Table of Contents

WEB PROGRAMMING& WEB SERVICES

WEB PAGE

A web page or webpage is a document commonly written in Hypertext Markup Language (HTML) that is accessible through the Internet or other network using a browser. A web browser displays a web page on a monitor or mobile device. A web page is accessed by entering a URL addresses and may contain text, graphics, and hyperlinks to other web pages and files. The page you're reading now is an example of a web page. The first web page was created at CERN by Tim Berners-Lee and put online August 6, 1991.

WEB SITE

A website, or web site, is a central location of various web pages that are all related and can be accessed by visiting the home page using a browser. A website refers to a central location that contains more than one web page. For example, www.w3schools.com is considered a website, which contains thousands of different pages. However, this page you're reading now is considered a web page on w3schools.A web site, or individual web page, can be ***static*** or ***dynamic***.

A static website contains information that does not change. It remains the same, or static, for every viewer of the site. A dynamic website contains information that changes, depending on the viewer of the site, the time of the day, the time zone, the native language of the country the viewer is in or many other factors.

A dynamic web site can contain client-side scripting or server-side scripting to generate the changing content, or a combination of both scripting types. These sites also include HTML programming for the basic structure. The client-side or server-side scripting takes care of the guts of the site.

CLIENT-SIDE ENVIRONMENT

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the user's computer over the internet and run directly in the browser. The scripting language needs to be enabled on the client computer. Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.

SERVER-SIDE ENVIRONMENT

The server-side environment that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

ASP.Net - This is from Microsoft and is designed to run on Window's servers. Its advantages include a familiarity among developers who already program in Microsoft's .Net environment, and easily connecting to their existing libraries to allow easy database access, and moving of programs from a desktop to a web environment.

PHP - A very common language which can be run on almost any server. PHP has been around for years and it is easy to find on-line resources for the language. It makes many tasks very easy to perform, as it doesn't require extensive knowledge of object-oriented programming to work with PHP. Almost all web servers include PHP in their hosting packages.

Python - Has become popular in recent years and works equally well as a server side language for the web, as it does for scripting on the desktop to automate tasks.

Ruby - Has been around for a while, but found new popularity recently with the Rails Framework (why you often hear of Ruby on Rails). Makes many complex tasks fairly simple, but does not work well on shared hosts.

JSP - Based on the Java Language, and part of the Enterprise Framework, it allows Java programmers a way to develop for the web. Allows for easy access to databases, encryption, and more, if you are familiar with Java and Object-Oriented programming.

Regardless of language you may find or use, all server side languages allow you to perform some of the same tasks.

Be able to handle input from forms - The server needs to be able to take information which an end user enters, and be able to process it according to what we want to do with the file (see below).

Work with external files - With a server side language, you can import a file to display or run, as well as write out a file to create a new file or update an existing file.

Access a database - While the databases supported may not be the same, server side languages allow you to access a database to quickly and easily store, update, and retrieve data. Most languages support many different databases, and we will look at some of those coming up.

Communicate with other Systems - The simplest version of this, is communicating with an email server to send emails. However, access to other systems may be available as well via messaging, network protocols, and/or SMS messaging.

WEB SERVER

A Web Server is a Computer or Combination of computers, which is connected through internet or intranet to serve the clients quests, coming from their web browser. It is a large repository of web pages which transfer to the client in response to their request. The client request server through protocol such as FTP, HTTP, SMTPfor their own specific use. Every web server has a unique IP address and domain name which identifies that machine on the network. A server contains the server software installed on it, which manages the client request and response them.

There are many types of web server, Enterprise uses according to their need. Some of the popular categories of web servers are -

HTTP Server - It handles HTTP request coming from client's browser and transfer the static pages to client in response to their request. This pages runs of the client browser. It generally contains the static pages.

FTP Server - This type of server used for file transfer from one machine (Computer) to another using the internet or intranet. It uses File Transfer Protocols to transfer file from one computer to another. Such type of server uses some file transfer policies, authentication, login validation etc

Mail Server - A Mail Server store and retrieve mail messages from client mail box.

Application Server - It is installed database and web servers.

APACHE - WEB SERVER

The Apache HTTP server is a software (or program) that runs in the background under an appropriate operating system, which supports multi-tasking, and provides services to other applications that connect to it, such as client web browsers. It was first developed to work with Linux/Unix operating systems, but was later adapted to work under other systems, including Windows and Mac.

IIS – WEB SERER

IIS (Internet Information Server) is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows 2000 Server operating systems. IIS is Microsoft's entry to compete in the Internet server market that is also addressed by Apache, Sun Microsystems, O'Reilly, and others. With IIS, Microsoft includes a set of programs for building and administering Web sites, a search engine, and support for writing Web-based applications that access databases. Microsoft points out that IIS is tightly integrated with the Windows NT and 2000 Servers in a number of ways, resulting in faster Web page serving.

A typical company that buys IIS can create pages for Web sites using Microsoft's Front Page product (with its WYSIWYG user interface). Web developers can use Microsoft's Active Server Page (ASP) technology, which means that applications - including ActiveX controls - can be imbedded in Web pages that modify the content sent back to users.

