over the page you want to restore and click on the Restore link. This will restore the page to its former state – either Published or Draft.

To PERMANENTLY DELETE(2): Hover over the page you want to delete and click on the Delete Permanently link.

You can also check the box prior to multiple pages and use the “Bulk Action” drop-down box to Restore or Permanently Delete multiple pages at once.

In addition, you can “Empty the Trash(3)” to delete all of the pages in the trash at once.

- **How to Add Media (pdfs, jpgs, etc.) to Posts and Pages**

In order to add or update content and/or media on a WordPress website, you must have Administrator, Editor, Author or Contributor privileges.

- **How to upload .jpg, .gif, .png and .pdf files:**

(It is not recommended to upload and /or link to .doc files)

1. Log into your WordPress website
2. Click on Media (left navigation) > Add New
You will see a ‘Multi-file uploader’ – you can do the same tasks with the ‘Browser uploader’ as well, however these instructions are for using the ‘Multi-file uploader’. File types you can upload: .jpg, .gif, .png and .pdf.
3. Either:
drag and drop the media files you wish to use on your website to the space inside the dotted lines;OR
click on ‘Select Files’, find the file you wish to upload, and click to upload.
Information about the file will then appear.
4. (Optional) You can now edit the image by clicking the ‘Edit Image’ button
5. Fill in the Alternate Text field
This is the text that would appear if a visitor’s images are turned off or if a visitor was using a text or screen reader. It should describe the image in a succinct manner.
6. Take note of the File URL
This is the URL you use to link to the image or file.
7. Click ‘Save all changes’ to keep your file in the system
You will be directed to the Library page that lists all uploaded media for your website.

- **How to delete a media file:**

If you no longer wish to have a media file in your system:

1. Click on Media > Library
2. Hover over the name of the file you wish to delete
Three options will appear: ‘Edit’, ‘Delete Permanently’ and ‘View’.

3. Click on 'Delete Permanently'
A pop-up window will appear.
4. Click on 'Ok'
Your file will now be permanently deleted and no longer available or in an archive.

Keep in mind that when a file is deleted, there is no back-up and it is not retrievable. It is very important to be sure that when a file is deleted, it is done intentionally and with care.

- **How to find a media file's URL:**

1. Click on Media > Library
2. Hover over the name of the file you wish to delete
Three options will appear: 'Edit', 'Delete Permanently' and 'View'.
3. Click on 'Edit'
A page with all associated data will appear.
4. At the bottom of the page is 'File URL' and a URL.
This is your file's URL.
5. Click on 'Update Media' to return to the Library page
Alternatively, you can simply navigate away from this page. If you have made any changes to the information, you will lose those changes if you do not click 'Update Media' to save them.

- **How to link to media:**

If you wish to link to a media file (eg. a PDF or ZIP file) from a page or post:

1. Open up the Page or Post you wish to add the media file to
2. Highlight the text you'd like to use as a hyperlink
3. Click on the chain icon in the area above the text field you're using – it usually appears in the middle of the top row and if you rest your mouse on it, the words 'Insert / Edit Link (Alt + Shift + A)' will appear with a yellow background. A pop-up box will appear
4. Enter the URL of your file (see above 'How to find a media file's URL')
5. Optional: Click 'Open link in a new window/tab'
Only links to PDF files should open up in separate window; images, videos and webpages should not.
6. Click on 'Add Link'
You will be returned to your original Page or Post, and the text you made into a hyperlink will now be underlined and a different colour than your regular text.

Note:

- A hyperlink's text should be descriptive of the link destination. Whenever possible it should be the title of the page or media file being linked to and never read as 'click here'.
 - If you are linking to a PDF, ZIP file, etc., warn your visitors by including the file type in brackets before the end of the hyperlink.
 - **How to insert an image onto your page:**
1. Open up the Page or Post you wish to insert the image file into

2. Click on 'Upload / Insert'

This is found below the title of your page and above the field where you add your content – there are usually icons on the right side. A new window will appear.

3. If your image has not yet been uploaded, stay on the default 'From Computer' tab and proceed as directed in the "How to upload .jpg, .gif, .png and .pdf files" above; If your image is already uploaded to the website, click on the 'Media Library' tab at the top of the window and continue as described below. When the 'Media Library' tab is opened, a list of all available images will be shown.

4. Find the image you wish to use, and click on 'Show' (on the right)

The section will expand to show all details associated with the image.

5. If Alternate Text has not yet been written, fill in that field

6. Click on where on the page the image should be aligned
Right alignment is recommended.

7. Select the appropriate size for your image

If the image was sized before uploading, click on 'Full Size'

8. Click on 'Insert into Post'

The window will close and you will be returned to your page or post.

User Roles and Capabilities

User roles

Roles are essentially user groups, WordPress comes with six out of the box. These go from the most basic of subscriber all the way up to the most powerful one of Super Admin. Let's learn a bit more about them.

- Subscriber: the most basic role a user can have. A subscriber can only read published posts and pages and modify their own profile.
- Contributor: a contributor can create new posts and modify their content, but can't publish them.
- Author: an author can create, modify and publish their own posts.
- Editor: the editor can do anything an author can do and additionally has the capability of modifying, publishing and deleting other users' posts.
- Administrator: along with all the above permissions and administrator can handle administrative tasks, such as installing plugins and themes, modifying site settings and more.
- Super Admin: the final user role can only be found on multisite installations and grants the user the ability to create and delete sites, install and activate themes and plugins network wide and more.

Capabilities

Capabilities are essentially permissions, each one grants the user a certain ability. For example `publish_posts` allows a user to publish a post they have created, this capability is available to users with the role of author and upwards. As you might imagine the more capabilities a user has, the more powerful they become. A subscriber has a single capability, that of read which means that the only thing they can visit in the WordPress dashboard is their own profile page under Users > Your Profile. At the moment of writing there are about 60 default capabilities which can be found here.

The default roles and their assigned capabilities will cover the needs of most websites out there. However in some cases they might not be perfect. For example, while contributors can create posts they can't upload files of any kind, so they can't add featured or inline images. Another thing that may cause trouble to multi author site owners is that the author role allows users to delete posts that have been already

published, which is problematic because an author can create the post, get paid for it and then delete it after it's published.

By default WordPress does not allow administrators to modify capabilities for user groups, making it hard to fine tune permissions to match their needs. This can be amended by writing some code to add or remove permissions from specific groups, or luckily, by using one of the many role manager plugins available in the plugin directory. Some of the most popular are User Role Editor, Members, Advanced Access Manager, WPFront User Role Editor, Capability Manager Enhanced and User Roles & Capabilities. Basic functionality on all plugins is the same, they will all allow you to modify the capabilities of existing user roles, create new user roles only with the capabilities you need and even create and assign new capabilities.

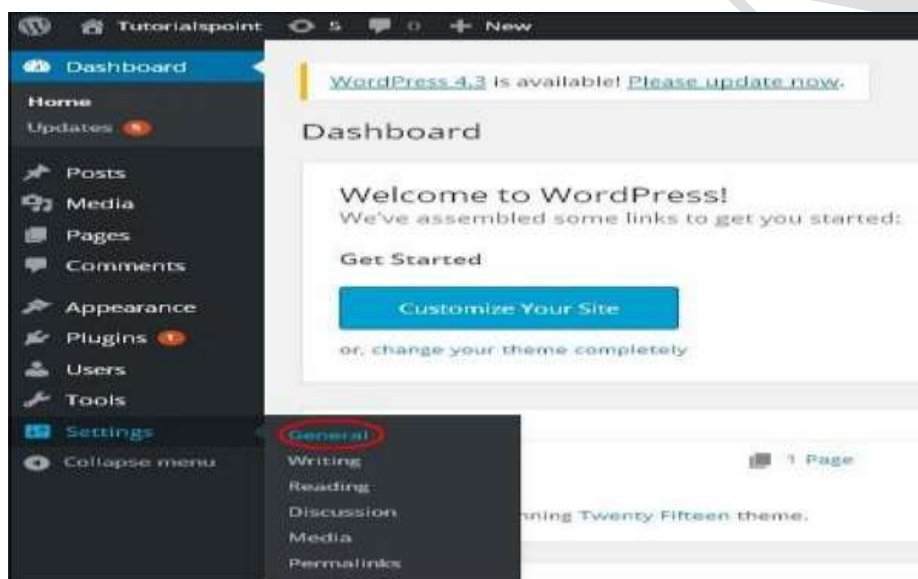
Settings (General, writing, Reading, Discussion, Media, Permalinks)

General Setting

WordPress general setting is used to set the basic configuration settings for your site. In the setting administration screen, it is a default setting screen.

Following are the steps to access the general settings –

Step 1 – Click on Settings → General option in WordPress.



Step 2 – The General Setting page is displayed as shown in the following snapshot.

The screenshot shows the WordPress General Settings page. The left sidebar has a menu with 'Settings' highlighted. The main content area is titled 'General Settings' and includes the following fields and options:

- Site Title:** Tutorialspoint
- Tagline:** Just another WordPress site. Below it, a note says: "In a few words, explain what this site is about."
- WordPress Address (URL):** http://localhost/wordpress
- Site Address (URL):** http://localhost/wordpress. Below it, a note says: "Enter the address here if you want your site home page to be different from your WordPress installation directory."
- E-mail Address:** rushali.bhatkande@tektudis.com. Below it, a note says: "This address is used for admin purposes, like new user notification."
- Membership:** ☐ Anyone can register.
- New User Default Role:** Subscriber
- Timezone:** UTC+0. Below it, a note says: "Choose a city in the same timezone as you." To the right, it says: "UTC time is: 2015-08-25 12:28:48"
- Date Format:** ☒ August 25, 2015. Other options: ☐ 2015-08-25, ☐ 08/25/2015, ☐ 25/08/2015, ☐ Custom: F j, Y August 25, 2015.
- Time Format:** ☒ 12:20 pm. Other options: ☐ 12:20 PM, ☐ 12:20, ☐ Custom: g:i a 12:20 pm. Below it, a link says: "Documentation on date and time formatting."
- Week Starts On:** Monday
- Site Language:** English (United States)

At the bottom, there is a "Save Changes" button and a "Get Version 4.3" link.

Following are the details of the fields on general settings page.

- Site Title – It displays the name of the site in the template header.
- Tagline – Displays a short sentence about your site.
- WordPress Address (URL) – It is the URL of WordPress directory where your all core application files are present.
- Site Address(URL) – Enter the site URL which you want your site to display on the browser.
- E-mail Address – Enter your e-mail address which helps to recover your password or any update.
- Membership – Anyone can register an account on your site after you check this checkbox.
- New User Default Role – The default role is set for the newly registered user or members.
- Timezone – Sets the time zone based on the particular city.
- Date Format – Sets the date format as you need to display on the site.
- Time Format – Sets the time format as you need to display on the site.

- Week Starts On – Select the week day which you prefer to start for WordPress calendar. By default it is set as Monday.
- Site Language – Sets the language for the WordPress dashboard.

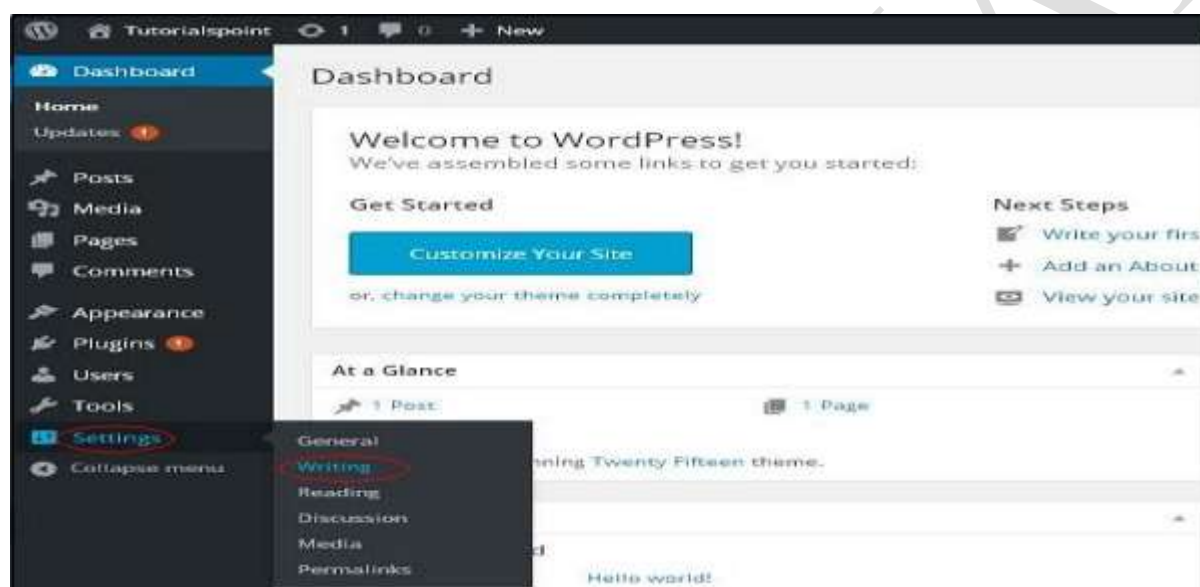
Step3 – After filling all the information about general settings, click on Save Changes button. It saves all your general setting information.

• Writing Setting

The writing settings controls the writing experience and provides options for customizing WordPress site. These settings control the features in the adding and editing posts, Pages, and Post Types, as well as the optional functions like Remote Publishing, Post via e-mail, and Update Services.

Following are the steps to access the writing settings –

Step (1) – To change writing settings, go to Settings → Writing option.



Step (2) – The Writing Setting page is displayed as shown in the following screen.

Writing Settings

Formatting

☒ Convert emoticons like :-) and :-P to graphics on display

☐ WordPress should correct invalidly nested XHTML automatically

Default Post Category

Uncategorized ▼

Default Post Format

Standard ▼

Post via e-mail

To post to WordPress by e-mail you must set up a secret e-mail account with POP3 access. Any mail received at this address will be posted. Random strings you could use: bU6m2bJp, bV2s?1xE, fd&Vj00x.

Mail Server

mail.example.com Port 110

Login Name

login@example.com

Password

password

Default Mail Category

Uncategorized ▼

Update Services

When you publish a new post, WordPress automatically notifies the following site update services. For more about this, see the WordPress Codex.

<http://rpc.pingomatic.com/>

Save Changes

Following are the details of the fields on the page.

- **Formatting** – This field defines two sub options for better user experience.
 - The first option Convert emoticons like :-) and :-P to graphics on display will turn text-based emoticons into graphic-based emoticons.
 - The second option WordPress should correct invalidly nested XHTML automatically corrects the invalid XHTML placed within the posts or pages.
- **Default Post Category** – It is a category to be applied to a post and you can leave it as Uncategorized.
- **Default Post Format** – It is used by themes to select post format to be applied to a post or create different styles for different types of posts.
- **Post via e-mail** – This option uses e-mail address to create posts and publishes posts on your blog through e-mail. To use this, you'll need to set up a secret e-mail account with a POP3 access, and any mail received at this address will be posted.
- **Mail Server** – It allows reading the e-mails that you send to WordPress and stores them for retrieval. For this, you need to have POP3 compatible mail server and it will have URI address such as mail.example.com, which you should enter here.
- **Login Name** – To create posts, WordPress will need its own e-mail account. The Login Name will use this e-mail address and should be kept as a secret as spammers will post links redirecting to their own websites.
- **Password** – Set password for the above e-mail address.
- **Default Mail Category** – It allows selecting custom category for all the posts that are published via Post by e-mail feature.

- Update Services – When you publish a new post, WordPress will automatically notify the site update services in the box. See the Update Services on the codex for the long list of possible services.

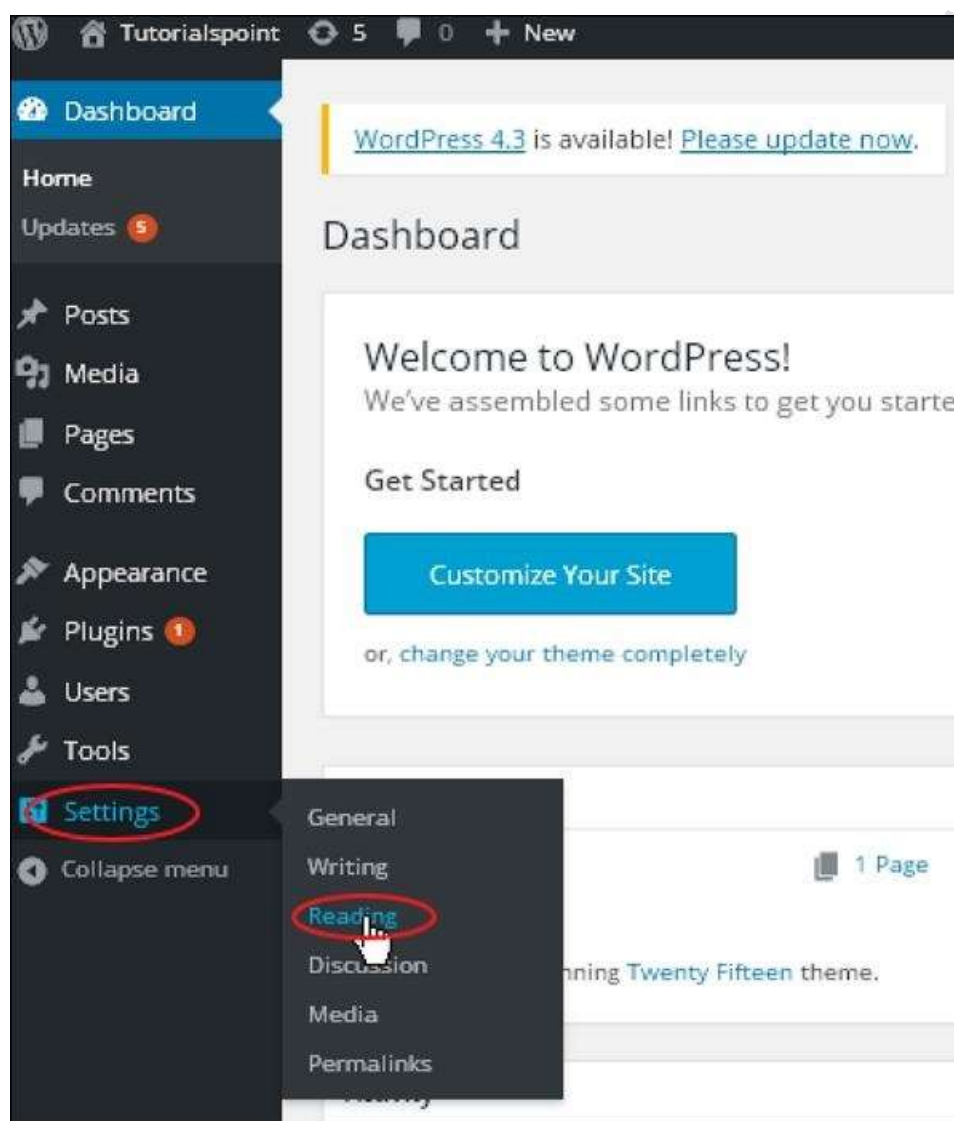
Step (3) – After filling all the above information, click on Save Changes button to save your information.