PROTOCOL

Protocol is a system of digital rules for message exchange within or between computers.Communicating systems use well-defined formats for exchanging messages. Each message has an exact meaning intended to provoke a particular response of the receiver. Thus, a protocol must define the syntax, semantics, and synchronization of communication; the specified behavior is typically independent of how it is to be implemented. A protocol can therefore be implemented as hardware, software, or both.

HTTP: Short for HyperText Transfer Protocol, HTTP is a set of standards that allow users of the World Wide Web to exchange information found on web pages. When wanting to access any web page enter http:// in front of the web address, which tells the browser to communicate over HTTP. For example, the full URL for w3schools is http://www.w3schools.com. Today's modern browsers no longer require HTTP in front of the URL since it is the default method of communication. However, it is still used in browsers because of the need to access other protocols such as FTP through the browser. The term HTTP was coined by Ted Nelson.HTTP commonly utilizes port 80, 8008, or 8080.HTTP/0.9 was the first version of the HTTP and was introduced in 1991.HTTP/1.0 is specified in RFC [i]1945 and introduced in 1996.HTTP/1.1 is specified in RFC 2616 and officially released in January 1997.

HTTPS: Short for Hypertext Transfer Protocol over Secure, HTTPS is a secure method of accessing or sending information across a web page. All data sent over HTTPS is encrypted before it is sent, this prevents anyone from understanding that information if intercepted. Because data is encrypted over HTTPS, it is slower than HTTP, which is why HTTPS is only used when requiring login information or with pages that contain sensitive information such as an online bank web page.HTTPS uses port 443 to transfer its information.HTTPS is first used in HTTP/1.1 and is defined in RFC 2616

FTP: Short for File Transfer Protocol, FTP was first proposed on April 16, 1971 by Abhay Bhushan for and developed for implementation on hosts at MIT and was later defined by RFC 959 published in 1985. FTP is a standard way of sending and receiving files between two computers. A good example of how FTP is used today is by web developers, who will connect to their web server using FTP and send updated versions of their web pages to the server.Although many FTP servers require logins, many FTP servers also allow anonymous ftp login, which only require a username, often an e-mail and no password. It is also important to realize that FTP is insecure. When your username and password are sent to the server they're both sent as plaintext and could be intercepted and read. If your server supports SFTP or FTP with TLS encryption, we suggest one of them instead of plain FTP to help keep your information private.

WEB HOST

A company in charge of hosting a web page and all its related content. A web host should not be confused with an ISP, which is who provides you with an Internet connection, although it may be the same.

Hosting is a service whereby one computer configured as an Internet server offers a part of, or its whole resources, for use in exchange for a certain rental fee. Thanks to this service, one or more users can use information, services or content located on this server using another computer called client. The client uses the Internet to connect with the server and displays the desired content to the user.

WEB HOSTING

Web hosting is the most widespread hosting service. It allows your website to be assessable on the Internet 24 hours a day, 365 days a year. The website itself is being hosted on a web server most often located in a specialized data center. The web server offers uninterrupted Internet connectivity, a certain set of software packages, which offer additional services such as e-mail, ftp, databases, as

well as an environment for utilizing different programming languages such as: perl, php, java, xhtml, html and others.

VHOST

Alternatively referred to as an Internet Presence Provider (IPP), Vhost is short for virtual host and is a remote host computer that is run and maintained by another company. By having another company or ISP run and maintain the computers that host the files, this enables an individual or company not to have to worry about the maintenance, setup, upgrade, or security of a computer or group of computers.

MULTIHOMING

Multihoming is the configuration of multiple network interfaces or IP addresses on a single computer. Multihoming is intended to increase the reliability of network applications but it does not necessarily improve their performance. In traditional multihoming, you install a second hardware network adapter on a computer that normally possesses only one. Then, you configure both adapters to utilize the same one local IP address. This setup allows a computer to continue using the network even if one or the other network adapter stops functioning. In some cases, you can also connect these adapters to different Internet/network access points and increase the total bandwidth available to use across multiple applications.

DOCUMENT ROOT

The document root is a directory (a folder) that is stored on your host's servers and that is designated for holding web pages. When someone else looks at your web site, this is the location they will be accessing.

In order for a website to be accessible to visitors, it must be published to the correct directory, the "document root."

JSON: JAVASCRIPT OBJECT NOTATION

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is syntax for storing and exchanging text Information, much like XML. JSON is smaller than XML, and faster and easier to parse.

```
Example :
var Object = {"array": [
    {"class": "one"},
    {"class": "two"},
    {"class": "three"}
  ]
};
```

WHAT IS JSON?

- JSON stands for JavaScript Object Notation.
- JSON is lightweight text-data interchange format.

- JSON is language independent.
- JSON is "self-describing" and easy to understand
- JSON uses JavaScript syntax for describing data objects, but JSON is still language and platform independent. JSON parsers and JSON libraries exists for many different programming languages.

MUCH LIKE XML

- JSON is plain text
- JSON is "self-describing" (human readable)
- JSON is hierarchical (values within values)
- JSON can be parsed by JavaScript
- JSON data can be transported using AJAX

WHY JSON?

For AJAX applications, JSON is faster and easier than XML:

- Using XML
- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

INSTALLATION/CONFIGURATION

As of PHP 5.2.0, the JSON extension is bundled and compiled into PHP by default. Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases downloads, source files, maintainer information, and a CHANGELOG, can be located here: http://pecl.php.net/package/json. This extension has no configuration directives defined in php.ini.

**** Note : This installation/configuration for PHP only as JSON is itself in JavaScript and JavaScript is client side language and it's supported in every browser and operating system.**

RESOURCE TYPES

This extension has no resource types defined.

JSONSERIALIZABLE

Objects implementing JsonSerializable can customize their JSON representation when encoded with json_encode().

```
JsonSerializable {
/* Methods */
abstract public mixed jsonSerialize ( void )
}
```

JsonSerializable::jsonSerialize — Specify data which should be serialized to JSON

- This function has no parameters.
- Returns data which can be serialized by json_encode(), which is a value of any type other than a resource.

```
<?php
class ArrayValue implements JsonSerializable {
    public function __construct(array $array) {
        $this->array = $array;
    }

    public function jsonSerialize() {
        return $this->array;
    }
}

$array = [1, 2, 3];
echo json_encode(new ArrayValue($array), JSON_PRETTY_PRINT);
?>

Output:
[    1,    2,    3]
```

JSON FUNCTIONS

- json_decode — Decodes a JSON string
- json_encode — Returns the JSON representation of a value
- json_last_error_msg — Returns the error string of the last json_encode() or json_decode() call
- json_last_error — Returns the last error occurred

json_encode

json_encode — Returns the JSON representation of a value

string json_encode (mixed $value [, int $options = 0 [, int $depth = 512]])

Returns a string containing the JSON representation of value.

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
echo json_encode($arr);
?>

Output:
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

json_decode

json_decode — Decodes a JSON string

mixed json_decode (string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0]]])

Takes a JSON encoded string and converts it into a PHP variable.

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
var_dump(json_decode($json));
var_dump(json_decode($json, true));
?>
```

Output:

```
object(stdClass)#1 (5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}

array(5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}
```

PHP BASIC

INTRODUCTION TO PHP

- PHP stands for PHP: Hypertext Preprocessor" is widely used as Open Source as general purpose scripting language that is especially suited for web development and can be embedded in HTML.
- PHP was created by RasmusLerdorf in 1995 as PHP/FI [Personal Home Page/Form Interpreter] with simple set of Perl Scripts for tracking accesses to his online resume.
- In November 1997 PHP/FI 2.0 was officially released with C Language implementation, and cult of several thousand users around the world with approximately 50,000 domains.
- Shortly PHP 3.0 as we know it today was created by AndiGutmans and ZeevSuraski as complete rewrite, after they found PHP/FI 2.0 has several underpowered function for developing and eCommerce application they were working on for University Project.
- Andi, Rasmus and Zeev decided to cooperate and announce PHP 3.0 as official successor of PHP/FI 2.0.
- By the end of 1998, PHP grew to an install base of tens of thousands of users (estimated) and hundreds of thousands of Web Sites reporting it installed. About 10% of the Web Servers on the internet.
- Andi and Zeev has begun working on rewrite of PHP's core and new engine, dubbed 'Zend Engine' (Comprised of their first name). And they introduced PHP 4.0 in May 2000 with wide range of additional new features and server on internet record to 20% of total.
- In July 2004 PHP 5 was released with Zend Engine 2.0 with a new object model and dozens of other new features.
- Anything PHP is mainly focused on server side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies and many more.

POWER OF PHP

- Windows desktop applications can also be written, PHP is probably not the very best language to create a desktop application with graphical user interface, but if know PHP very well and would like to use some advanced PHP features in your client side application you can also use PHP-GTK.
- PHP can be used on all major operating systems such as Linux, UNIX variants, Microsoft Windows, Mac OS X, RISC OS and others.
- Also support all major web servers such as Apache, IIS [Microsoft Internet Information Server], Personal web server, Netscape, iPlanet, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd and many others.
- You are also having choice of using procedural programming or object oriented programming. With PHP 5 which fixes the OOP related weaknesses of PHP 4 and introduces a complete object model.

- With PHP you are not limited to output HTML. It has abilities also to outputting images, PDF files and even Flash movies generated on the fly.You can also output easily and text such as XHTML and any other XML file.
- The strongest features of PHP is that it support a wide range of database which are Adabas D, dBase, Empress, IBM DB2, PostgreSQL, SQLite, MySql, Oracle, etc.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

COMMON USES OF PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, and modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

CONFIGURATION OF PHP WITH IIS AND APACHE

TO INSTALL IIS ON WINDOWS SERVER 2012

1. On the **Start** page, click the **Server Manager** tile, and then click **OK**.
2. In **Server Manager**, select **Dashboard**, and click **Add roles and features**.
3. In the **Add Roles and Features Wizard**, on the **Before You Begin** page, click **Next**.
4. On the **Select Installation Type** page, select **Role-based or Feature-based Installation** and click **Next**
5. On the **Select Destination Server** page, select **Select a server from the server pool**, select your server, and click **Next**.
6. On the **Select Server Roles** page, select **Web Server (IIS)**, and then click **Next**.
7. On the **Select Features** page, note the preselected features that are installed by default, and then select **CGI**. This selection also installs FastCGI, which is recommended for PHP applications.
8. Click **Next**.
9. On the **Web Server Role (IIS)** page, click **Next**.
10. On the **Select Role Services** page, note the preselected role services that are installed by default, and then click **Next**.

    Note

    You only have to install the IIS 8 default role services for a static-content web server.
11. On the **Confirm Installation Selections** page, confirm your selections, and then click **Install**.
12. On the **Installation Progress** page, confirm that your installation of the Web Server (IIS) role and required role services completed successfully, and then click **Close**.
13. To verify that IIS installed successfully, type the following into a web browser:
    http://localhost
    You should see the default IIS Welcome page.

TO INSTALL IIS ON WINDOWS 8

1. On the **Start** page, type **Control Panel**, and then click the **Control Panel** icon in the search results.
2. In **Control Panel**, click **Programs**, and then click **Turn Windows features on or off**.
3. In the **Windows Features** dialog box, click **Internet Information Services**, note the preselected features that are installed by default, and then select **CGI**. This selection also installs FastCGI, which is recommended for PHP applications.
4. Click **OK**.
5. To verify that IIS installed successfully, type the following into a web browser:
   http://localhost
   You see the default IIS Welcome page.

CONFIGURE PHP WITH IIS

1. Open your browser to Windows for PHP Download Page and download the PHP 5.3 non-thread-safe zip package.

   Caution

   The PHP 5.4 version does not work with the WinCache extension version 1.1. Use PHP 5.3 until this problem is resolved.
2. Download the WinCache extension (Php_wincache-svn20110402-5.2-nts-vc6-x86.zip) from the List of Windows Extensions for PHP.
3. Extract all files in the PHP .zip package to a folder of your choice, for example C:\PHP\.
4. Extract the WinCache .zip package to the PHP extensions folder (\ext), for example C:\PHP\ext.
   The WinCache .zip package contains one file (Php_wincache.dll).
5. Open **Control Panel**, click **System and Security**, click **System**, and then click **Advanced system settings**.
6. In the **System Properties** window, select the **Advanced** tab, and then click **Environment Variables**.
7. Under **System variables**, select **Path**, and then click **Edit**.
8. Add the path to your PHP installation folder to the end of the **Variable value**, for example **;C:\PHP**. Click **OK**.
9. Open IIS Manager, select the hostname of your computer in the **Connections** panel, and then double-click **Handler Mappings**.
10. In the **Action** panel, click **Add Module Mapping**.
11. In **Request path**, type ***.php**.
12. From the **Module** menu, select FastCgiModule.
13. In the **Executable** box, type the full path to Php-cgi.exe, for example **C:\PHP\Php-cgi.exe**.
14. In **Name**, type a name for the module mapping, for example **FastCGI**.
15. Click **OK**.
16. Select the hostname of your computer in the **Connections** panel, and double-click **Default Document**.
17. In the **Action** panel, click **Add**. Type **Index.php** in the **Name** box, and then click **OK**.
18. Click **Add** again. Type **Default.php** in the **Name** box, and then click **OK**.

INSTALLATION STEPS FOR APACHE 2.0.XX [ANY WINDOWS OS]

1. First of check is there any other server is installed or not. *Control Panel – Administrative tools – services.*
2. If any server found then uninstall. *Control Panel – Add Remove programs [To uninstall].*

3. If you don't want to uninstall the disable it. *Control Panel – Administrative tools – Services.*
4. To install the Apache server, you can download it from http://httpd.apache.org/download.cgi or download it from Google search engine.
5. Execute [Run] the setup. In the first 'Welcome' screen click next.
6. In the agreement screen. Select the 'I accept the …' .Click next
7. Click next on version detail screen.
8. In Server Information screen. Provide Network domain 'localhost.com'. Server name 'www.localhost.com'. Admin e-mail addresses 'me@localhost.com'. Select installation for all users.
9. Select 'Typical' installation type. Click next
10. Change the destination directory if required [Change it to "c:\" i.e. At C drive root]. Click next
11. Click 'Install' to start the installation.
12. If any Windows Security Alert – Firewall boxed is appeared which is trying to block your server. Click on the "Unblock option"
13. Click on Finish button. Also check the system tray, Apache icon should appear with green marked.
14. To check the Apache. Open any browser; type 'http://localhost' in the address bar. Home Page should open of the server

CONFIGURE PHP WITH APACHE

1. You need to insert the following lines into your Apache httpd.conf configuration file to load the PHP module for Apache 2.x:

```
LoadModule php5_module "c:/php/php5apache2.dll"
AddHandler application/x-httpd-php .php
# configure the path to php.ini
PHPIniDir "C:/php"
```

2. Note: Remember to substitute your actual path to PHP for the C:/php/ in the above examples. Take care to use either php5apache2.dll or php5apache2_2.dll in your LoadModule directive and verify that the referenced file is in fact located at the file path that you point to in this directive.
3. The above configuration will enable PHP handling of any file that has a .php extension, even if there are other file extensions. For example, a file named example.php.txt will be executed by the PHP handler. To ensure that only files that end in .php are executed, use the following configuration instead:

```
<FilesMatch \.php$>
   SetHandler application/x-httpd-php
</FilesMatch>
```

PHP.INI

- The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality. The php.ini file is read each time PHP is initialized.

- In other words, whenever https is restarted for the module version or with each script execution for the CGI version. If your change isn't showing up, remember to stop and restart https. If it still isn't showing up, use phpinfo () to check the path to php.ini.
- The configuration file is well commented and thorough. Keys are case sensitive, keyword values are not; whitespace, and lines beginning with semicolons are ignored.
- Booleans can be represented by 1/0, Yes/No, On/Off, or True/False. The default values in php.ini-dist will result in a reasonable PHP installation that can be tweaked later.
- Here we are explaining the important settings in php.ini which you may need for your PHP Parser.

- *short_open_tag = Off*

Short open tags look like this: <? ?>. This option must be set to Off if you want to use XML functions.

- *max_execution_time = 30*

The function set_time_limit() won.t work in safe mode, so this is the main way to make a script time out in safe mode. In Windows, you have to abort based on maximum memory consumed rather than time. You can also use the Apache timeout setting to timeout if you use Apache, but that will apply to non-PHP files on the site too.

- *error_reporting = E_ALL & ~E_NOTICE*

The default value is E_ALL & ~E_NOTICE, all errors except notices. Development servers should be set to at least the default; only production servers should even consider a lesser value

- *include_path = [DIR]*

If you set this value, you will only be allowed to include or require files from these directories. It include directory is generally under your document root; this is mandatory if you.re running in safe mode. Set this to .in order to include files from the same directory your script is in. Multiple directories are separated by colons: .:/usr/local/apache/htdocs:/usr/local/lib.

- *doc_root = [DIR]*

If you.re using Apache, you.ve already set a document root for this server or virtual host in httpd.conf. Set this value here if you.re using safe mode or if you want to enable PHP only on a portion of your site (for example, only in one subdirectory of your Web root).

- *file_uploads = [on/off]*

Turn on this flag if you will upload files using PHP script.

- *upload_tmp_dir = [DIR]*

Do not uncomment this line unless you understand the implications of HTTP uploads!

- *mysql.default_host = hostname*

The default server host to use when connecting to the database server if no other host is specified.

- *mysql.default_user = username*

The default user name to use when connecting to the database server if no other name is specified.

- *mysql.default_password = password*

The default password to use when connecting to the database server if no other password is specified.

.HTACCESS

Hypertext Access, commonly shortened to .htaccess, is a configuration file which controls the directory it is placed in and all the subdirectories underneath it.

It's an incredibly useful feature which allows webmasters to control how many aspects of their website works. You can 301 redirect pages, change the extensions of pages, rewrite URL for better keyword ranking presence, password protect directories, Error 404 Document redirect and much more.

'.htaccess' is the filename in full, it is not a file extension. For instance, you would not create a file called, 'file.htaccess', it is simply called, '.htaccess'. This file will take effect when placed in any directory which is then in turn loaded via the Apache Web Server software. The file will take effect over the entire directory it is placed in and all files and subdirectories within the specified directory.
You can create a .htaccess file using any good text editor such as TextPad, UltraEdit, Microsoft WordPad and similar (you cannot use Microsoft NotePad).
Here is an example of what you might include in a .htaccess file.

```
AuthName "Member's Area Name"
AuthUserFile /path/to/password/file/.htpasswd
AuthType Basic
require valid-user
ErrorDocument 401 /error_pages/401.html
AddHandler server-parsed .html
```

- Allow/Deny Directory Browsing

When directory browsing is on, people accessing a URL from your site with no index page or no pages at all, will see a list of files and folders. To prevent such directory access, just place the following line in your .htaccess file.

```
IndexIgnore */*
```

Many hosting companies, by default deny directory browsing and having said that, just in case you need to enable directory browsing, place the following line in your .htaccess file.

```
Options +Indexes
```

- Change the default index page of a directory or site

Almost every hosting company will have index.htm, index.html, index.php, index.asp, default.asp, default.html as the default index page names in their web server settings. So, in case your site or directory does not has a file name which matches a name from the list above, chances are that your visitors will either see a list of all the files and folders [through directory browsing] or will not see anything at all. To change the default index page's name for a directory or the site, place the following line in the .htaccess file of the root folder or the particular directory for which you want to change the index page's name.

```
DirectoryIndex homepage.htm
DirectoryIndex somepage.htm
```

- Preventing hot linking of images from your website

If your website contains images which people from other websites are linking to and you get charged for the extra bandwidth, then placing the following lines will prevent any such image hot linking.

Most of the hosting companies provide this feature in their control panel itself, such as CPanel. This trick requires mod_rewrite engine to be on in Apache on your web server.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://(www\.)?your-domain.com/.*$
[NC]
RewriteRule .(gif|jpg)$ - [F]
```

In the above code, replace [your-domain] with your actual domain name [without www], and instead of (www.\), use your actual subdomain name (sub-domain.\)

BASIC PHP SYNTAX

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
      <body>
            <?php echo "Hello World"; ?>
      </body>
</html>
```

PHP is not the case-sensitive language in Windows OS but in other OS it's 100% case sensitive.