- **Reading Setting**

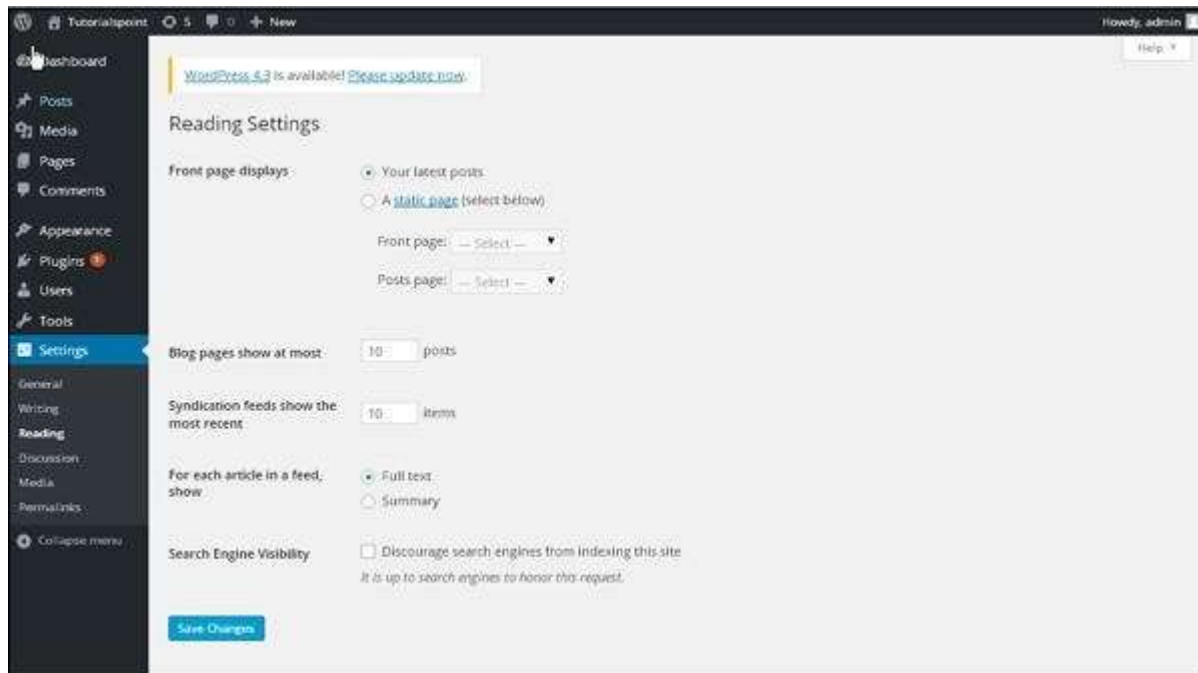
Reading Setting is used to set the content related to the front page. You can set the number of post to be displayed on the main page.

Following are the steps to access the reading settings –

Step (1) – Click on Settings → Reading option in WordPress.



Step(2) – The Reading Settings page is displayed as shown in the following screen.



Following are the details of the fields on reading settings.

- Front page displays – This section is used to display the front page in any of the following format –
 - Your latest posts – It displays latest posts on the front page.
 - A static page – It displays the static pages on the front page.
 - Front Page – You can select the actual page you want to display on front page from the drop down.
 - Posts Page – You can select the page from the drop down which contains posts.
- Blog pages show at most – The number of posts to be displayed per page or site. By default, it is set as 10.
- Syndication feeds show the most recent – The user can view the number of posts when they download one of the site feeds. By default, it is set as 10.
- For each article in a feed, show – This section is used to display the post by selecting any of the following formats –
 - Full Text – It displays the complete post. It is set as default.
 - Summary – It displays the summary of the post.
- Search Engine Visibility – After clicking on the checkbox, Discourage search engines from indexing this site, your site will be ignored by the search engine.

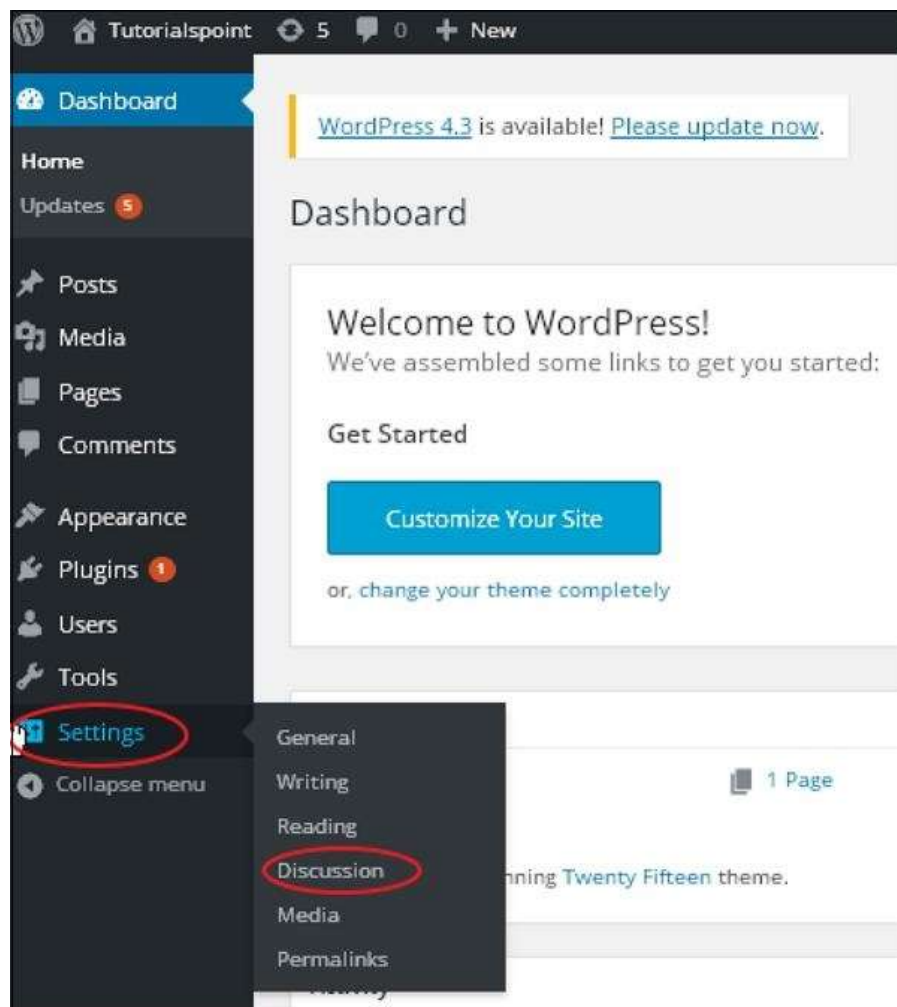
Step(3) – After filling all the information, click on Save Changes button to save your Reading Setting information.

- **Discussion Setting**

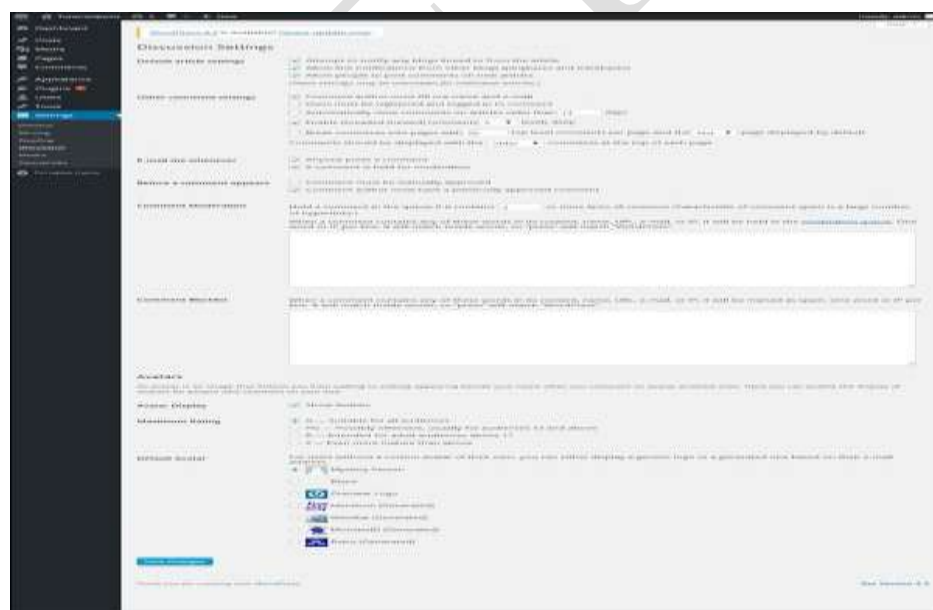
WordPress discussion setting can be defined as the interaction between the blogger and the visitors. These settings are done by the admin to have a control over the posts/pages that come in through users.

Following are the steps to access the Discussion setting –

Step (1) – Click on Settings → Discussion option in WordPress.



Step (2) – The Discussion Settings page is displayed as shown in the following snapshot.



Following fields are seen in Discussion settings.

- Default article settings – These settings are default to the new pages you create or new posts. This contains three more settings. They are –
 - Attempt to notify any blogs linked to from the article – When you publish articles then it sends a notification (sends pings and trackback) to other blogs.
 - Allow link notifications from other blogs (pingbacks and trackbacks) – Accepts pings from other blogs.
 - Allow people to post comments on new articles – You can allow or disallow other people to comment on your article using this setting.

You can change the settings as per your will for individual articles.

- Other Comment Settings – This setting has the following options –
 - Comment author must fill out name and e-mail – When you check this box, it is mandatory for visitors to fill their name and email address.
 - Users must be registered and logged in to comment – If you check this box, only those registered visitors can leave comments, if not checked anyone can leave any number of comments.
 - Automatically close comments on articles older than days – This option allows you to accept comments only for a particular time period as per your wish.
 - Enable threaded (nested) comments – When you check this option, visitors can reply or have a discussion and get responses.
 - Break comments into pages with top level comments per page and the page displayed by default – If your pages are getting a lot of comments then you can split them into different pages by checking this box.
 - Comments should be displayed with the comments at the top of each page – You can arrange the comments in the form of ascending or descending order.
- Email me whenever – This setting contains two options, namely –
 - Anyone posts a comment – When you check into this box, the author gets an e-mail for every single comment that is posted.
 - A comment is held for moderation – This is used in case you do not want your comment to be updated before it's moderated by the admin.
- Before a comment appears – This setting allows how your posts are controlled. There are two more settings as followed –
 - Comment must be manually approved – If you check this box then only the approved comments by the admin can be displayed on the posts or pages.
 - Comment author must have a previously approved comment – This can be checked when you want to approve a comment of an author whose has commented and his e-mail address matches the e-mail address of the previous posted comment. Otherwise the comment is held for moderation.
- Comment Moderation – Contain only a specific number of links that are allowed into a comment.
- Comment Blacklist – You can input your own spam words which you do not want your visitors to enter into the comments, URL, e-mail etc.; later it would filter the comments.
- Avatars – Avatar is a small image that displays at the top-right-hand corner of the dashboard screen beside your name. It is like your profile picture. Here you have a few more options where you can set your avatar for WordPress site.
 - Avatar Display – It displays your avatar besides your name when it is checked.

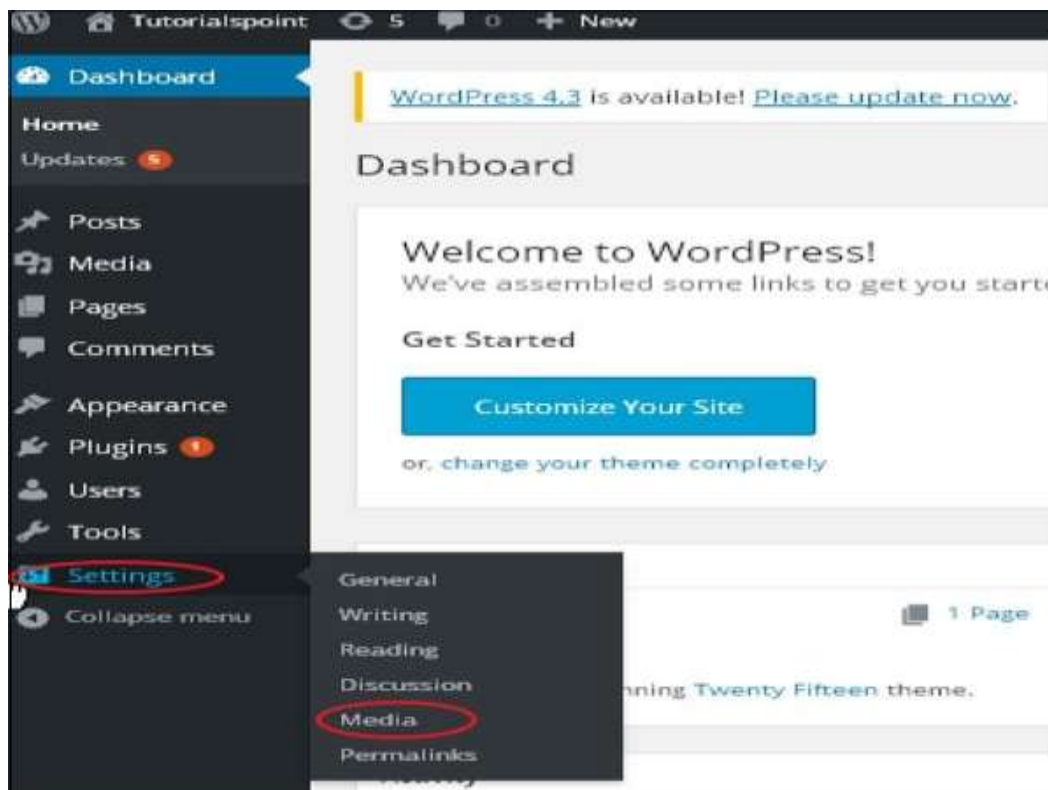
- Maximum rating – You have a four other options of avatars you can use. They are G, PG, R and X. This is the age section where you select according to which type of audience you want to display your posts.
- Default Avatar – In this option, there are few more types of avatars with images; you can keep these avatars according to your visitors e-mail address.

Step (3) – Click on Save Changes button to save the changes.

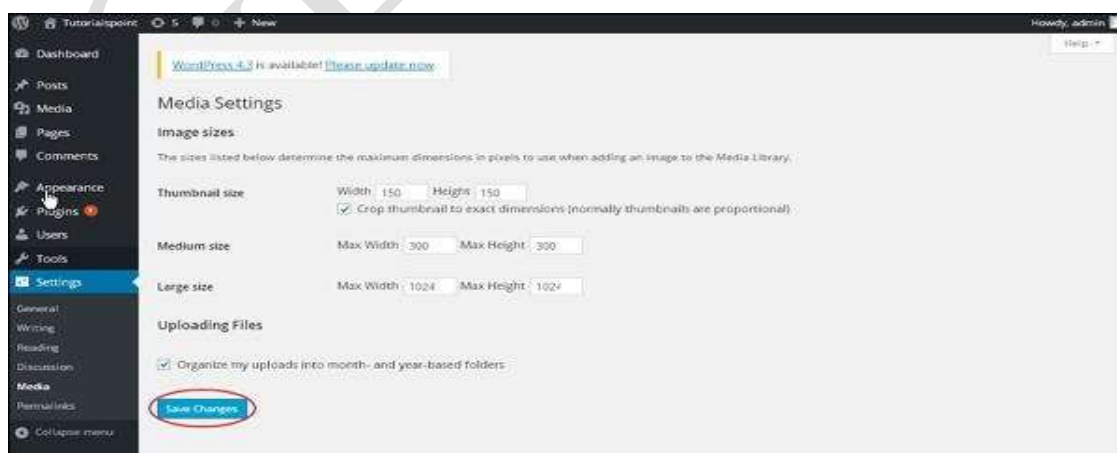
- **Media Setting**

It is used to set the height and width of the images which you're going to use on your website.

Step (1) – Click on Settings → Media option in WordPress.



Step (2) – The Media Settings page is displayed as seen in the following screenshot.



Following are the details of the fields on Media settings –

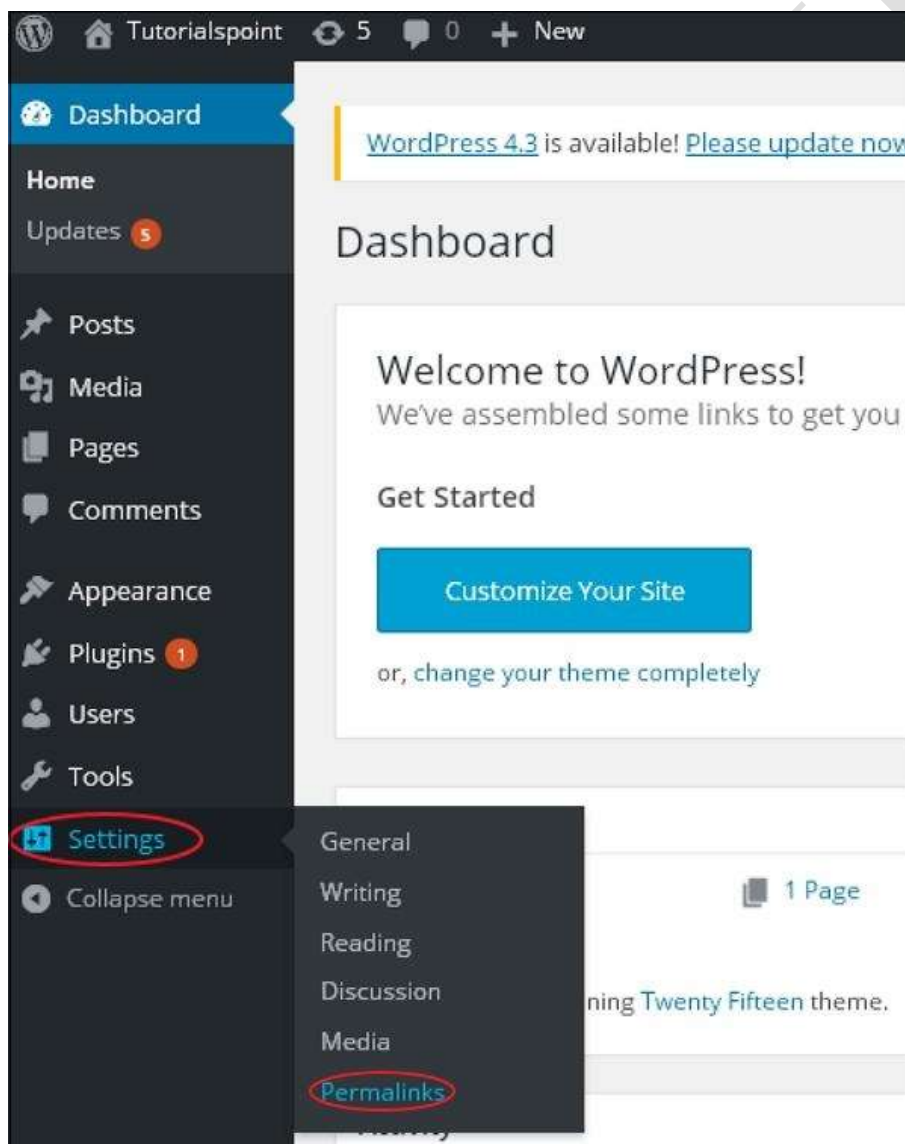
- Thumbnail size – Set the size of the thumbnail.
- Medium size – Set the height and width of medium size images.
- Large size – Set width and height of larger images.
- Uploading files – After checking this checkbox, the uploaded image will be arranged into year and month based folder.

Step (3) – After setting the dimension in pixels, click on Save Changes button. It saves your media setting information.

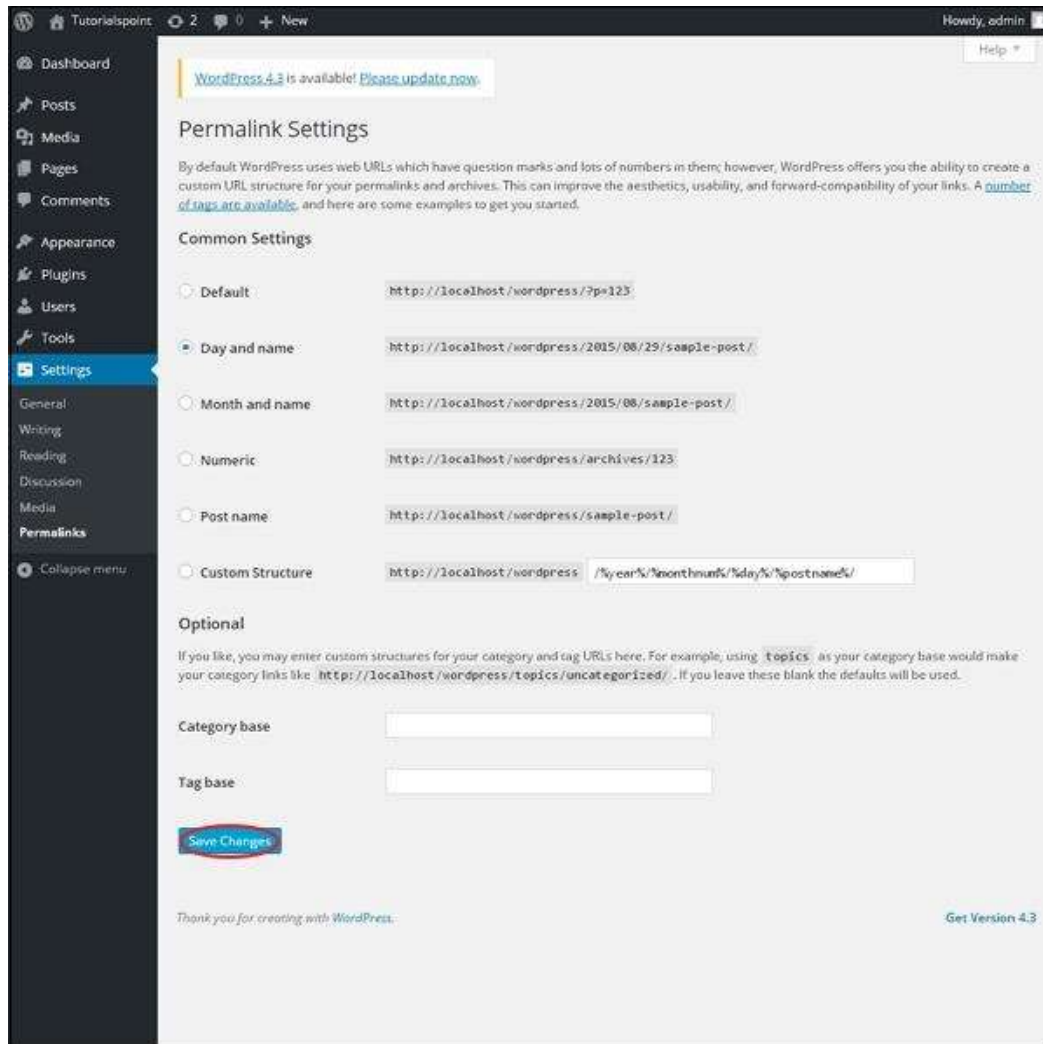
- **Permalinks Setting**

Permalink is a permanent link to a particular blog post or category. It allows setting the default permalink structure. These settings are used to add permalinks to your posts in WordPress. Following are the steps to access permalink settings.

Step (1) – Click on Settings → Permalinks option from the left navigation menu.



Step (2) – When you click on Permalinks, the following page appears on the screen.



Here are a few settings you can make –

- **Common settings –**

Check any of the radio buttons to choose your permalink structure for your blogs

- Default – It sets the default URL structure in Wordpress.
- Day and name – It sets URL structure according to the date and name in your posts.
- Month and name – It sets the URL structure according to the month and name in your post.
- Numeric – It sets numbers in the URL structure in your post.
- Post name – It sets post name in the URL structure in your post.
- Custom Structure – It sets the URL structure of your choice by writing the desired name in the given text box.

- **Optional**

These are optional. You can add custom structure for main category or tag URL. If your text box is empty then default settings is used. Here you have two options.

- Category Base – Add custom prefix for your category URL.

- Tag Base – Add custom prefix to your Tags URL.

Step (3) – Once you are done with changes, click on Save Changes button to save the permalink settings.

Upgrading WordPress Core Manually

1. First create a full backup of your website. This is very important in case you make a mistake.
2. Download the newest WordPress ZIP file from wordpress.org.
3. Unzip the file into a directory on your local machine or in a separate directory on your website.
4. Deactivate all of the plugins on your WordPress site.
5. Go to your website root directory and delete your 'wp-includes' and 'wp-admin' directories. You can do this via sFTP or via SSH.
6. Upload (or copy over) the new wp-includes and wp-admin directories from the new version of WordPress you unzipped to your website root directory to replace the directories you just deleted.
7. Don't delete your wp-content directory or any of the files in that directory. Copy over the files from the wp-content directory in the new version of WordPress to your existing wp-content directory. You will overwrite any existing files with the same name. All of your other files in wp-content will remain in place.
8. Copy all files from the root ('/') directory of the new version of WordPress that you unzipped into your website root directory (or the root directory of your WordPress installation). You will overwrite any existing files and new files will also be copied across. Your wp-config.php file will not be affected because WordPress is never distributed with a wp-config.php file.
9. Examine the wp-config-sample.php which is distributed with WordPress to see if any new settings have been added that you may want to use or modify.
10. If you are upgrading manually after a failed auto-update, remove the .maintenance file from your WordPress root directory. This will remove the 'failed update' message from your site.
11. Visit your main WordPress admin page at /wp-admin/ where you may be asked to sign-in again. You may also have to upgrade your database and will be prompted if this is needed. If you can't sign-in, try clearing your cookies.
12. Re-enable your plugins which you disabled earlier.
13. Clear your browser cache to ensure you can see all changes. If you are using a front-end cache like 'varnish' you should also clear that to ensure that your customers can see the newest changes on your site.
14. Your upgrade is now complete and you should be running the newest version of WordPress.

Upgrading WordPress Plugins Manually

1. First back-up your WordPress site if you haven't already.
2. Download a ZIP file of the plugin you need to upgrade. You can usually find most plugins on the plugin repository along with a link to download the newest ZIP file.
3. Unzip the plugin onto your local machine. It will create a directory called 'plugin-name' with all the files under it.
4. Use sFTP to delete the existing plugin directory from the wp-content/plugins/ directory on your site.
5. Replace the deleted directory by uploading the unzipped plugin to the wp-content/plugins/ directory leaving it in a directory that looks like (for example) wp-content/plugins/plugin-name
6. Sign in to your WordPress site. Go to the 'Plugins' menu and verify that the plugin you upgraded is the newest version.

Upgrading WordPress Themes Manually

Note, if you are running a customized theme you will lose any customizations if you simply overwrite your theme with a new one. In this case you will need to work with a developer to integrate your customizations into your new WordPress theme instead of just overwriting the old theme. If you have not customized your theme's code and have merely customized it using the web admin interface that WordPress provides (without changing any of its files), then you can follow the procedure below:

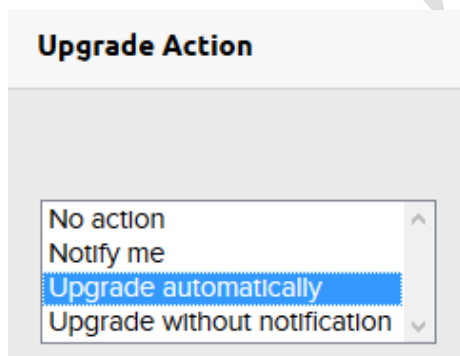
1. First create a backup of your WordPress site.
2. Download a ZIP file of the theme you plan to upgrade.
3. Unzip the theme files onto your local machine.
4. Use sFTP to delete your existing theme directory from the wp-content/themes/ directory.
5. Replace the deleted directory by uploading the unzipped theme into your wp-content/themes/ directory. You should now have a structure that looks something like wp-content/themes/theme-name/
6. Sign into your WordPress site. Go to Appearance > Themes and verify you are running the newest version of your theme.

One-Click WordPress installation to automatically update

You can use either the One-Click Installs page in the panel or log into your WordPress panel to set up automatic upgrades.

To use the One-Click Install feature in the panel to request upgrade notifications:

1. Open your panel.
2. Choose your domain.
3. Click the edit link under the 'Upgrade Action' column.



4. Select the Upgrade automatically option.

Please note that all One-Click Installs require that the domain they are being installed onto is set to Fully Hosted.

- When you request automatic upgrades, DreamHost updates the packages in the installer whenever they become available and notifies you via email.

- You can also update your site by logging into your WordPress dashboard. From there, you can update your core WordPress install, plugins, and themes.

WordPress Database Structure

WordPress is written using PHP as its scripting language and MySQL as its database management system. For using WordPress you don't need to learn php & MySQL, but if you have basic understanding of how it works that will be enough to solve any problems that you will get into. Here in this article you will learn about WordPress database structure.

Default installation of WordPress comes with eleven tables. These are the following tables.

1. **wp_commentmeta**
 - Each comment features information called the meta data and it is stored in the wp_commentmeta.
2. **wp_comments**
 - The comments within WordPress are stored in the wp_comments table.
3. **wp_links**
 - The wp_links holds information related to the links entered into the Links feature of WordPress.
4. **wp_options**
 - The Options set under the Administration > Settings panel are stored in the wp_options table.
5. **wp_postmeta**
 - Each post features information called the meta data and it is stored in the wp_postmeta. Some plugins may add their own information to this table.
6. **wp_posts**
 - The core of the WordPress data is the posts. It is stored in the wp_posts table. Also Pages and navigation menu items are stored in this table.
7. **wp_terms**
 - The categories for both posts and links and the tags for posts are found within the wp_terms table.
8. **wp_term_relationships**
 - Posts are associated with categories and tags from the wp_terms table and this association is maintained in the wp_term_relationships table. The association of links to their respective categories are also kept in this table.
9. **wp_term_taxonomy**
 - This table describes the taxonomy (category, link, or tag) for the entries in the wp_terms table.
10. **wp_usermeta**
 - Each user features information called the meta data and it is stored in wp_usermeta.
11. **wp_users**
 - The list of users is maintained in table wp_users.

Above are the eleven tables, wp_ is the database prefix, we can change database prefix while installing WordPress.

UNIT 3

Theme

What is a WordPress Theme?

A WordPress theme provides all of the front end styling of your WordPress site.

Most WordPress themes provide:

- the overall design or style of your site
- font styling
- colors
- widget locations
- page layouts (or templates)
- styles for blog posts and blog archives
- additional stylistic details

Themes take the content and data stored by WordPress and display it in the browser. When you create a WordPress theme, you decide how that content looks and is displayed. There are many options available to you when building your theme. For example:

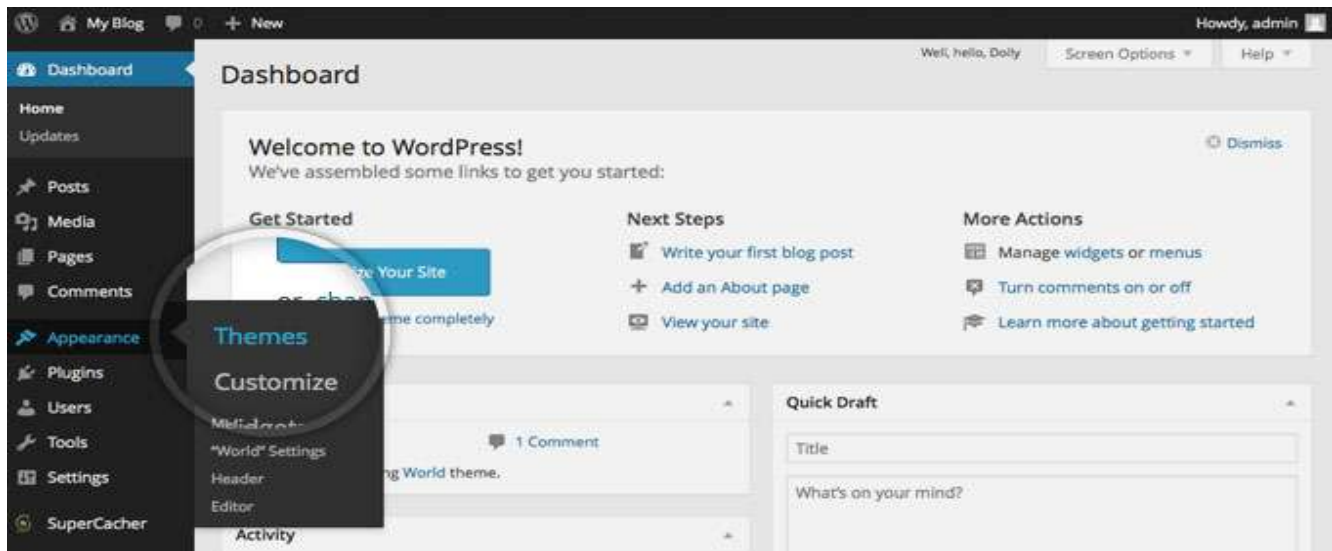
- Your theme can have different layouts, such as static or responsive, using one column or two.
- Your theme can display content anywhere you want it to be displayed.
- Your theme can specify which devices or actions make your content visible.
- Your theme can customize its typography and design elements using CSS.
- Other design elements like images and videos can be included anywhere in your theme.

WordPress themes are incredibly powerful. But, as with every web design project, a theme is more than color and layout. Good themes improve engagement with your website's content in addition to being beautiful.

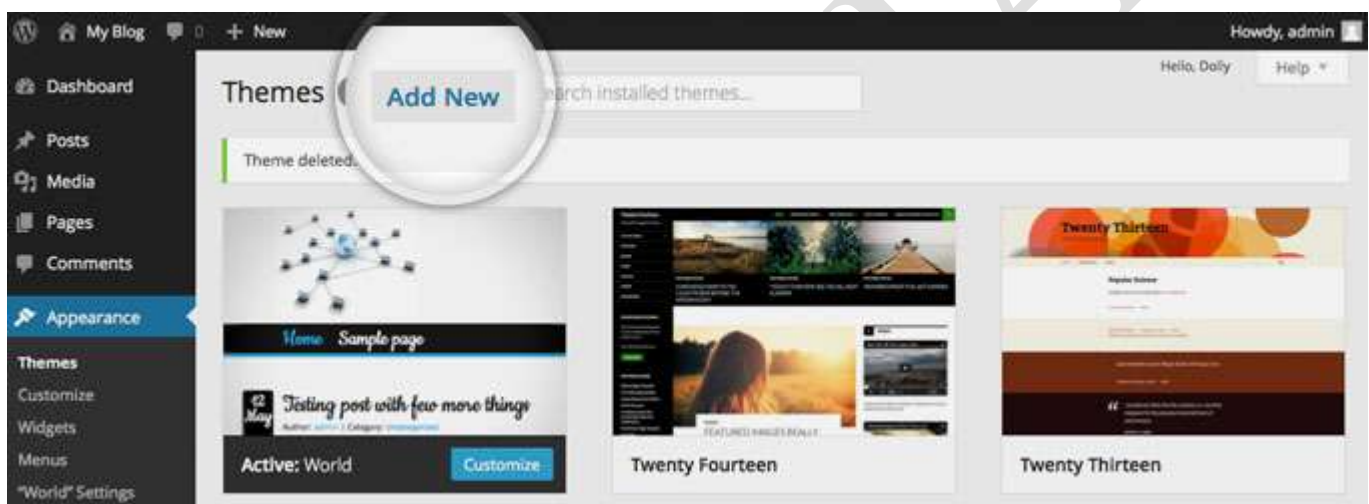
How to install & activate theme

Access install theme functionality in your WP admin

The first thing you need to do when you want to install a new WordPress theme is to login to your site admin page. Once there, go to Appearance -> Themes.



Here, you will see all the themes you have currently installed in your application. To add another one, simply click on the Install Themes tab.

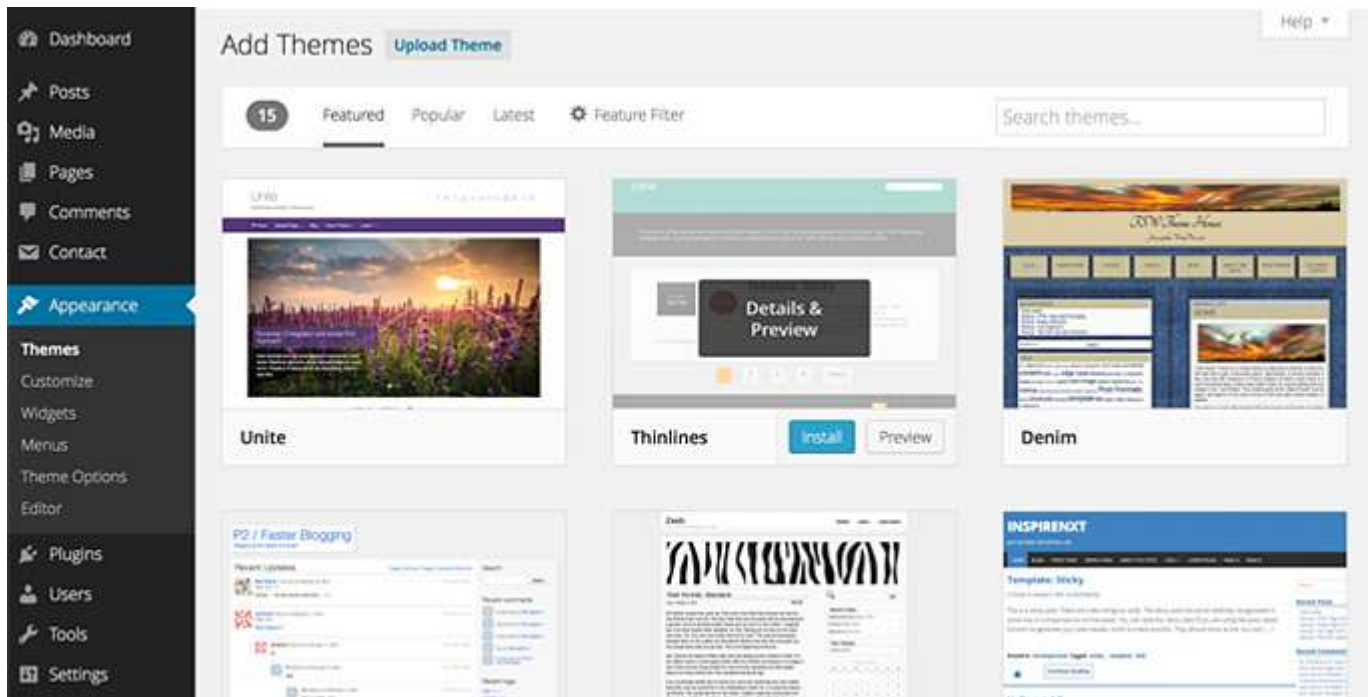


On this page there are two ways to add a new theme. You can either use the search field which will return results from the WordPress.org theme directory or you can upload a theme from your local hard drive. In this tutorial we will show you how to add themes in both ways.