```
<?php
      echo "hello";
      for($i=0;$i<=10;$i++)
      {
            echo "hello";
      }
?>
```

A PHP scripting block always starts with <?php and ends with ?>. A PHP scripting block can be placed anywhere in the document and for any number of times. Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another same as C language. There are two basic statements to output text with PHP: echo and print same as 'printf' of 'C Language'. In the example above we have used the echo statement to output the text "Hello World".

The Php script block can also be started with <script language="php"> and should end with </script>. Other two option are there but they are not the default available option in the Php. They have to be configured in the configuration file of Phpi.e php.ini. Which will available in directory where Php is been installed. The first option is known as short tags i.e. <? ?>. To use it search the statement "short_open_tag = off" in the php.ini file replace it with "short_open_tag = on". Another is Asp style tags i.e. <% %>. To use it search the statement "asp_tags = off" replace it with "asp_tags = on".

HOW TO RUN PHP PROGRAM IN LOCAL COMPUTER

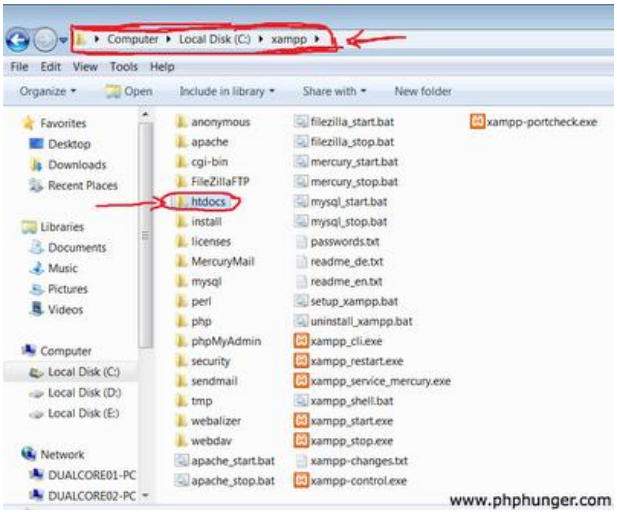
Start XAMPP Control Panel Application and start **Apache and MySQL services**.

1. To run PHP code or PHP script you need a file. So create a file with a name called "myFirstPHPScript.php".
2. To create a file you should have any editor like Notepad or Notepad++.

File Name : *myFirstPHPScript.php*

```
<?php
      echo "Hello World! This is my first PHP code snippet!!";
?>
```

Where to save PHP files and folders?
The XAMPP gives you a directory for storing all PHP files or folders in the "***htdocs"*** directory.



Local Storage Location of PHP files and folders:

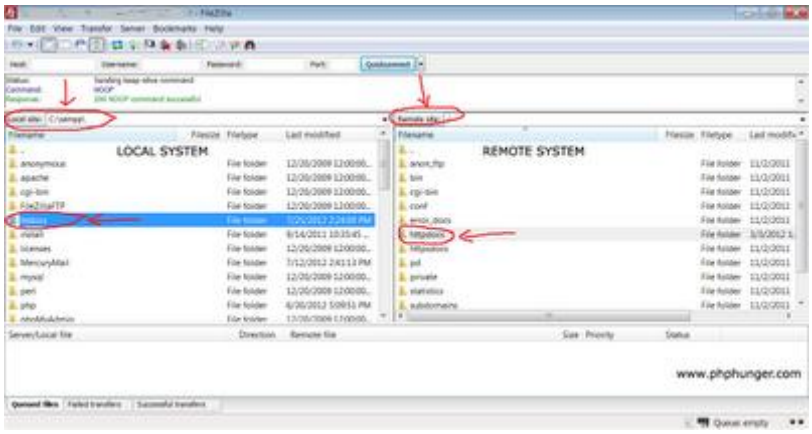
Let's see the location of **htdocs** directory. See the below picture to find the location of **"htdocs"**directory.

Note:
Every PHP file or PHP Project must be stored in this htdocs directory.

If it's a big project then keep all the PHP, HTML, CSS, JavaScript files in one folder and place them in the htdocs directory.

REMOTE STORAGE LOCATION OF PHP FILES AND FOLDERS

Let's see the remote storage location of PHP files and folders. Like htdocs in local system,**httpdocs** in the remote system where you will store all the PHP files and folders. See the below picture.

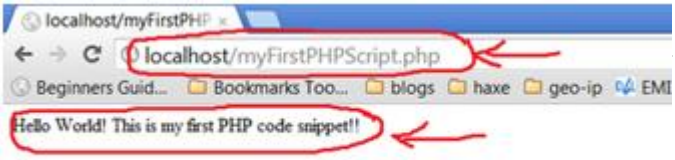


Note :

1. For storing files or folders in the remote location you need to use a tool called FileZilla. I have already discussed about this tool in my other article please visit this for better understanding.
2. Simply transfer the files from local system to remote system by using the respective credentials like hostname, username, password and port number for the respective website. Here our scope is for local system only. How to work with Remote system will be covered in future articles. Just keep this in mind.

As you have written the code, saved the file and what's left. Now how can i run this code. I am eagerly waiting to see the result of my code. Let's see.

How to run the PHP code or PHP Script?



1. Open any web browser(Chrome, IE, Firefox, Safari, Opera etc.)
2. Simply type **http://localhost/myFirstPHPScript.php** in the address bar. See below picture.

Points to remember about the PHP code file:

1. Every PHP file you write must start with the **<?php** and ends with **?>**. Its a must rule for every PHP file. If you forgot to write like this then **PHP Parser** simply treats the file as atext file and displays the content as it is.
2. Every PHP file must be saved with **".php"** extension.
3. The PHP file name may be a combination of letters or lower case and upper case letters or combination of letters with numbers.
4. A single PHP file may contain as many as starting and closing tags. Like see below code.