Choose and Install a theme from the official WordPress theme directory

The easiest way to install themes to your WordPress site is if they are listed in the official themes directory. This allows you to search for the theme you need directly from your site admin page. Each theme in the official directory has to have Tags that describe its functionality allowing you to easily search for the right theme.

If you know the theme's name, you can simply search for it.



However, usually that's not the case. This is why, you can use the Feature Filter. For example, you can search for a Black and White, Two columns theme that has Flexible Width. Simply check those tags and press the Apply Filters button.

You will now see all themes that meet your search. Hover over any of them and you will see two options - to see a demo of the theme or to install it. Once you choose which theme you want to use for your site, press the blue Install button.

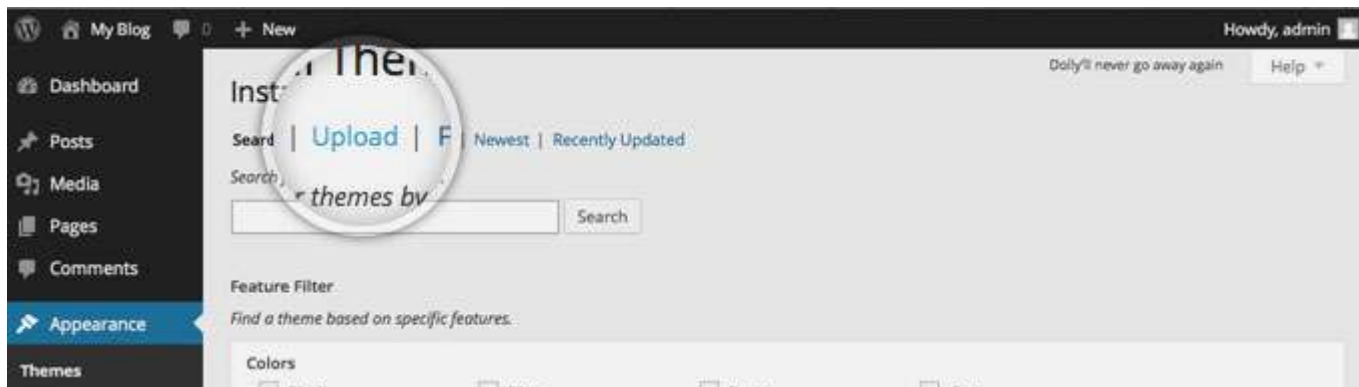
Your WordPress application will download and install the theme for you. Simply click the Activate link on the next page you will be redirected to.



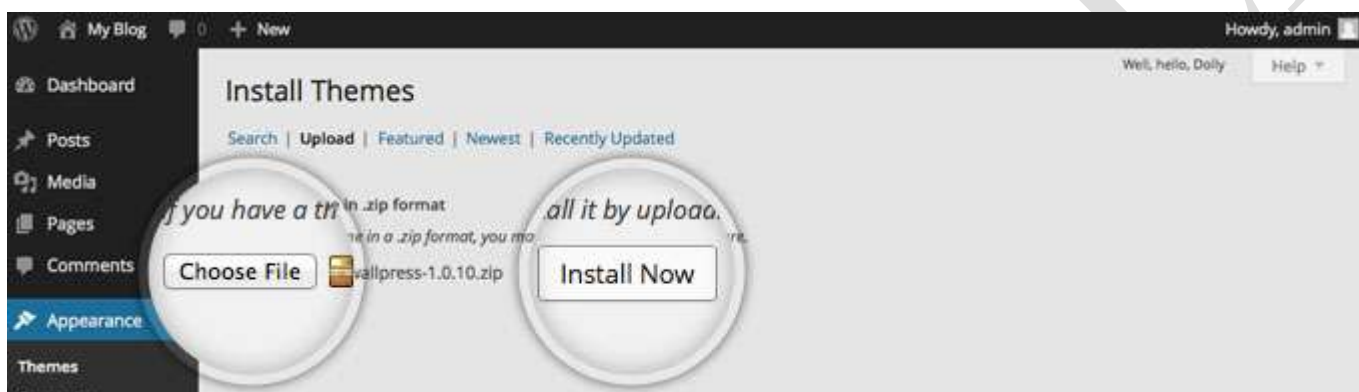
That's all - you can go to the front page of your site to see its new looks.

Upload a theme you have already downloaded

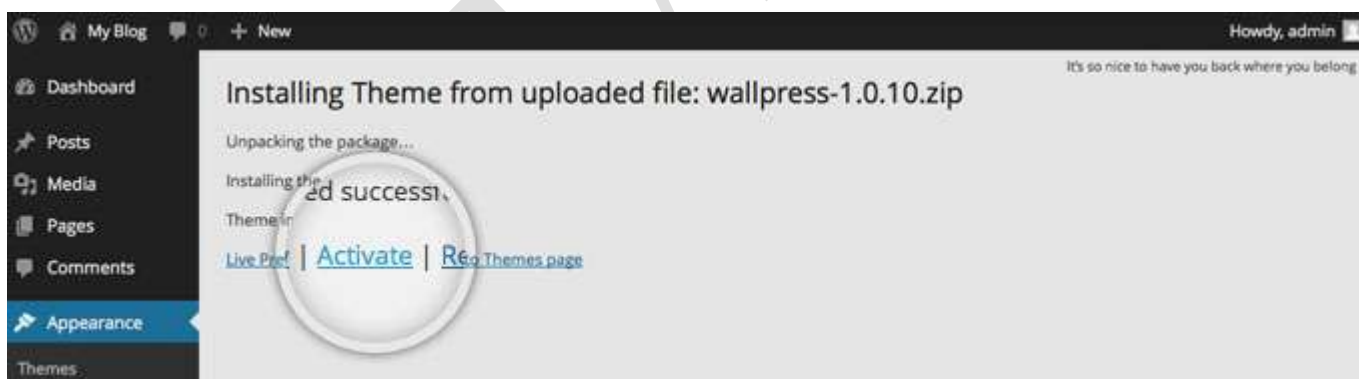
If you have a theme that's not listed in the WordPress.org directory, you can simply upload it from the Themes -> Install Themes section in WordPress. To do this click on the Upload link at the top of the page.



Now, click the Choose File button, select the archive of your theme from your local computer and press the Install Now button.



In few seconds, WordPress will upload and extract the theme archive for you. All you need to do next is to press the Activate link under the message for successful theme installation.



That's it! You can now go to the front end of your site and see the newly activated theme.

Introduction of common WordPress theme template files

Template Files List

Here is the list of the Theme files recognized by WordPress. Of course, your Theme can contain any other stylesheets, images, or files. Just keep in mind that the following have special meaning to WordPress -- see Template Hierarchy for more information.

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.

These files have a special meaning with regard to WordPress because they are used as a replacement for index.php, when available, according to the Template Hierarchy, and when the corresponding Conditional Tag returns true. For example, if only a single post is being displayed, the `is_single()` function returns 'true', and, if there is a single.php file in the active Theme, that template is used to generate the page.

Basic Templates

At the very minimum, a WordPress Theme consists of two files:

- style.css
- index.php

Both of these files go into the Theme directory. The index.php template file is very flexible. It can be used to include all references to the header, sidebar, footer, content, categories, archives, search, error, and any other page created in WordPress.

Or, it can be divided into modular template files, each one taking on part of the workload. If you do not provide other template files, WordPress may have default files or functions to perform their jobs. For example, if you do not provide a searchform.php template file, WordPress has a default function to display the search form.

Typical template files include:

- comments.php
- comments-popup.php
- footer.php
- header.php
- sidebar.php

Using these template files you can put template tags within the index.php master file to include these other files where you want them to appear in the final generated page.

- To include the header, use `get_header()`.
- To include the sidebar, use `get_sidebar()`.
- To include the footer, use `get_footer()`.
- To include the search form, use `get_search_form()`.

Here is an example of the include usage:

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

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

Widget

What is widget ?

WordPress Widgets add content and features to your Sidebars. Examples are the default widgets that come with WordPress; for Categories, Tag cloud, Search, etc. Plugins will often add their own widgets.

Widgets were originally designed to provide a simple and easy-to-use way of giving design and structure control of the WordPress Theme to the user, which is now available on properly "widgetized" WordPress Themes to include the header, footer, and elsewhere in the WordPress design and structure. Widgets require no code experience or expertise. They can be added, removed, and rearranged on the Theme Customizer or Appearance > Widgets in the WordPress Administration Screens.

Some WordPress Widgets offer customization and options such as forms to fill out, includes or excludes of data and information, optional images, and other customization features.

The Appearance Widgets Screen explains how to use the various Widgets that come delivered with WordPress.

Plugins that come bundled with widgets can be found in the WordPress Plugin Directory.

Installing Widgets

WordPress comes pre-packaged with a variety of Glossary#Widget Widgets. If those are insufficient for your needs you can install new ones by searching the WordPress Plugin Directory which is accessible from the WordPress Administration Plugins > Add New Screen.

Displaying Widgets

Existing Widgets in Existing Widget Areas

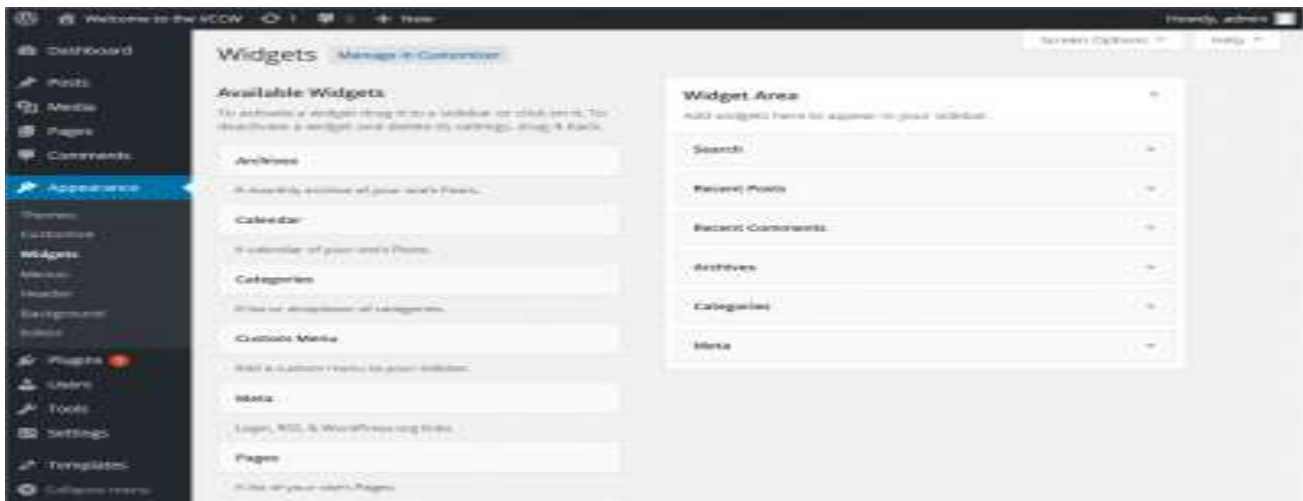
Before you can add a Widget you must verify that the Theme you're using supports Widgets (more specifically: Glossary#Widget_Area Widget Areas). You can do so by simply navigating to the Appearance menu and looking for a sub menu titled "Widgets".

If your Theme supports Theme Customizer then you can use the following Steps. In Theme Customizer, the live preview of changes is available.



1. Go to Appearance > Customize in the WordPress Administration Screens.
2. Click the Widget menu in the Theme Customizer to access to the Widget Customize Screen.
3. Click the down arrow of Widget Area to list the already registered Widgets.
4. Click Add a Widget button at the bottom of sidebar. It shows the list of available widgets.
5. Click a widget you want to add. The widgets should be added in the sidebar.
6. Preview your site and you should see the content from your new Widget.
7. To arrange the Widgets within the Sidebar, drag and drop the widgets in the order you want or click Reorder link and click up arrow and down arrow of each widget and click Done after the arrange operation.
8. To customize the Widget features, click the down arrow in the right to expand the Widget's interface.
9. To remove the widget, click Remove from Widget's interface in above step.

If your Theme does not support Theme Customizer then you can use the following conventional steps:



1. Go to Appearance > Widgets in the WordPress Administration Screens.
2. Choose a Widget and either drag it to the sidebar where you wish it to appear, or click the widget, (select a destination sidebar if your theme has more than one) and click the Add Widget button. There might be more than one sidebar option, so begin with the first one. Once in place, WordPress automatically updates the Theme.
3. Preview the site. You should find that the "default" sidebar elements are now gone and only the new addition is visible.
4. Return to the Widgets Screen to continue adding Widgets.
5. To arrange the Widgets within the sidebar or Widget area, click and drag it into place.
6. To customize the Widget features, click the down arrow in the upper right corner to expand the Widget's interface.
7. To save the Widget's customization, click Save.
8. To remove the Widget, click Delete.

If you want to remove the widget but save its setting for possible future use, just drag it into the Inactive Widgets area. You can add them back anytime from there. This is especially helpful when you switch to a theme with fewer or different widget areas.

When changing themes, there is often some variation in the number and setup of widget areas/sidebars and sometimes these conflicts make the transition a bit less smooth. If you changed themes and seem to be missing widgets, scroll down on the screen to the Inactive Widgets area, where all of your widgets and their settings will have been saved.

Enabling Accessibility Mode, via Screen Options, allows you to use Add and Edit buttons instead of using drag and drop.

Widget Areas

While widget areas typically occur in webpage sidebars, a theme can place widget areas anywhere on a page. For example, besides the usual sidebar locations, the Twenty Fourteen theme has a widget area in the footer of every page.

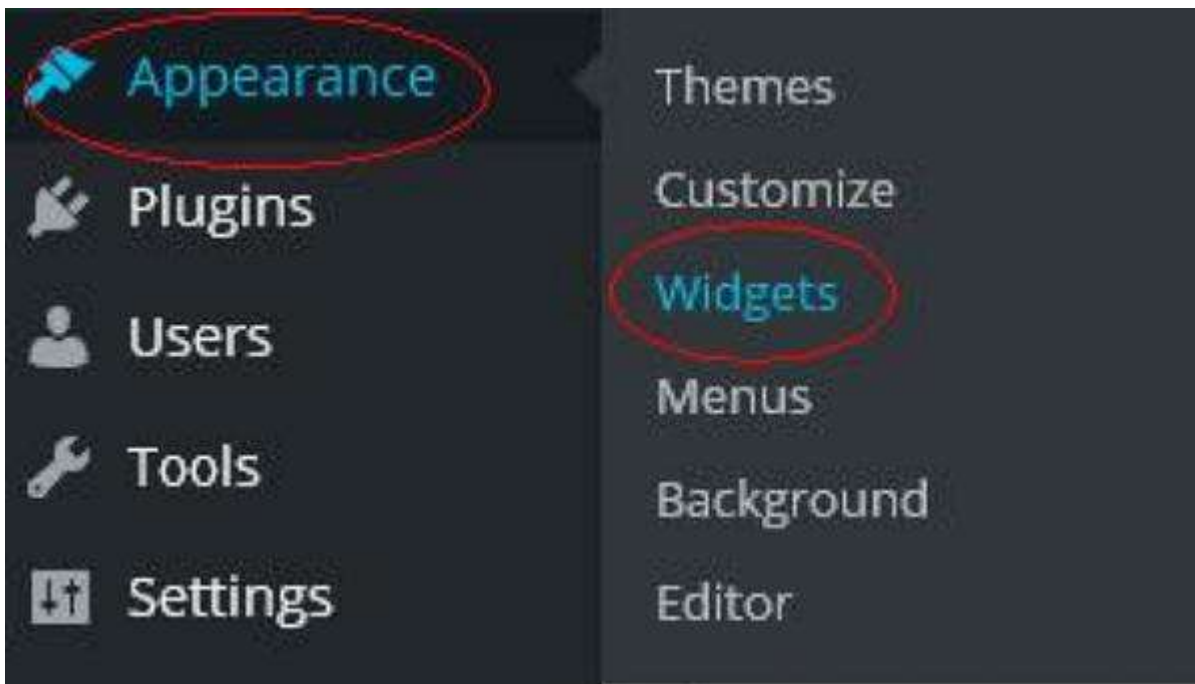
If you would like to place a Widget somewhere on your Theme that does not have a pre-defined Widget Area, you will need some programming knowledge and should follow the instructions on the Widgets API section found [here](#).

Widget Management

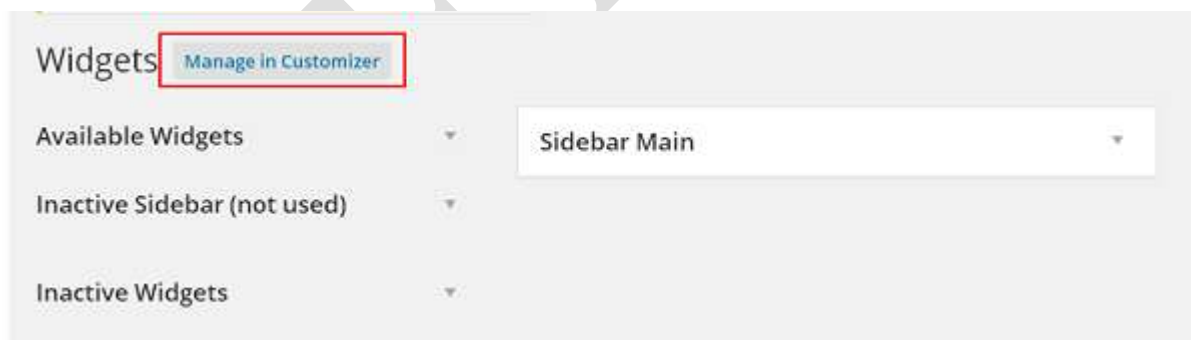
Widgets are small blocks that perform specific functions. These give design and structure control to the WordPress theme. Some specific features of a widget are –

- They help you add content and features.
- They can be easily dragged and dropped in widget area.
- They vary from theme to theme. They are not same for every theme.

Step (1) – Click on Appearance → Widgets.



Step (2) – The following screen showing available widgets appear.



The following functions appear on the page –

- Available Widgets – You can use these to add into your sidebar main.
- Inactive Sidebar (not used) – These are not used and can be removed permanently from the widget list.
- Inactive Widgets – Removes the widgets from sidebar but keep it in the settings.
- Sidebar Main – Any widget you add here will appear on your site.
- Manage in Customizer – Takes you back to customization page.

Available Widgets

To activate a widget drag it to a sidebar or click on it. To deactivate a widget and delete its settings, drag it back.

Archives

A monthly archive of your site's Posts.

Calendar

A calendar of your site's Posts.

Categories

A list or dropdown of categories.

Custom Menu

Add a custom menu to your sidebar.

Meta

Login, RSS, & WordPress.org links.

Pages

A list of your site's Pages.

Recent Comments

Your site's most recent comments.

Recent Posts

Your site's most recent Posts.

RSS

Entries from any RSS or Atom feed.

Search

A search form for your site.

Tag Cloud

A cloud of your most used tags.

Text

Step (3) – Drag and drop in the Sidebar Main. Any widget you add here, shows up on your site.



Plugin

Introduction

Plugins are ways to extend and add to the functionality that already exists in WordPress.

The core of WordPress is designed to be lean and lightweight, to maximize flexibility and minimize code bloat. Plugins then offer custom functions and features so that each user can tailor their site to their specific needs.

For instructions and information on downloading, installing, upgrading, troubleshooting, and managing your WordPress Plugins, see Managing Plugins. If you want to develop your own plugin, there is a comprehensive list of resources in Plugin Resources.

For instructions and information on downloading, installing, upgrading, troubleshooting, and managing your WordPress Plugins, see Managing Plugins. If you want to develop your own plugin, there is a comprehensive list of resources in Plugin Resources.

Plugin Repositories

WordPress Plugins are available from several sources. The most popular and official source for WordPress Plugins is the WordPress.org repo.

- Official WordPress Plugins Repository

Just to note, not all WordPress Plugins make it into the above repository. Search the web for "WordPress Plugin" and the keywords for the type of functionality you are seeking. There is bound to be a solution out there for you.

Default Plugins

The following two plugins are included with WordPress core:

Akismet

Akismet checks your comments against the Akismet web service to see if they look like spam or not. You can review the spam it catches under "Manage" and it automatically deletes old spam after 15 days.

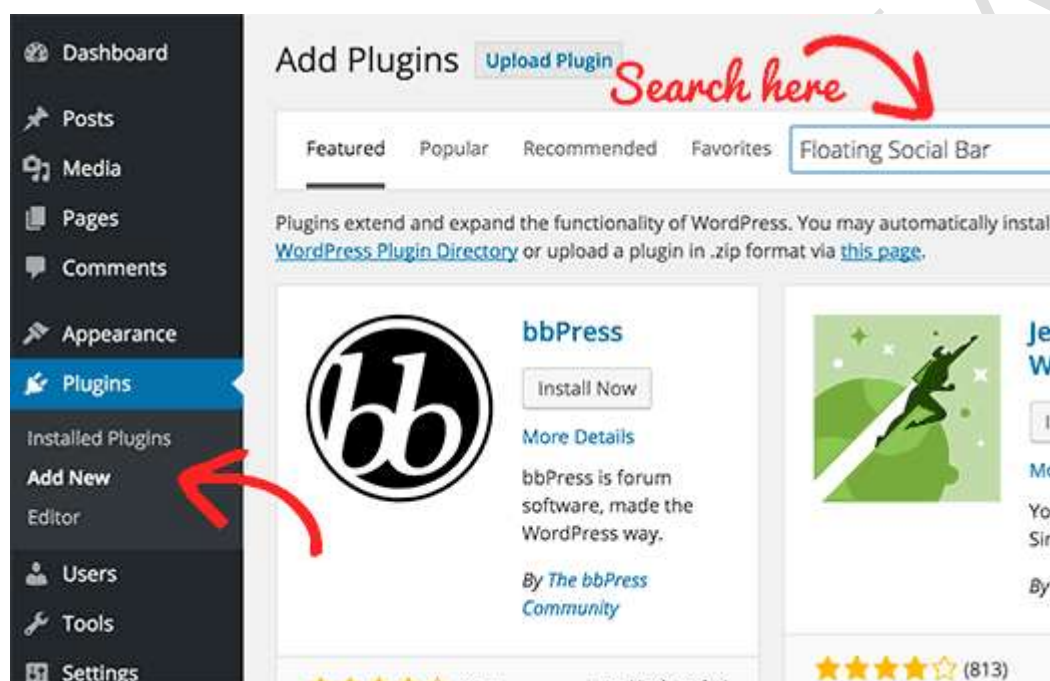
Install and activate plugin

There are three methods: installing a WordPress plugin using search, uploading a WordPress plugin, and manually installing a WordPress plugin using FTP.

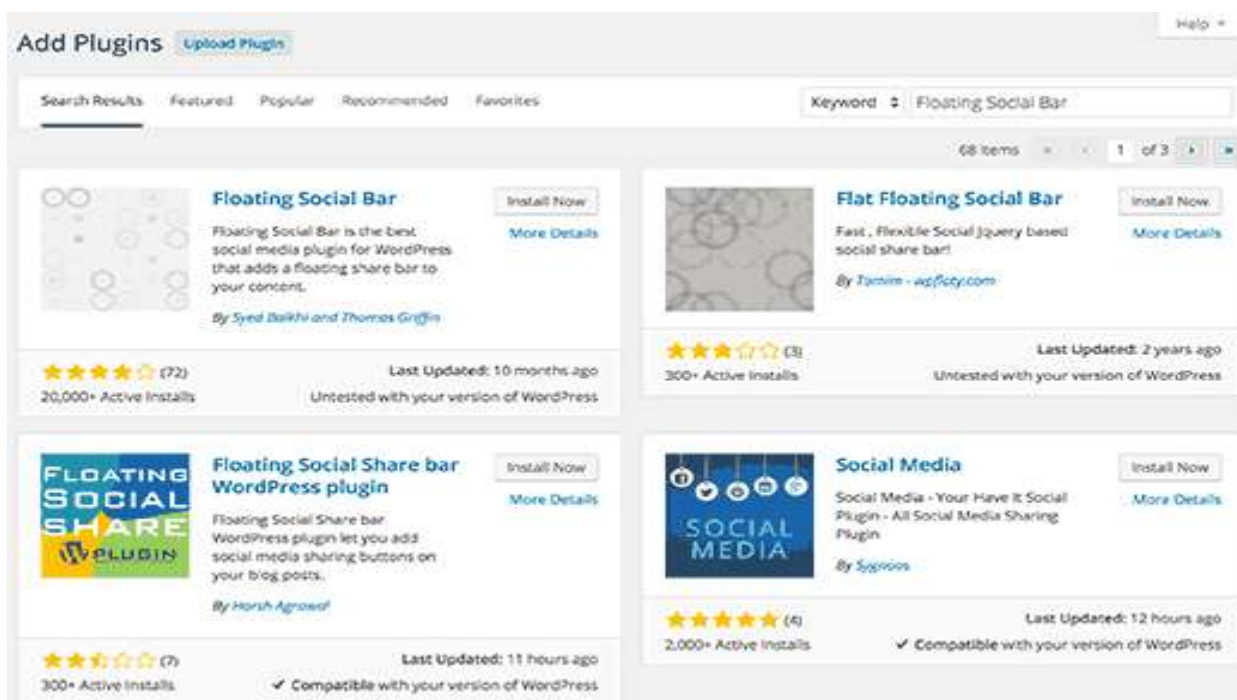
Install a Plugin using WordPress Plugin Search

The easiest way of installing a WordPress plugin is to use the plugin search. The only downside of this option is that a plugin must be in the [WordPress plugin directory](#) which is limited to only free plugins.

First thing you need to do is go to your WordPress [admin area](#) and click on Plugins » Add New.



You will see a screen like the one in the screenshot above. Find the plugin by typing the plugin name or the functionality you are looking for, like we did. After that, you will see a bunch of listings like the example below:



You can pick the plugin that is best for you. Since in our search, we were looking for [Floating Social Bar](#) which happens to be the first plugin, we will click the 'Install Now' button.

WordPress will now download and install the plugin for you. After this, you will see the success message with a link to activate the plugin or return to plugin installer.

Installing Plugin: Floating Social Bar 1.1.7

Downloading install package from <https://downloads.wordpress.org/plugin/floating-social-bar-1.1.7>

Unpacking the package...

Installing the plugin...

Successfully installed the plugin Floating Social Bar 1.1.7.

[Activate Plugin](#) | [Return to Plugin Installer](#)

A WordPress plugin can be installed on your site, but it will not work unless you activate it. So go ahead and click on the activate plugin link to activate the plugin on your WordPress site.

That's all, you have successfully installed your first WordPress plugin.

The next step is to configure the plugin settings. These settings will vary for each plugin therefore we will not be covering that in this post.

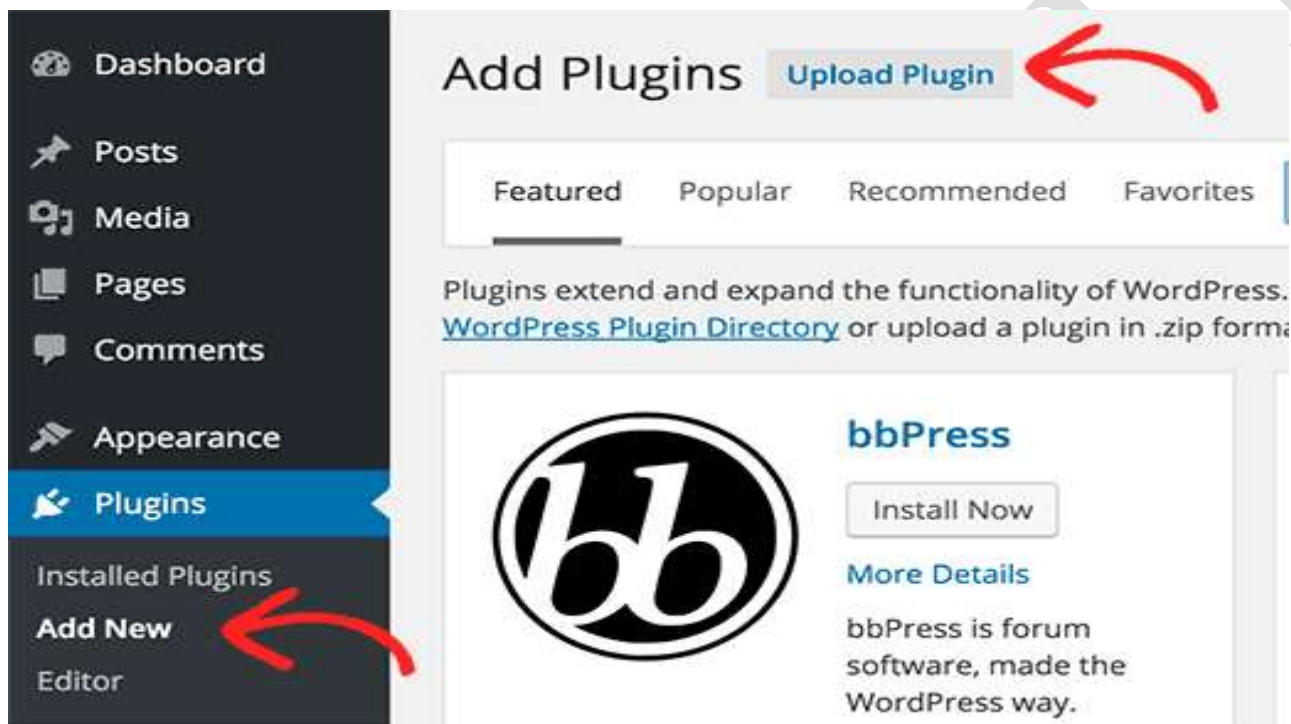
Install a Plugin using the WordPress Admin Plugin Upload

Paid WordPress plugins are not listed in the WordPress plugin directory. These plugins cannot be installed using the first method.

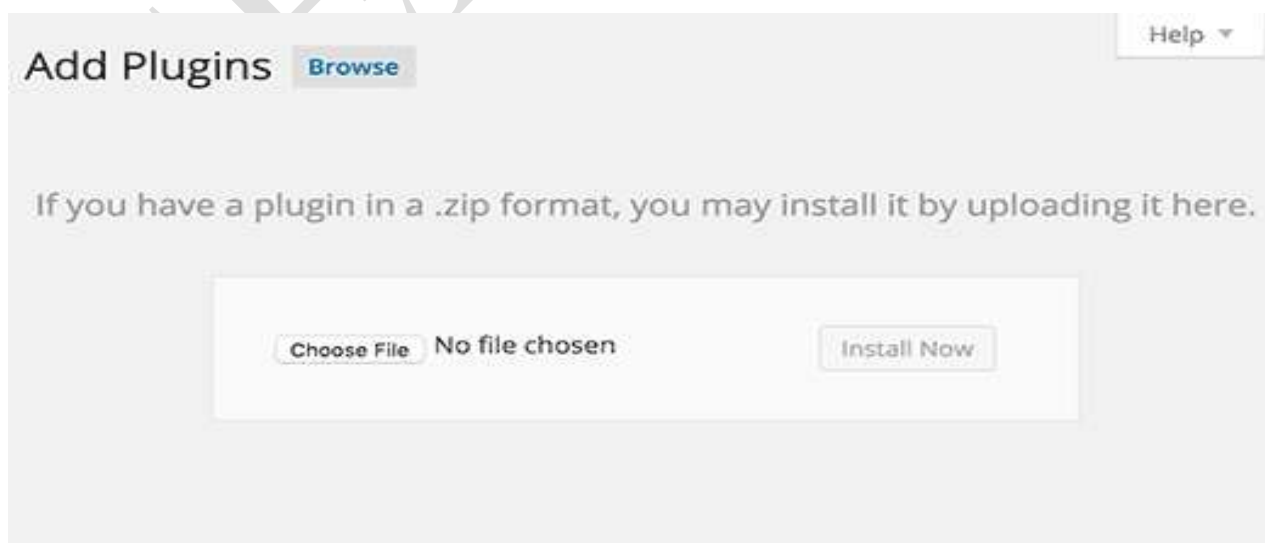
That's why WordPress has the Upload method to install such plugins. We will show you how to install WordPress plugin using the upload option in the admin area.

First, you need to download the plugin from the source (which will be a zip file). Next, you need to go to WordPress admin area and visit Plugins » Add New page.

After that, click on the Upload Plugin button on top of the page.

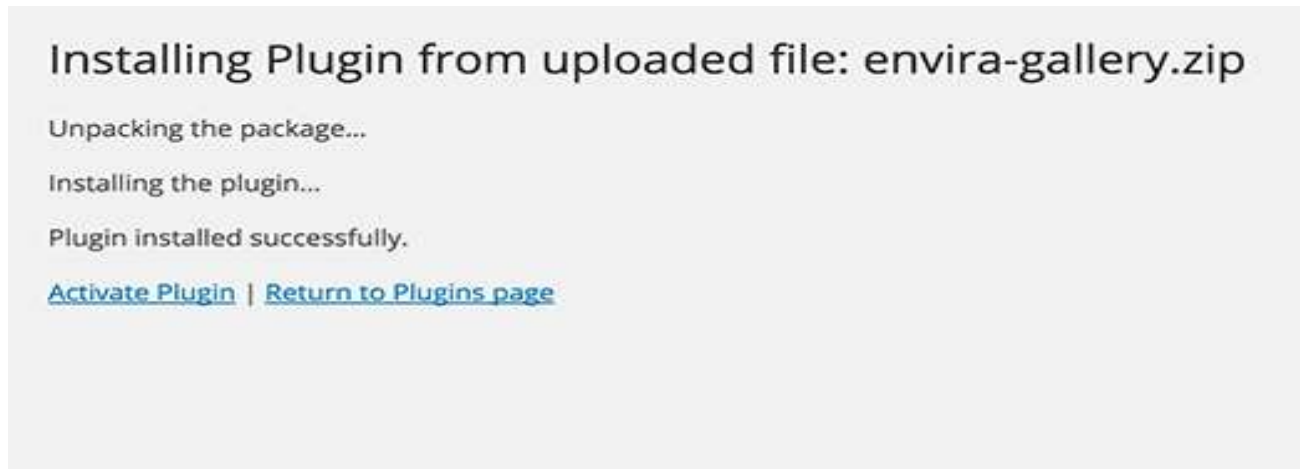


This will bring you to the plugin upload page. Here you need to click on the choose file button and select the plugin file you downloaded earlier to your computer.



After you have selected the file, you need to click on the install now button.

WordPress will now upload the plugin file from your computer and install it for you. You will see a success message like this after installation is finished.



Once installed, you need to click on the Activate Plugin link to start using the plugin.

You would have to configure the settings to fit your needs. These settings will vary for each plugin therefore we will not be covering that in this post.

Manually Install a WordPress Plugin using FTP

In some cases, your WordPress hosting provider may have file restrictions that could limit your ability to install a plugin from the admin area.

In this situation, your best bet is to install the plugin manually using FTP.

The FTP manager method is the least friendly for beginners.

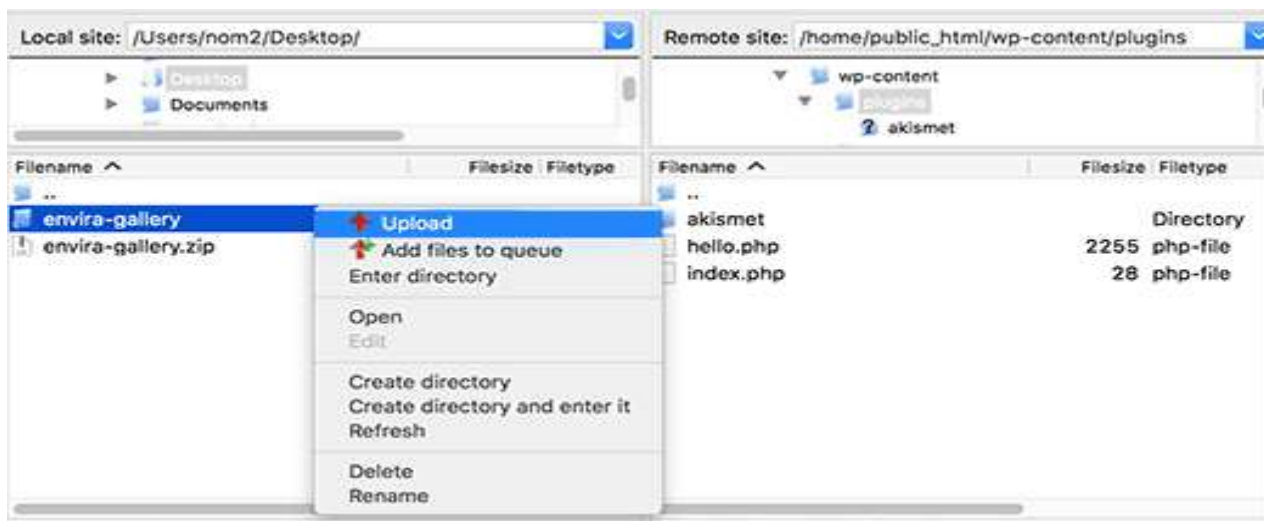
First you will need to download the plugin's source file (it will be a zip file). Next, you need to extract the zip file on your computer.

Extracting the plugin zip file will create a new folder with the same name. This is the folder that you need to manually upload to your website using a FTP client.