```
<?php
      echo "Hello World! This is my first PHP code snippet!!";
?>
<?php
      echo "Hello World! This is 2nd opening and closing tag!!";
?>
<?php
      echo "Hello World! This is 3rd opening and closing tag!!";
?>
<?php
      echo "Hello World! This is 4th opening and closing tag!!";
?>
```

COMMENTS

PHP supports 'C', 'C++' and UNIX shell-style (Perl style) comments

```
<?php
      echo 'This is a test'; // This is a one-line c++ style comment
      /* This is a multi line comment
      yet another line of comment */
      echo 'This is yet another test';
      echo 'One Final Test'; # This is shell-style style comment
?>
```

Double forward slash and the hex sign provide single line comment. Forward slash with asterisk provide the multiline comment

VARIABLES IN PHP

- All variables in PHP start with a $ sign symbol.
- After the $ symbol first character should be the alphabet or underscore.
- Combination of alphabet and digit can be used in name.
- Word should not be separated by space or any other symbol or sign but underscore can be used.
- Variables may contain strings, numbers, or arrays.
- Reserve words or keywords are not allowed.

Below, the PHP script assigns the string "Hello World" to a variable called $txt:

```
<?php
      $txt="Hello World";echo $txt;
?>
```

To concatenate two or more variables together, use the dot (.) operator or (,) comma:

```
<?php
      $txt1="Hello World";  $txt2="1234";
      echo $txt1." ".$txt2."<br>";
      echo $txt1.$txt2."<br>" ;
      echo $txt1,$txt2."<br>";
?>
```

PREDEFINED VARIABLES

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server are running, the version and setup of the server, and other factors. Some of the predefined variables are $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIE, $_FILES, $_ENV, $_REQUEST, $_SESSIONS

VARIABLE SCOPE

LOCAL VARIABLE

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope.

```
<?php
      $a = 1;
      function test(){
            $b=10;
            echo "A=".$a;     //Prints "A='
            echo "B=".$b;    //Prints "B=10"
      }
      test();
      echo "A=".$a;    //Prints "A=1"
      echo "B=".$b;    //Prints "B="
?>
```

GLOBAL VARIABLE

To make the variable global 'global' keyword is used. There is no limit to the number of global variables that can be manipulated by a function.

```
<?php
      $a = 1; $b = 2;
      function sum(){
            global $a, $b;   $b = $a + $b;
      }
      sum();      echo $b;
```

```
?>
```

The another option to use variable globally is to used predefined variable $GLOBALS

```
<?php
      $a = 1;
      $b = 2;
      function sum()
      {
            echo $GLOBALS['a'];
            echo "<br>";
            echo $GLOBALS['b'];
      }
      sum();
?>
```

The $GLOBALS array is an array with the name of the global variable being the key and the contents of that variable being the value of the array element. $GLOBALS exists in any scope, this is because $GLOBALS is a super global. $GLOBALS contains a reference to every variable which is currently available within the global scope of the script. $GLOBALS has existed since PHP 3.

STATIC VARIABLE

If the variable is the local (variable for any function) then value of variable exists till the control is in the function. Once the control gets out variable value is not available, and again getting the control to the same function any how will reinitialize the variable value. To get the previous value 'static' keyword is used to make the variable static.