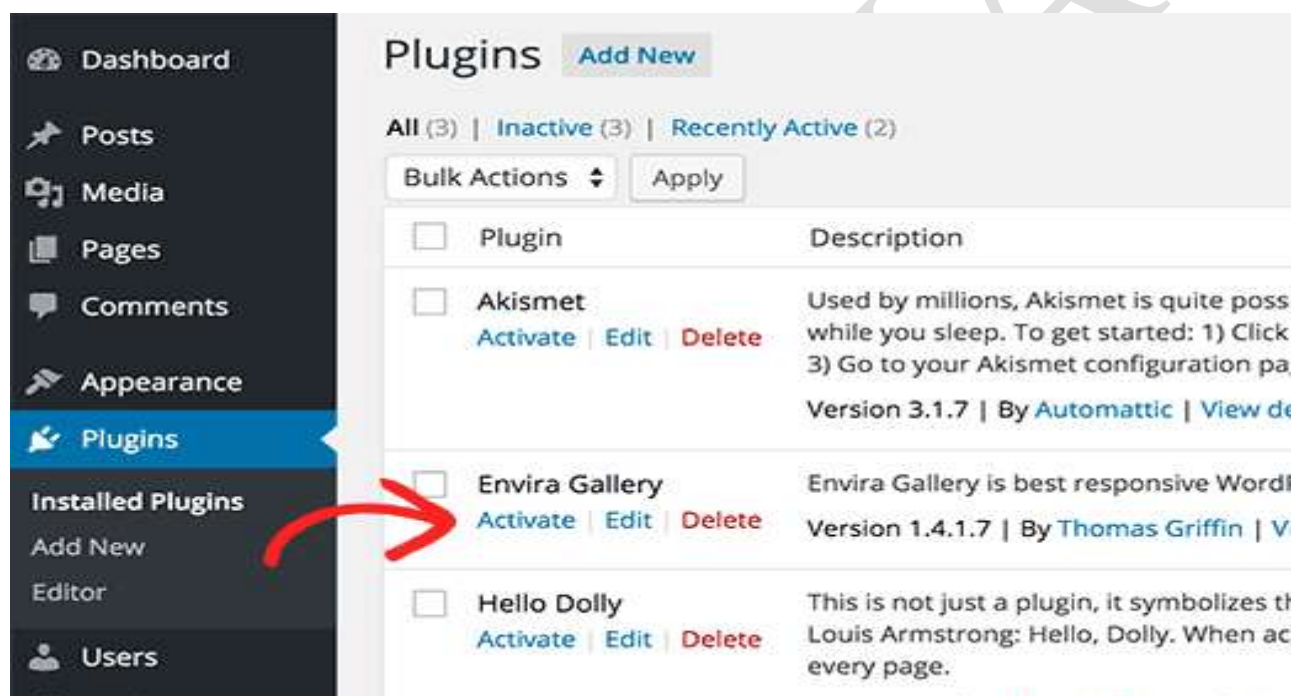
You would need to access your host through the FTP manager. If you do not have your FTP username and password, then contact your WordPress hosting provider and ask them.

Open the FTP client on your computer and connect to your website using the login credentials provided by your web host. Once connected, you need to access the path /wp-content/plugins/

Next, upload the folder you extracted from the zip file to the /wp-content/plugins/ folder on your web server.



After uploading the files, you need to visit the WordPress admin area and click on the Plugins link in the admin menu. You will see your plugin successfully installed on the plugins page.



You need to click on the Activate link below the plugin. Upon activating, you may need to configure the plugin settings. WordPress plugins come with their own settings which differ from one plugin to another so we will not describe them here.

Useful plugins for website

Seo yoast

Everyone who is blogging has heard a thing or two about SEO. You know that it is important for your site to have good SEO. WordPress is extremely well-coded which is why many call it SEO friendly. But the real SEO benefit comes from WordPress plugins WordPress SEO by Yoast. This plugin is by far the most complete SEO solution that is available for WordPress. It has received over a million downloads. We use WordPress SEO plugin to improve our on-page SEO.

Unlike most other plugins this plugin is way more than simple meta tag additions. We use it to add custom post title, meta description, and meta keyword for our posts, pages, and taxonomies (tag, category, etc). When writing the meta information, WordPress SEO plugin shows you a Google search result snippet preview, so you can see exactly how your content will look when someone search for it in Google. It even has the ability to get you verified google authorship for your site.

It helps us create XML sitemaps that support images. It also gives us the ability to have advanced configuration such as removing a specific post, page, post type, or a taxonomy from the sitemap. It notifies search engines automatically once your content is published.

Contact form 7

Any Blog, Website or service is incomplete without a contact page. A contact page is basically, where your readers can use to contact you. When it comes to WordPress, there are many contact form WordPress plugin and one of the most simple and smart one is, Contact form 7 plugin. This plugin, is more like activate, copy, paste and ready to go plugin. CF7 plugin is a WordPress Email form plugin which is useful to add contact form on WordPress contact or service pages.

WordPress contact form 7 plugin, comes with multiple contact form options and you can add additional field, for example if you need to get phone number of the sender, you can add another field into your form easily.

Wp contact form 7 Plugin features :

- Multiple contact form
- Spam prevention
- Bot prevention with CAPTCHA
- Customization enabled

Woocommerce

WooCommerce is a free eCommerce plugin that allows you to sell anything, beautifully. Built to integrate seamlessly with WordPress, WooCommerce is the world's favorite eCommerce solution that gives both store owners and developers complete control.

With endless flexibility and access to hundreds of free and premium WordPress extensions, WooCommerce now powers 30% of all online stores — more than any other platform.

With WooCommerce, you can sell both physical and digital goods in all shapes and sizes, offer product variations, multiple configurations, and instant downloads to shoppers, and even sell affiliate goods from online marketplaces.

WP Super Cache

This plugin generates static html files from your dynamic WordPress blog. After a html file is generated your webserver will serve that file instead of processing the comparatively heavier and more expensive WordPress PHP scripts.

The static html files will be served to the vast majority of your users, but because a user's details are displayed in the comment form after they leave a comment those requests are handled by the legacy caching engine. Static files are served to:

1. Users who are not logged in.
2. Users who have not left a comment on your blog.
3. Or users who have not viewed a password protected post.

99% of your visitors will be served static html files. Those users who don't see the static files will still benefit because they will see different cached files that aren't quite as efficient but still better than uncached. This plugin will help your server cope with a front page appearance on digg.com or other social networking site.

If for some reason "supercaching" doesn't work on your server then don't worry. Caching will still be performed, but every request will require loading the PHP engine. In normal circumstances this isn't bad at all. Visitors to your site will notice no slowdown or difference. Supercache really comes into it's own if your server is underpowered, or you're experiencing heavy traffic.

Super Cached html files will be served more quickly than PHP generated cached files but in every day use, the difference isn't noticeable.

Regenerate Thumbnails

Regenerate Thumbnails allows you to regenerate the thumbnails for your image attachments. This is very handy if you've changed any of your thumbnail dimensions (via Settings -> Media) after previously uploading images or have changed to a theme with different featured post image dimensions.

You can either regenerate the thumbnails for all image uploads, individual image uploads, or specific multiple image uploads.

Advanced Custom Fields

Advanced Custom Fields is the perfect solution for any WordPress website which needs more flexible data like other Content Management Systems.

- Visually create your Fields
- Select from multiple input types (text, textarea, wysiwyg, image, file, page link, post object, relationship, select, checkbox, radio buttons, date picker, true / false, repeater, flexible content, gallery and more to come!)
- Assign your fields to multiple edit pages (via custom location rules)
- Easily load data through a simple and friendly API
- Uses the native WordPress custom post type for ease of use and fast processing
- Uses the native WordPress metadata for ease of use and fast processing.

Unit 4

Theme development

Anatomy of a Theme: header.php, footer.php and sidebar.php

Anatomy of a Theme

WordPress Themes live in subdirectories of the WordPress themes directory (wp-content/themes/ by default) which cannot be directly moved using the wp-config.php file. The Theme's subdirectory holds all of the Theme's stylesheet files, template files, and optional functions file (functions.php), JavaScript files, and images. For example, a Theme named "test" would reside in the directory wp-content/themes/test/. Avoid using numbers for the theme name, as this prevents it from being displayed in the available themes list.

WordPress includes a default theme in each new installation. Examine the files in the default theme carefully to get a better idea of how to build your own Theme files.

For a visual guide, see this info graphic on WordPress Theme Anatomy.

WordPress Themes typically consist of three main types of files, in addition to images and JavaScript files.

1. The stylesheet called style.css, which controls the presentation (visual design and layout) of the website pages.
2. WordPress template files which control the way the site pages generate the information from your WordPress database to be displayed on the site.
3. The optional functions file (functions.php) as part of the WordPress Theme files.

Document Head (header.php)

- Use the proper DOCTYPE.
- The opening <html> tag should include language_attributes().
- The <meta> charset element should be placed before everything else, including the <title> element.
- Use bloginfo() to set the <meta> charset and description elements.
- Use wp_title() to set the <title> element. See why.
- Use Automatic Feed Links to add feed links.
- Add a call to wp_head() before the closing </head> tag. Plugins use this action hook to add their own scripts, stylesheets, and other functionality.
- Do not link the theme stylesheets in the Header template. Use the wp_enqueue_scripts action hook in a theme function instead.

Here's an example of a correctly-formatted HTML5 compliant head area:

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo( 'charset' ); ?>" />
    <title><?php wp_title(); ?></title>
    <link rel="profile" href="http://gmpg.org/xfn/11" />
```

```

<link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>" />
<?php if ( is_singular() && get_option( 'thread_comments' ) ) wp_enqueue_script( 'comment-reply' ); ?>
<?php wp_head(); ?>
</head>

```

Navigation Menus (header.php)

- The Theme's main navigation should support a custom menu with `wp_nav_menu()`.
 - Menus should support long link titles and a large amount of list items. These items should not break the design or layout.
 - Submenu items should display correctly. If possible, support drop-down menu styles for submenu items. Drop-downs allowing showing menu depth instead of just showing the top level.

Widgets (sidebar.php)

- The Theme should be widgetized as fully as possible. Any area in the layout that works like a widget (tag cloud, blogroll, list of categories) or could accept widgets (sidebar) should allow widgets.
- Content that appears in widgetized areas by default (hard-coded into the sidebar, for example) should disappear when widgets are enabled from Appearance > Widgets.

Footer (footer.php)

- Use the `wp_footer()` call, to appear just before closing body tag.

```

<?php wp_footer(); ?>
</body>
</html>

```

Template Files List

Here is the list of the Theme files recognized by WordPress. Of course, your Theme can contain any other stylesheets, images, or files. Just keep in mind that the following have special meaning to WordPress -- see Template Hierarchy for more information.

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.

These files have a special meaning with regard to WordPress because they are used as a replacement for index.php, when available, according to the Template Hierarchy, and when the corresponding Conditional Tag returns true. For example, if only a single post is being displayed, the `is_single()` function returns 'true', and, if there is a single.php file in the active Theme, that template is used to generate the page.

The Loop

The Loop is PHP code used by WordPress to display posts. Using The Loop, WordPress processes each post to be displayed on the current page, and formats it according to how it matches specified criteria within The Loop tags. Any HTML or PHP code in the Loop will be processed on each post.

When WordPress documentation says "This tag must be within The Loop", such as for specific Template Tags or plugins, the tag will be repeated for each post. For example, The Loop displays the following information by default for each post:

- Title (`the_title()`)
- Time (`the_time()`)
- Categories (`the_category()`).

You can display other information about each post using the appropriate Template Tags or (for advanced users) by accessing the `$post` variable, which is set with the current post's information while The Loop is running.

The loop starts here:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
```

and ends here:

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

This is using PHP's alternative syntax for control structures, and could also be expressed as:

```
<?php
if ( have_posts() ) {
    while ( have_posts() ) {
        the_post();
        //
        // Post Content here
        //
    } // end while
} // end if
?>
```

Template Tags

1. General tags

`wp_head()`

Description

Fire the 'wp_head' action. Put this template tag immediately before `</head>` tag in a theme template (ex. `header.php`, `index.php`).

Usage

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

Parameters

This function does not accept any parameters.

Return values

None.

Examples

wp-content/themes/twentyten/header.php:

```
<?php
...
/* Always have wp_head() just before the closing </head>
 * tag of your theme, or you will break many plugins, which
 * generally use this hook to add elements to <head> such
 * as styles, scripts, and meta tags.
 */

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

get_footer()

Description

Includes the footer.php template file from your current theme's directory. if a name is specified then a specialised footer footer-{name}.php will be included.

If the theme contains no footer.php file then the footer from the default theme wp-includes/theme-compat/footer.php will be included.

Usage

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

Parameters

\$name

(string) (optional) Calls for footer-name.php.
Default: None

Return Values

None.

Examples

Simple 404 page

The following code is a simple example of a template for an "HTTP 404: Not Found" error (which you could include in your Theme as 404.php).

```
<?php get_header(); ?>
<h2>Error 404 - Not Found</h2>
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Multiple Footers

Different footer for different pages.

```
<?php
if ( is_home() ) :
    get_footer( 'home' );
elseif ( is_404() ) :
    get_footer( '404' );
else :
    get_footer();
endif;
?>
```

get_header()

Description

Includes the header.php template file from your current theme's directory. If a name is specified then a specialised header header-{name}.php will be included.

If the theme contains no header.php file then the header from the default theme wp-includes/theme-compat/header.php will be included.

Usage

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

Parameters

\$name

(string) (optional) Calls for header-name.php.

Default: None

Examples

Simple 404 page

The following code is a simple example of a template for an "HTTP 404: Not Found" error (which you could include in your Theme as 404.php).

```
<?php get_header(); ?>
<h2>Error 404 - Not Found</h2>
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Multiple Headers

Different header for different pages.

```
<?php
if ( is_home() ) :
    get_header( 'home' );
elseif ( is_404() ) :
    get_header( '404' );
else :
    get_header();
endif;
?>
```

get_sidebar()

Description

This function calls each of the active widget callbacks in order, which prints the markup for the sidebar. If you have more than one sidebar, you should give this function the name or number of the sidebar you want to print. This function returns true on success and false on failure.

The return value should be used to determine whether to display a static sidebar. This ensures that your theme will look good even when the Widgets plug-in is not active.

If your sidebars were registered by number, they should be retrieved by number. If they had names when you registered them, use their names to retrieve them.

Usage

```
<?php dynamic_sidebar( $index ); ?>
```

Parameters

index

(integer/string) (optional) Name or ID of dynamic sidebar.

Default: 1

Return Value

(boolean)

True, if widget sidebar was found and called. False if not found or not called.

Examples

Here is the recommended use of this function:

```
<?php if ( is_active_sidebar( 'left-sidebar' ) ) : ?>
    <ul id="sidebar">
        <?php dynamic_sidebar( 'left-sidebar' ); ?>
    </ul>
<?php endif; ?>
<ul id="sidebar">
    <?php dynamic_sidebar( 'right-sidebar' ); ?>
</ul>
<ul id="sidebar">
<?php if ( ! dynamic_sidebar() ) : ?>
    <li>{static sidebar item 1}</li>
    <li>{static sidebar item 2}</li>
<?php endif; ?>
</ul>
```

get_search_form()

Description

Will first attempt to locate the searchform.php file in either the child or the parent, then load it. If it doesn't exist, then the default search form will be displayed. The default search form is HTML, which will be displayed. There is a filter applied to the search form HTML in order to edit or replace it. The filter is 'get_search_form'.

This function is primarily used by themes which want to hardcode the search form into the sidebar and also by the search widget in WordPress.

There is also an action that is called whenever the function is run called, 'pre_get_search_form'. This can be useful for outputting JavaScript that the search relies on or various formatting that applies to the beginning of the search. To give a few examples of what it can be used for.

Parameters

\$echo

(bool) (Optional) Default to echo and not return the form.

Default value: true

Return

(string|void) String when \$echo is false.

bloginfo()

Description

Retrieves information about the current blog. The function is located in: /wp-includes/general-template.php

Parameters

\$show

(string) (optional) What information to get.
Default: name

\$filter

(string) (optional) Get raw or display information.
Default: raw

\$show

- url
- wpurl
- description
- rdf_url
- rss_url
- rss2_url
- atom_url
- comments_atom_url
- comments_rss2_url
- pingback_url
- stylesheet_url
- stylesheet_directory
- template_directory
- template_url
- admin_email
- charset
- html_type
- version
- language
- text_direction
- name

Return Values

(string)

Returns the requested information.

Usage

```
<?php get_bloginfo($show, $filter); ?>
```

wp_title()

Description

By default, the page title will display the separator before the page title, so that the blog title will be before the page title. This is not good for title display, since the blog title shows up on most tabs and not what is important, which is the page that the user is looking at.

There are also SEO benefits to having the blog title after or to the 'right' of the page title. However, it is mostly common sense to have the blog title to the right with most browsers supporting tabs. You can achieve this by using the seplocation parameter and setting the value to 'right'. This change was introduced around 2.5.0, in case backward compatibility of themes is important.

Parameters

\$sep

(string) (Optional) default is '»'. How to separate the various items within the page title.

Default value: '»'

\$display

(bool) (Optional) Whether to display or retrieve title.

Default value: true

\$seplocation

(string) (Optional) Direction to display title, 'right'.

Default value: ''

Return

(string|null) String on retrieve, null when displaying.

single_post_title()

Description

Displays or returns the title of the post when on a single post page (permalink page). This tag can be useful for displaying post titles outside The Loop.

Usage

```
<?php single_post_title( $prefix, $display ); ?>
```

Default Usage

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

Parameters

\$prefix

(string) (optional) Text to place before the title.

Default: None

\$display

(boolean) (optional) Should the title be displayed (TRUE) or returned for use in PHP (FALSE).

Default: TRUE

Example

```
<h2><?php single_post_title( 'Current post: ' ); ?></h2>
```

wp_footer()

Description

Fire the 'wp_footer' action. Put this template tag immediately before `</body>` tag in a theme template (ex. footer.php, index.php).

Usage

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

Parameters

This function does not accept any parameters.

Examples

In twentyten theme

wp-content/themes/twentyten/footer.php:

```
...
<?php
/* Always have wp_footer() just before the closing </body>
 * tag of your theme, or you will break many plugins, which
 * generally use this hook to reference JavaScript files.
 */
wp_footer();
?>
</body>
</html>
```

comments_template()

Description

The `comments_template` filter hook filters the path to the theme template file used for the comments template. It is part of the `comments_template()` function.

The `comments_template` filter can be used to load a custom template from a plugin which replaces the theme's default comment template.

Parameters

`$theme_template`

(string) (required) The path to the theme template file.

Default: None

Examples

A plugin can register as a content filter with the code:

```
<?php add_filter( "comments_template", "my_plugin_comment_template" ); ?>
```

Where `my_plugin_comment_template` is the function WordPress should call when the `comment_template()` function is called on the theme. Note that the filter function the plugin defines must return the a full path to a template file or the resulting page will be blank.

This is an example of loading a different comments template for a custom post type:

```
<?php
function my_plugin_comment_template( $comment_template ) {
    global $post;
    if ( !( is_singular() && ( have_comments() || 'open' == $post->comment_status ) ) ) {
        return;
    }
    if($post->post_type == 'business'){ // assuming there is a post type called business
        return dirname(__FILE__) . '/reviews.php';
    }
}

add_filter( "comments_template", "my_plugin_comment_template" );
?>
```

add_theme_support()

Description

Allows a theme or plugin to get support of a certain theme feature.

Usage

```
<?php get_theme_support( $feature ); ?>
```

Parameters

\$feature

(string) (required) Name for the feature being added.

Default: None

Features list:

- Sidebar Widgets
- Navigation Menus
- Post Formats
- Post Thumbnails
- Custom Backgrounds
- Custom Headers
- Automatic Feed Links
- Editor Style

See Theme Features for more information.

Feature Means

See [Function_Reference/add_theme_support](#) for more.

Return

Returns either:

- `array()` - containing the arguments passed when support for the feature was registered via `add_theme_support($feature, $arguments);` OR
- `true (bool)` - if support for the feature was added without arguments (`add_theme_support($feature);`); OR
- `false (bool)` - if the feature is not supported by the current theme.

get_template_directory_uri()

Description

Retrieves the absolute path to the directory of the current theme.

Note: Does not contain a trailing slash.

Returns an absolute server path (eg: `/home/user/public_html/wp-content/themes/my_theme`), not a URI.

In the case a child theme is being used, the absolute path to the parent theme directory will be returned. Use `get_stylesheet_directory()` to get the absolute path to the child theme directory.

To retrieve the URI of the stylesheet directory use `get_stylesheet_directory_uri()` instead.

Usage

```
<?php echo get_template_directory(); ?>
```

Parameters

This tag has no parameters.

Return Values

(string)

Absolute path to the directory of the current theme (without the trailing slash).

Examples

Include a PHP file

```
<?php include( get_template_directory() . '/includes/myfile.php'); ?>
```

body_class()

Description

The "body_class" filter is used to filter the classes that are assigned to the body HTML element on the current page.

A plugin (or theme) can filter these classes with the code:

```
<?php add_filter( 'body_class', 'filter_function_name' ) ?>
```

Where 'filter_function_name' is the function WordPress should call when the classes are being assigned. Note that the filter function must return the array of classes after it is finished processing, or all of the classes will be cleared and could seriously impact the visual state of a user's site.

filter_function_name should be unique function name. It cannot match any other function name already declared.

This filter is used by the `get_body_class()` function.

Examples

Classes in WordPress Multisite

This could be used to provide custom classes for applying different styles to specific sites in Multisite that all use the same theme:

```
// Apply filter  
add_filter('body_class', 'multisite_body_classes');
```

```
function multisite_body_classes($classes) {  
    $id = get_current_blog_id();  
    $slug = strtolower(str_replace(' ', '-', trim(get_bloginfo('name'))));  
    $classes[] = $slug;  
    $classes[] = 'site-id-'. $id;  
    return $classes;  
}
```

2. Author tags

the_author()

Description

The author of a post can be displayed by using this Template Tag. This tag must be used within The Loop.

To return to PHP rather than displaying, use `get_the_author()`.

Usage

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

Examples

Display Author's 'Public' Name

Displays the value in the user's Display name publicly as field.

```
<p>This post was written by <?php the_author(); ?></p>
```

get_the_author()

Description

Retrieve the post author's display name. This tag must be used within The Loop.

To get the post author's ID, use `get_the_author_meta('ID')`.

To display a page for authors which have no posts, see this discussion.

Since WordPress 2.1 parameters are deprecated (not the function).

Usage

```
<?php $author = get_the_author(); ?>
```

Parameters

\$deprecated

(string) (optional) Deprecated.
Default: ''

Returns

(string)

The author's display name.

Examples

Grab the Author's 'Public' Name

Grabs the value in the user's Display name publicly as field.

```
<?php $author = get_the_author(); ?>
```

the_author_link()

Description

This tag displays a link to the Website for the author of a post. The Website field is set in the user's profile (Administration > Profile > Your Profile). The text for the link is the author's Profile Display name publicly as field. This tag must be used within The Loop.

Usage

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

Parameters

This function does not accept any parameters.

Example

Displays the author's Website URL as a link and the text for the link is the author's Profile Display name publicly as field. In this example, the author's Display Name is James Smith.

```
<p>Written by:  
<?php the_author_link(); ?></p>
```

get_the_author_link()

Description

This tag returns a link to the Website for the author of a post. The Website field is set in the user's profile (Administration > Users > Your Profile). The text for the link is the author's Profile Display name publicly as field. This tag must be used within The Loop.

`get_the_author_link()` returns the link for use in PHP. To display the link instead, use `the_author_link()`.

Usage

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

Parameters

This tag does not accept any parameters.

Example

The example echos (displays) the author's Website URL as a link and the text for the link is the author's Profile Display name publicly as field. In this example, the author's Display Name is James Smith.

<p>Written by:

```
<?php echo get_the_author_link(); ?></p>
```

the_author_meta()

Description

The the_author_meta Template Tag displays a desired meta data field for a user. Only one field is returned at a time, you need to specify which you want.

If this tag is used within The Loop, the user ID value need not be specified, and the displayed data is that of the current post author. A user ID can be specified if this tag is used outside The Loop.

If the meta field does not exist, nothing is printed.

NOTE: Use get_the_author_meta() if you need to return (and do something with) the field, rather than just display it.

Usage

```
<?php the_author_meta( $field, $userID ); ?>
```

Parameters

\$field

(string) Field name for the data item to be displayed. Valid values:

- user_login
- user_pass
- user_nicename
- user_email
- user_url
- user_registered
- user_activation_key
- user_status
- display_name
- nickname

- first_name
- last_name
- description
- jabber
- aim
- yim
- user_level
- user_firstname
- user_lastname
- user_description
- rich_editing
- comment_shortcuts
- admin_color
- plugins_per_page
- plugins_last_view
- ID

\$userID

(integer) (optional) If the user ID fields is used, then this function display the specific field for this user ID.

Default: false

Examples

Display the Author's AIM screenname

Displays the value in the author's AIM (AOL Instant Messenger screenname) field.

```
<p>This author's AIM address is <?php the_author_meta('aim'); ?></p>
```

Display a User Email Address

Displays the email address for user ID 25.

```
<p>The email address for user id 25 is <?php the_author_meta('user_email',25); ?></p>
```

Advanced Uses

A plugin may add an additional field in the registration or manage users, which adds a new value in the wp_usermeta table (where wp_ is your data base prefix). For this example we will use a Twitter ID. For a meta_key value of "twitter" and meta_value of "WordPress" then

```
<p>This author's Twitter name is <?php the_author_meta('twitter'); ?></p>
```

the_author_posts()

Description

Displays the total number of posts an author has published. Drafts and private posts aren't counted. This tag must be used within The Loop.

Usage

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

Example

Displays the author's name and number of posts.

```
<p><?php the_author(); ?> has blogged <?php the_author_posts(); ?> posts</p>
```

3. Category tags

category_description()

Description

Returns the description of a category defined in the category settings screen for the current category (Posts > Categories).

If used in the archive.php template, place this function within the `is_category()` conditional statement. Otherwise, this function will stop the processing of the page for monthly and other archive pages.

Usage

```
<?php echo category_description( $category_id ); ?>
```

Parameters

`$category_id`

(integer) (optional) The ID of the category to return a description.

Default: Description of current query category.

Example

Default Usage

Displays the description of a category, given its id, by echoing the return value of the tag. If no category given and used on a category page, it returns the description of the current category.

```
<div><?php echo category_description(3); ?></div>
```

single_cat_title()

Description

Retrieve the name of a category from its ID.

Usage

```
<?php get_cat_name( $cat_id ) ?>
```

Parameters

\$cat_id

(integer) (required) Category ID

Default: None

Return Values

(string)

Category name

Examples

```
<?php echo get_cat_name(4);?>
```

returns the name for the category with the id '4'.

the_category()

Description

Displays a link to the category or categories a post belongs to. This tag must be used within The Loop.

Usage

```
<?php the_category( $separator, $parents, $post_id ); ?>
```

Parameters

\$separator

(string) (optional) Text or character to display between each category link. By default, the links are placed in an HTML unordered list. An empty string will result in the default behavior.

Default: empty string

\$parents

(string) (optional) How to display links that reside in child (sub) categories. Options are:

- 'multiple' - Display separate links to parent and child categories, exhibiting "parent/child" relationship.

- 'single' - Display link to child category only, with link text exhibiting "parent/child" relationship.

Default: empty string

Note: Default is a link to the child category, with no relationship exhibited.

\$post_id

(int) (optional) Post ID to retrieve categories. The default value false results in the category list of the current post.

Default: false

Examples

Separated by Space

List categories with a space as the separator. `<?php the_category(' '); ?>`

Separated by Comma

Displays links to categories, each category separated by a comma (if more than one).

```
<?php the_category( ' ' ); ?>
```

4. Link tags

the_permalink()

Description

Displays the URL for the permalink to the post currently being processed in The Loop. This tag must be within The Loop, and is generally used to display the permalink for each post, when the posts are being displayed. Since this template tag is limited to displaying the permalink for the post that is being processed, you cannot use it to display the permalink to an arbitrary post on your weblog. Refer to `get_permalink()` if you want to get the permalink for a post, given its unique post id.

Usage

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

Parameters

Before 4.4.0, this tag has no parameters. Since 4.4.0: Added the ``$post`` parameter.

Examples

Display Post URL as Text

Displays the URL to the post, without creating a link:

This address for this post is: `<?php the_permalink(); ?>`

As Link With Text

You can use whatever text you like as the link text, in this case, "permalink".

```
<a href="<?php the_permalink(); ?>">permalink</a>
```

Used as Link With Title Tag

Creates a link for the permalink, with the post's title as the link text. This is a common way to put the post's title.

```
<a href="<?php the_permalink(); ?>" title="<?php the_title_attribute(); ?>"><?php the_title(); ?></a>
```

get_permalink()

Description

Displays the URL for the permalink to the post currently being processed in The Loop. This tag must be within The Loop, and is generally used to display the permalink for each post, when the posts are being displayed. Since this template tag is limited to displaying the permalink for the post that is being processed, you cannot use it to display the permalink to an arbitrary post on your weblog. Refer to `get_permalink()` if you want to get the permalink for a post, given its unique post id.

Usage

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

Parameters

Before 4.4.0, this tag has no parameters. Since 4.4.0: Added the `'$post'` parameter.

Examples

Display Post URL as Text

Displays the URL to the post, without creating a link:

This address for this post is: `<?php the_permalink(); ?>`

As Link With Text

You can use whatever text you like as the link text, in this case, "permalink".

```
<a href="<?php the_permalink(); ?>">permalink</a>
```

Used as Link With Title Tag

Creates a link for the permalink, with the post's title as the link text. This is a common way to put the post's title.

```
<a href="<?php the_permalink(); ?>" title="<?php the_title_attribute(); ?>"><?php the_title(); ?></a>
```

home_url()

Description

The `home_url` template tag retrieves the home URL for the current site, optionally with the `$path` argument appended. The function determines the appropriate protocol, "https" if `is_ssl()` and "http" otherwise. If the `$scheme` argument is "http" or "https" the `is_ssl()` check is overridden.

In case of WordPress Network Setup, use `network_home_url()` instead.

Usage

```
<?php home_url( $path, $scheme ); ?>
```

Default Usage

```
<?php echo esc_url( home_url( '/' ) ); ?>
```

Parameters

\$path

(string) (optional) Path relative to the home URL.

Default: None

\$scheme

(string) (optional) Scheme to use for the home URL. Currently, only "http", "https" and "relative" are supported.

Default: null

Return

(string)

Home URL with the optional \$path argument appended.

Example

```
$url = home_url();  
echo esc_url( $url );
```

Output: <http://www.example.com>

get_home_url()

Description

The `home_url` template tag retrieves the home URL for the current site, optionally with the `$path` argument appended. The function determines the appropriate protocol, "https" if `is_ssl()` and "http" otherwise. If the `$scheme` argument is "http" or "https" the `is_ssl()` check is overridden.

In case of WordPress Network Setup, use `network_home_url()` instead.

Usage

```
<?php home_url( $path, $scheme ); ?>
```

Default Usage

```
<?php echo esc_url( home_url( '/' ) ); ?>
```

Parameters

\$path

(string) (optional) Path relative to the home URL.

Default: None

\$scheme

(string) (optional) Scheme to use for the home URL. Currently, only "http", "https" and "relative" are supported.

Default: null

Return

(string)

Home URL with the optional \$path argument appended.

Example

```
$url = home_url();  
echo esc_url( $url );
```

Output: <http://www.example.com>

site_url()

Description

The `site_url` template tag retrieves the site url for the current site (where the WordPress core files reside) with the appropriate protocol, 'https' if `is_ssl()` and 'http' otherwise. If scheme is 'http' or 'https', `is_ssl()` is overridden. Use this to get the "WordPress address" as defined in general settings. Use `home_url()` to get the "site address" as defined in general settings.

In case of WordPress Network setup, use `network_site_url()` instead.

Usage

```
<?php site_url( $path, $scheme ); ?>
```

Default Usage

```
<?php echo site_url(); ?>
```

Parameters

\$path

(string) (optional) Path to be appended to the site url.

Default: None

\$scheme

(string) (optional) Context for the protocol for the url returned. Setting \$scheme will override the default context. Allowed values are 'http', 'https', 'login', 'login_post', 'admin', or 'relative'.

Default: null

Return

(string)

Site url link with optional path appended.

Examples

```
$url = site_url();  
echo $url;
```

get_site_url()

Description

The `site_url` template tag retrieves the site url for the current site (where the WordPress core files reside) with the appropriate protocol, 'https' if `is_ssl()` and 'http' otherwise. If scheme is 'http' or 'https', `is_ssl()` is overridden. Use this to get the "WordPress address" as defined in general settings. Use `home_url()` to get the "site address" as defined in general settings.

In case of WordPress Network setup, use `network_site_url()` instead.

Usage

```
<?php site_url( $path, $scheme ); ?>
```

Default Usage

```
<?php echo site_url(); ?>
```

Parameters

\$path

(string) (optional) Path to be appended to the site url.

Default: None

\$scheme

(string) (optional) Context for the protocol for the url returned. Setting `$scheme` will override the default context. Allowed values are 'http', 'https', 'login', 'login_post', 'admin', or 'relative'.

Default: null

Return

(string)

Site url link with optional path appended.

Examples

```
$url = site_url();  
echo $url;
```

5. Post tags

`the_content()`

Description

The "the_content" filter is used to filter the content of the post after it is retrieved from the database and before it is printed to the screen.

A plugin (or theme) can register as a content filter with the code:

```
<?php add_filter( 'the_content', 'filter_function_name' ) ?>
```

Where 'filter_function_name' is the function WordPress should call when the content is being retrieved. Note that the filter function must return the content after it is finished processing, or site visitors will see a blank page and other plugins also filtering the content may generate errors.

filter_function_name should be unique function name. It cannot match any other function name already declared.

Examples

Debug Page

This could be used to provide generated content for a page (as an alternative to the Shortcode_API), or for a set of pages sharing some characteristics (e.g. the same author):

```
// returns the content of $GLOBALS['post']
// if the page is called 'debug'
function my_the_content_filter($content) {
    // assuming you have created a page/post entitled 'debug'
    if ($GLOBALS['post']->post_name == 'debug') {
        return var_export($GLOBALS['post'], TRUE );
    }
    // otherwise returns the database content
    return $content;
}
```

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

the_ID()

Description

Displays the numeric ID of the current post. This tag must be within The Loop.

Note: This function displays the ID of the post, to return the ID use get_the_ID().

Usage

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

Parameters

This tag has no parameters.

Examples

Default Usage

```
<p>Post Number: <?php the_ID(); ?></p>
```

Post Anchor Identifier

Provides a unique anchor identifier to each post:

```
<h3 id="post-<?php the_ID(); ?>"><?php the_title(); ?></h3>
```

the_tags()

Description

This template tag displays a link to the tag or tags a post belongs to. If no tags are associated with the current entry, nothing is displayed. This tag should be used within The Loop.

Usage

```
<?php the_tags( $before, $sep, $after ); ?>
```

Parameters

\$before

(string) Text to display before the actual tags are displayed. Defaults to Tags:

\$sep

(string) Text or character to display between each tag link. The default is a comma (,) between each tag.

\$after

(string) Text to display after the last tag. The default is to display nothing.

Return Values

None.

Examples

Default Usage

The default usage lists tags with each tag (if more than one) separated by a comma (,) and preceded with the default text Tags: .

```
<p><?php the_tags(); ?></p>
```

Separated by Commas

Displays a list of the tags with a line break after it.

```
<?php the_tags( 'Tags: ', ', ', '<br />' ); ?>
```


the_title()

Description

Displays or returns the title of the current post. This tag may only be used within The Loop, to get the title of a post outside of the loop use `get_the_title`. If the post is protected or private, this will be noted by the words "Protected: " or "Private: " prepended to the title.

Usage

```
<?php the_title( $before, $after, $echo ); ?>
```

Parameters

\$before

(string) (optional) Text to place before the title.
Default: None

\$after

(string) (optional) Text to place after the title.
Default: None

\$echo

(Boolean) (optional) Display the title (TRUE) or return it for use in PHP (FALSE).
Default: TRUE

Example

```
<?php the_title( '<h3>', '</h3>' ); ?>
```

get_the_title()

Description

Displays or returns the title of the current post. This tag may only be used within The Loop, to get the title of a post outside of the loop use `get_the_title`. If the post is protected or private, this will be noted by the words "Protected: " or "Private: " prepended to the title.

Usage

```
<?php the_title( $before, $after, $echo ); ?>
```

Parameters

\$before

(string) (optional) Text to place before the title.
Default: None

\$after

(string) (optional) Text to place after the title.
Default: None

\$echo

(Boolean) (optional) Display the title (TRUE) or return it for use in PHP (FALSE).
Default: TRUE

Example

```
<?php the_title( '<h3>', '</h3>' ); ?>
```

the_date()

Description

Displays or returns the date of a post, or a set of posts if published on the same day.

Usage

```
<?php the_date( $format, $before, $after, $echo ); ?>
```

Parameters

\$format

(string) (optional) The format for the date. Defaults to the date format configured in your WordPress options. See Formatting Date and Time.

Default: F j, Y

\$before

(string) (optional) Text to place before the date.

Default: None

\$after

(string) (optional) Text to place after the date

Default: None

\$echo

(boolean) (optional) Display the date (TRUE), or return the date to be used in PHP (FALSE).

Default: TRUE

Return

(string|null) Null if displaying, string if retrieving.

Examples

Default Usage

Displays the date using defaults.

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

Date as Year, Month, Date in Heading

Displays the date using the '2007-07-23' format (ex: 2004-11-30), inside an <h2> tag.

```
<?php the_date('Y-m-d', '<h2>', '</h2>'); ?>
```

Date in Heading Using \$my_date Variable

Returns the date in the default format inside an <h2> tag and assigns it to the \$my_date variable. The variable's value is then displayed with the PHP echo command.

```
<?php $my_date = the_date('', '<h2>', '</h2>', FALSE); echo $my_date; ?>
```

get_the_date()

Description

The `get_the_date` template tag retrieves the date the current \$post was written. Unlike `the_date()` this tag will always return the date. Modify output with 'get_the_date' filter.

Usage

```
<?php $pfx_date = get_the_date( $format, $post_id ); ?>
```

Parameters

`$format`

(string) (optional) PHP date format.

Default: the `date_format` option ('Date Format' on Settings > General panel)

`$post`

(integer) (optional) The ID of the post you'd like to fetch. By default the current post is fetched.

Default: null

Return

(string) The formatted date string

Filter

- `apply_filters('get_the_date', $the_date, $format)`

Changelog

- 3.0.0 : New template tag.

Examples

Default Usage

```
<span class="entry-date"><?php echo get_the_date(); ?></span>
```

the_time()

Description

Displays the time of the current post. To return the time of a post, use `get_the_time()`. This tag must be used within The Loop.

Usage

```
<?php the_time( $d ); ?>
```

Parameters

\$d

(string) (optional) The format the time is to display in. Defaults to the time format configured in your WordPress options. See [Formatting Date and Time](#).

Default: None

Examples

Default Usage

Displays the time using your WordPress defaults.

```
<p>Time posted: <?php the_time(); ?></p>
```

Time as AM/PM VS. 24H format

Displays the time using the format parameter string 'g:i a' (ex: 10:36 pm).

```
<p>Time posted: <?php the_time('g:i a'); ?></p>
```

Displays the time using the 24 hours format parameter string 'G:i' (ex: 17:52).

```
<p>Time posted: <?php the_time('G:i'); ?></p>
```

Date as Month Day, Year

Displays the time in the date format 'F j, Y' (ex: December 2, 2004), which could be used to replace the tag `the_date()`.

```
<div><?php the_time('F j, Y'); ?></div>
```

Date and Time

Displays the date and time.

```
<p>Posted: <?php the_date('F j, Y'); ?> at <?php the_time('g:i a'); ?></p>
```

next_post_link()

Description

Used on single post permalink pages, this template tag displays a link to the next post which exists in chronological order from the current post.

In standard usage (within the default, unaltered loop) `next_post_link` will generate a link to a post that is newer (more recent) than the current post. This is in contrary to the similarly-named `previous_posts_link`, which will typically link to a page of posts that is older than the current batch.

This tag must be used in The Loop.

Usage

```
<?php next_post_link( $format, $link, $in_same_term = false, $excluded_terms = '', $taxonomy = 'category' ); ?>
```

Parameters

format

(string) (Optional) Format string for the link. This is where to control what comes before and after the link. '%link' in string will be replaced with whatever is declared as 'link' (see next parameter). 'Go to %link' will generate "Go to <a href=..." Put HTML tags here to style the final results.
Default: '%link »'

link

(string) (Optional) Link text to display.
Default: '%title' (next post's title)

in_same_term

(boolean) (optional) Indicates whether next post must be within the same taxonomy term as the current post. If set to 'true', only posts from the current taxonomy term will be displayed. If the post is in both the parent and subcategory, or more than one term, the next post link will lead to the next post in any of those terms.

- true
- false

Default: false

excluded_terms

(string/array) (optional) Array or a comma-separated list of numeric terms IDs from which the next post should not be listed. For example array(1, 5) or '1,5'. This argument used to accept a list of IDs separated by 'and', this was deprecated in WordPress 3.3

Default: None

taxonomy

(string) (Optional) Taxonomy, if \$in_same_term is true. Added in WordPress 3.8.
Default: 'category'

Examples

Default Usage

Displays link with the post title of the next post (chronological post date order), followed by a right angular quote (»).

Next Post Title »

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

previous_post_link()

Description

Used on single post permalink pages, this template tag displays a link to the previous post which exists in chronological order from the current post.

This tag must be used in The Loop.

Arguments

```
<?php previous_post_link( $format, $link, $in_same_term = false, $excluded_terms = '', $taxonomy = 'category' ); ?>
```

Parameters

format

(string) (Optional) Format string for the link. This is where to control what comes before and after the link. '%link' in string will be replaced with whatever is declared as 'link' (see next parameter). 'Go to %link' will generate "Go to <a href=..." Put HTML tags here to style the final results.
Default: '« %link'

link

(string) (Optional) Link text to display.
Default: '%title' (previous post's title)

in_same_term

(boolean) (optional) Indicates whether previous post must be within the same taxonomy term as the current post. If set to 'true', only posts from the current taxonomy term will be displayed. If the post is in both the parent and subcategory, or more than one term, the previous post link will lead to the previous post in any of those terms.

- true
- false

Default: false

excluded_terms

(string/array) (optional) Array or a comma-separated list of numeric terms IDs from which the next post should not be listed. For example array(1, 5) or '1,5'. This argument used to accept a list of IDs separated by 'and', this was deprecated in WordPress 3.3

Default: None

taxonomy

(string) (Optional) Taxonomy, if \$in_same_term is true. Added in WordPress 3.8.
Default: 'category'

Examples

Default Usage

Displays link with left angular quote («) followed by the post title of the previous post (chronological post date order).

« Previous Post Title

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

posts_nav_link()

Description

Displays links for next and previous pages. Useful for providing "paged" navigation of index, category and archive pages.

For displaying next and previous pages of posts see `next_posts_link()` and `previous_posts_link()`.

For displaying next and previous post navigation on individual posts, see `next_post_link()` and `previous_post_link()`.

Usage

```
<?php posts_nav_link( $sep, $prelabel, $nextlabel ); ?>
```

Note: since weblog posts are traditionally listed in reverse chronological order (with most recent posts at the top), there is some ambiguity in the definition of "next page". WordPress defines "next page" as the "next page toward the past". In WordPress 1.5, the default Kubrick theme addresses this ambiguity by labeling the "next page" link as "previous entries". See Example: Kubrick Theme Format.

Parameters

\$sep

(string) Text displayed between the links.

- Defaults to ' :: ' in 1.2.x.
- Defaults to ' — ' in 1.5.

\$prelabel

(string) Link text for the previous page.

- Defaults to '<< Previous Page' in 1.2.x.
- Defaults to '« Previous Page' in 1.5.

\$nextlabel

(string) Link text for the next page.

- Defaults to 'Next Page >>' in 1.2.x.
- Defaults to 'Next Page »' in 1.5

Examples

Default Usage

By default, the `posts_nav_link()` look like this:

« Previous Page — Next Page »

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


post_class()

Description

WordPress theme authors who want to have finer css control options for their post styling, have the `post_class` function available. When the `post_class` function is added to a tag within the loop, for example `<div <?php post_class(); ?> >`, it will print out and add various post-related classes to the div tag. It can also be used outside the loop with the optional `post_id` parameter. This function is typically used in the `index.php`, `single.php`, and other template files that feature hierarchical post listings.

If you would prefer to have the post classes returned instead of echoed, you would want to use `get_post_class()`. Note: `get_post_class()` does not return a string, but an array that must be processed to produce text similar to what is echoed by `post_class()`.

For css classes intended to help target entire pages, see `body_class()`, and for classes targeting comment listings, see `comment_class()`.

Usage

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```

The `post_class` may include one or more of the following values for the class attribute, dependent upon the pageview.

- `.post-[id]`
- `.[post-type]`
- `.type-[post-type]`
- `.status-[post-status]`
- `.format-[post-format]` (default to 'standard')
- `.post-password-required`
- `.post-password-protected`
- `.has-post-thumbnail`
- `.sticky`
- `.hentry` (hAtom microformat pages)
- `.[taxonomy]-[taxonomy-slug]` (includes category)
- `.tag-[tag-name]`

Default Values

The `post_class` CSS classes appear based upon the post pageview Conditional Tags as follows.

Front Page

If posts are found on the front page of the blog, be it static or not, the class selectors are: `post post-id hentry`

Single Post

Single post template files and pageviews feature the class selectors: `post post-id hentry`

Sticky Post

Posts designated as "sticky," sticking to the front page of the blog, feature the class selector: `sticky`

Author

Author template files and pageviews displaying posts feature the class selectors: post post-id

Category

Category template files and pageviews displaying posts feature the class selectors: post post-id category-ID category-name

Tags

Tag template files and pageviews with posts feature the class selectors: tag-name post post-id

Archives

Archive pageviews and pageviews with posts feature CSS classes: post post-id

Search

Search template files and pageviews with posts feature the class selectors: post post-id

Attachment

Attachment template files and pageviews feature the class selectors: attachment post post-id

Format

Posts using Post Formats feature the class selector: format-name

Parameters

How to pass parameters to tags with PHP function-style parameters

class

(string or array) (optional) One or more classes to add to the class attribute, separated by a single space.

Default: null

\$post_id

(int) (optional) An optional post ID, used when calling this function from outside The Loop

Default: null

Examples

Implementation

The following example shows how to implement the `post_class` template tag into a theme.

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```

The actual HTML output might resemble something like this for a post in the "dancing" category:

```
<div id="post-4564" class="post post-4564 category-48 category-dancing logged-in">
```

In the WordPress Theme stylesheet, add the appropriate styles, such as:

```
.post {  
    /* styles for all posts */  
}  
.post-4564 {  
    /* styles for only post ID number 4564 */  
}  
.category-dancing {  
    /* styles for all posts within the category of dancing */  
}
```

UNIT 5

Advanced development

Advanced functions

add_action()

Description

Hooks a function to a specific action (removes the function execution from a special location or prevents it from being executed).

Action hook list: Plugin API / Action Reference Action is usually executed by `do_action()`.

Usage

```
<?php add_action ( $tag , $function_to_add , $priority , $accepted_args ); ?>
```

Parameters

\$ Tag

(String) (required) Name of hook to connect (Action hook list: Plugin API / Action Reference).

Alternatively, you can include actions included in a theme or plugin.

Default: None

\$ Function_to_add

(Callback) (required) The name of the function to call. Note: Only string-formatted syntaxes are listed in the PHP documentation for the 'callback' type .

Default: None

\$ Priority

(Int) (optional) The importance of the function to be called. Change this value to be called before or after another function. The default value is 10, for example, set to 5, and set to 12 to run later.

Default: 10

\$ Accepted_args

(Int) (optional) How many arguments your function takes. In WordPress 1.5.1+, hooked functions can take extra arguments that do not match the `do_action()` or `apply_filters()` call is run. For example, the action `comment_id_not_found` will pass any functions that hook onto it.

Default: 1

Examples

Simple Hook

An example of sending an email to a friend every time they post a blog:

Function `email_friends($ post_ID)`

```
{
    $ Friends = 'bob@example.org, susie@example.org';
    Wp_mail ( $ friends, "sally's blog updated", "I just put something on my blog: http://blog.example.com");

    Return $ post_ID;
}
Add_action ( 'publish_post', 'email_friends');
```

Take Arguments

The hooked function takes one argument from the action. Specifically, the 'echo_comment_id' function takes the argument `$comment_ID` . It then echos the value of the received argument.

Function `echo_comment_id($ comment_ID)`

```
{
    Echo "I just received $ comment_ID";
}
```

```
}  
Add_action ( 'comment_id_not_found', 'echo_comment_id', 10, 1);
```

add_filter()

Description

The "the_content" filter is used to filter the content of the post after it is retrieved from the database and before it is printed to the screen.

A plugin (or theme) can register as a content filter with the code:

```
<?php add_filter( 'the_content', 'filter_function_name' ) ?>
```

Where 'filter_function_name' is the function WordPress should call when the content is being retrieved.

Note that the filter function must return the content after it is finished processing, or site visitors will see a blank page and other plugins also filtering the content may generate errors.

filter_function_name should be unique function name. It cannot match any other function name already declared.

Examples

Debug Page

This could be used to provide generated content for a page (as an alternative to the Shortcode_API), or for a set of pages sharing some characteristics (e.g. the same author):

```
// returns the content of $GLOBALS['post']  
// if the page is called 'debug'  
function my_the_content_filter($content) {  
    // assuming you have created a page/post entitled 'debug'  
    if ($GLOBALS['post']->post_name == 'debug') {  
        return var_export($GLOBALS['post'], TRUE );  
    }  
    // otherwise returns the database content  
    return $content;  
}
```

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

Post Icon

This filter function adds an image before the post on the post page (see is_single()). It assumes an image named post_icon.png exists in the theme images folder. It runs at a lower priority (20) which runs later than most other filters (default is 10).

```
add_filter( 'the_content', 'my_the_content_filter', 20 );
```

```
/**
```

```
 * Add a icon to the beginning of every post page.
```

```
 *
```

```
 * @uses is_single()
```

```
 */
```

```
function my_the_content_filter( $content ) {
```

```
    if ( is_single() )
```

```
        // Add image to the beginning of each page
```

```
        $content = sprintf(
```

```
            '%s',
```

```
            get_bloginfo( 'stylesheet_directory' ),
```

```
            $content
```

```
        );
```

```
// Returns the content.
return $content;
}
```

Featured Image

Adds a featured image set from the single post Edit screen which displays before the content on single posts only.

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

function featured_image_before_content( $content ) {
    if ( is_singular('post') && has_post_thumbnail() ) {
        $thumbnail = get_the_post_thumbnail();
        $content = $thumbnail . $content;
    }
    return $content;
}
```

add_shortcode()

Description

Adds a hook for a shortcode tag.

Usage

```
<?php add_shortcode( $tag , $func ); ?>
```

Parameters

\$tag

(string) (required) Shortcode tag to be searched in post content

Default: None

\$func

(callable) (required) Hook to run when shortcode is found

Default: None

Return Values

(none)

Examples

Simplest example of a shortcode tag using the API: [footag foo="bar"]

```
function footag_func( $atts ) {
    return "foo = {$atts['foo']}";
}

```

```
add_shortcode( 'footag', 'footag_func' );
```

Example with nice attribute defaults: [bartag foo="bar"]

```
function bartag_func( $atts ) {
    $atts = shortcode_atts( array(
        'foo' => 'no foo',
        'baz' => 'default baz'
    ), $atts, 'bartag' );

    return "foo = {$atts['foo']}";
}

```

```
add_shortcode( 'bartag', 'bartag_func' );
```

do_shortcode()

Description

Search the shortcode content and filter them through their hooks.

use

```
<?php echo do_shortcode ( $content ) ?>
```

Parameters

\$ content

(String) (required) Content to search shortcodes

Default: None

Values returned

(String)

Content with shortcode replaced by output from the functions that manage them.

Examples

```
Add_filter ( 'the_content', 'do_shortcode', 11 ); // from shortcodes.php
```

```
// Use a shortcode in a PHP file (outside the content editor).
```

```
Echo do_shortcode ( '[gallery]' );
```

```
// In case there was an opening shortcode and one of the closure.
```

```
Echo do_shortcode ( '[incorrect]'. $ Testo_da_include_nello_shortcode. '[/ Iscorrect]' );
```

```
// Use shortcuts in text widgets.
```

```
Add_filter ( 'widget_text', 'do_shortcode' );
```

```
// Example of a shortcode that creates a module
```

```
Echo do_shortcode ( '[contact-form-7 id = "91" title = "quote"]' );
```

register_nav_menu()

Description

Registers a single custom Navigation Menu in the custom menu editor (in WordPress 3.0 and above). This allows administration users to create custom menus for use in a theme.

See register_nav_menus() for creating multiple menus at once.

Usage

```
<?php register_nav_menu( $location, $description ); ?>
```

Parameters

\$location

(string) (required) Menu location identifier, like a slug.

Default: None

\$description

(string) (required) Menu description - for identifying the menu in the dashboard.

Default: None

Return Values

None.

Examples

```
<?php
```

```
add_action( 'after_setup_theme', 'register_my_menu' );
```

```
function register_my_menu() {
```

```
    register_nav_menu( 'primary', __( 'Primary Menu', 'theme-slug' ) );
```

```
}
```

```
?>
```

Custom Post Types

register_post_type()

Description

Create or modify a post type. `register_post_type` should only be invoked through the 'init' action. It will not work if called before 'init', and aspects of the newly created or modified post type will work incorrectly if called later.

Note: You can use this function in themes and plugins. However, if you use it in a theme, your post type will disappear from the admin if a user switches away from your theme.

Taxonomies

When registering a post type, always register your taxonomies using the `taxonomies` argument. If you do not, the taxonomies and post type will not be recognized as connected when using filters such as `parse_query` or `pre_get_posts`. This can lead to unexpected results and failures.

Even if you register a taxonomy while creating the post type, you must still explicitly register and define the taxonomy using `register_taxonomy()`.

Reserved Post Types

The following post types are reserved and used by WordPress already.

- `post`
- `page`
- `attachment`
- `revision`
- `nav_menu_item`
- `custom_css`
- `customize_changeset`

In addition, the following post types should not be used as they interfere with other WordPress functions.

- `action`
- `author`
- `order`
- `theme`

In general, you should always prefix your post types, or specify a custom `'query_var'`, to avoid conflicting with existing WordPress query variables.

More information: [Post Types](#).

Usage

```
<?php register_post_type( $post_type, $args ); ?>
```

Parameters

`$post_type`

(string) (required) Post type. (max. 20 characters, cannot contain capital letters or spaces)

Default: None

`$args`

(array) (optional) An array of arguments.

Default: None

register_taxonomy()

Description

This function adds or overwrites a taxonomy. It takes in a name, an object name that it affects, and an array of parameters. It does not return anything.

Care should be used in selecting a taxonomy name so that it does not conflict with other taxonomies, post

types, and reserved WordPress public and private query variables. A complete list of those is described in the Reserved Terms section. In particular, capital letters should be avoided (This was allowed in 3.0, but not enforced until 3.1 with the "Cheatin'" error).

Usage

```
<?php register_taxonomy( $taxonomy, $object_type, $args ); ?>
```

Use the init action to call this function. Calling it outside of an action can lead to troubles. See #15568 for details.

Better be safe than sorry when registering custom taxonomies for custom post types. Use `register_taxonomy_for_object_type()` right after the function to interconnect them. Else you could run into minetraps where the post type isn't attached inside filter callback that run during `parse_request` or `pre_get_posts`.

Parameters

`$taxonomy`

(string) (required) The name of the taxonomy. Name should only contain lowercase letters and the underscore character, and not be more than 32 characters long (database structure restriction).

Default: None

`$object_type`

(array/string) (required) Name of the object type for the taxonomy object. Object-types can be built-in Post Type or any Custom Post Type that may be registered.

Default: None

Built-in Post Types:

- `post`
- `page`
- `attachment`
- `revision`
- `nav_menu_item`
- `custom_css`
- `customize_changeset`

Custom Post Types:

- `{custom_post_type}` - Custom Post Type names must be all in lower-case and without any spaces.
- `null` - Setting explicitly to null registers the taxonomy but doesn't associate it with any objects, so it won't be directly available within the Admin UI. You will need to manually register it using the 'taxonomy' parameter (passed through `$args`) when registering a custom post_type (see `register_post_type()`), or using `register_taxonomy_for_object_type()`.

`$args`

(array/string) (optional) An array of Arguments.

Default: None

Widget Area

`register_sidebar()`

Description

Builds the definition for a single sidebar and returns the ID. Call on "widgets_init" action.

Usage

```
<?php register_sidebar( $args ); ?>
```

Default Usage

```
<?php $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>'; ?>

```

Parameters

args

(string/array) (optional) Builds Sidebar based off of 'name' and 'id' values.

Default: None

- name - Sidebar name (default is localized 'Sidebar' and numeric ID).
- id - Sidebar id - Must be all in lowercase, with no spaces (default is a numeric auto-incremented ID). If you do not set the id argument value, you will get E_USER_NOTICE messages in debug mode, starting with version 4.2.
- description - Text description of what/where the sidebar is. Shown on widget management screen. (Since 2.9) (default: empty)
- class - CSS class to assign to the Sidebar in the Appearance -> Widget admin page. This class will only appear in the source of the WordPress Widget admin page. It will not be included in the frontend of your website. Note: The value "sidebar" will be prepended to the class value. For example, a class of "tal" will result in a class value of "sidebar-tal". (default: empty).
- before_widget - HTML to place before every widget (default: <li id="%1\$s" class="widget %2\$s">)
Note: uses sprintf for variable substitution
- after_widget - HTML to place after every widget (default: \n).
- before_title - HTML to place before every title (default: <h2 class="widgettitle">).
- after_title - HTML to place after every title (default: </h2>\n).

dynamic_sidebar()

Description

This function calls each of the active widget callbacks in order, which prints the markup for the sidebar. If you have more than one sidebar, you should give this function the name or number of the sidebar you want to print. This function returns true on success and false on failure.

The return value should be used to determine whether to display a static sidebar. This ensures that your theme will look good even when the Widgets plug-in is not active.

If your sidebars were registered by number, they should be retrieved by number. If they had names when you registered them, use their names to retrieve them.

Usage

```
<?php dynamic_sidebar( $index ); ?>
```

Parameters

index

(integer/string) (optional) Name or ID of dynamic sidebar.

Default: 1

Return Value

(boolean)

True, if widget sidebar was found and called. False if not found or not called.

Examples

Here is the recommended use of this function:

```

<?php if ( is_active_sidebar( 'left-sidebar' ) ) : ?>
    <ul id="sidebar">
        <?php dynamic_sidebar( 'left-sidebar' ); ?>
    </ul>

```

```
<?php endif; ?>
<ul id="sidebar">
    <?php dynamic_sidebar( 'right-sidebar' ); ?>
</ul>
<ul id="sidebar">
<?php if ( ! dynamic_sidebar() ) : ?>
    <li>{static sidebar item 1}</li>
    <li>{static sidebar item 2}</li>
<?php endif; ?>
</ul>
```