```
<?php
      function test(){
      static $count1 = 0;
         $count2 = 0;
         $count1++;    $count2++;
      echo "#Count 1 : ".$count1;
      echo "#Count 2 : ".$count2;
      echo "<br>";
      }
      test(); test(); test();
?>
```

OPERATORS

ARITHMETIC OPERATORS

Sign	Description	Effect	Example	Result
+	Addition	Sum of variable or values	$x=2 $x+2	4
-	Subtraction	Subtraction of variable or values	$x=2 5-$x	3
*	Multiplication	Multiplication of variable or values	$x=4 $x*5	20

/	Division	Division of variable or values	15/5 5/2	3 2.5
%	Modulus	Reminder of first variable or value by second	5%2 10%8	1 2
++	Increment	(Post) Return and increment. (Pre) Increment and return	$x=5 $x++ ++$x	6 7
--	Decrement	(Post) Return and decrement. (Pre) Decrement and return	$x=5 $x-- --$x	4 3

ASSIGNMENT OPERATORS

Sign	Description	Effect	Example	Result
=	Assign	Transfer the value	$x=$y	$x=$y
+=	Sum	Add and transfer	$x+=$y	$x=$x+$y
-=	Subtract	Subtract and transfer	$x-=$y	$x=$x-$y
***=**	Multiply	Multiply and transfer	$x*=$y	$x=$x*$y
/=	Divide	Divide and transfer	$x/=$y	$x=$x/$y
%=	Modulus	Modulus and transfer	$x%=$y	$x=$x%$y

COMPARISON OPERATORS

Sign	Description	Effect	Example	Result
==	is equal to	True if both value are equal	5=="5"	True
===	strictly equal	True if both values and data type are equal	"5"===5	False
!=	is not equal	True if both are not equal	5!=8	True
>	is greater than	True if first is greater than second	5>8	False
<	is less than	True if first is less then second	5<8	True
>=	is greater than or equal to	True if first is greater than or equal to second	5>=8	False
<=	is less than or equal to	True if first is less than or equal to second	5<=8	True

LOGICAL OPERATORS

Sign	Description	Effect	Example	Result
&&	And	True if both are true	$x=6 $y=3 ($x<10 && $y > 1)	True
||	Or	True if both or any of it true	$x=6 $y=3 ($x==5 || $y==5)	False
!	Not	True if false. False if true	$x=6 $y=3 !($x==$y)	True
and	And	True if both are true	5>8	False
or	Or	True if both or any of it true	5<8	True
not	Not	True if false. False if true	5>=8	False

STRING OPERATOR

Sign	Description	Effect	Example	Result
.	Dot	Merge two string	$x="A" $y="B $x .= $y	"AB"

ERROR OPERATOR

Sign	Description	Effect	Example	Result
@	Error	Hides Warring.	@$x="A"	"A"

TERNARY OPERATOR

Sign	Description	Effect	Example	Result
?:	Same as if	True and False part in single line	$max= ($a>$b)?$a:$b	Value of $a

CONTROL STRUCTURE (SELECTION CONTROL)

IF STATEMENT

If statement is used when a part of statement is to be executed or when not to be executed. When the condition following 'if' is true then it will excuted the the true part. If the condition is false then it executed the else part.

```
if(condition)
    do this;
```

```
if(condition)
{
    do this;
    and this;
}
```

```
if(condition)
{
   if(conditon)
   {
      do this;
      and this;
    }
    also do this;
}
else
   do this;
```

```
if(condition)
   do this;
else
   do this;
```

```
if(condition)
{
   do this;
   and this;
}
else
   do this;
```

```
<?php
      if($a > $b)
            echo "a is bigger than b";
      elseif($a == $b)
            echo "a is equal to b";
      else
            echo "a is smaller than b";
?>
```

SWITCH

The Switch statement in PHP is used to perform one of several different actions based on one of several different conditions. We may compare the same variable with different values and if we want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

```
switch(expression)
{
    case :
        do this and
exit
}
```

```
switch(expression)
{
    case :
        do this and
exit
    case :
        do this and
exit
}
```

```
switch(expression)
{
    case :
    case :
        do this and
exit
    case :
    case :
        do this and
exit
}
```

```
switch(expression)
{
    case :
        do this and
exit
    case :
        do this and
exit
    case :
        do this and
exit
    case :
        do this and
exit
    default:
        do this
}
```

```
<?php
      $i="bar";
      switch($i){
            case "apple":
                  echo "i is apple";
                  break;
            case "bar":
                  echo "i is bar";
                  break;
            case "cake":
                  echo "i is cake";
                  break;
      }
?>
```

```
<?php
      $i=5;
      switch($i){
            case 0:
            echo "i equals 0";
            break;
            case 1:
            echo "i equals 1";
            break;
            case 2:
            echo "i equals 2";
            break;
            default:
            echo "i is not equal to 0, 1 or 2";
```

```
      }
?>
```

CONTROL STRUCTURE (ITERATION CONTROL)

WHILE LOOP

Looping statements in PHP are used to execute the same block of code a specified number of times. It tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statement(s) won't even be run once.

```
while(expression)
    do this;
```

```
while(expression)
{
    do this;    and this;
}
```

```
while(expression)
{
   do this;
   while(expression)
   {
        do this;
        and this;
   }
}
```

```
<?php
      $i=1;
      while($i<=5){
            echo "The number is ";
            echo $i."<br>";
            $i++;
      }
?>
```

DO- WHILE LOOP

The do…while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true. Same as the C language while and for loop can be said as entry control loop where else do-while is exit control loop

```
do
{
       do this;
       and this;
}while(expresion);
```

```
do
{
       do this;
       and this;
      do
      {
           do this
       }while(expresion);
}while(expresion);
```

```
<?php
      $i = 5;
```

```
      do{
            echo $i;
            $i--;
      } while ($i > 0);
?>
```

FOR LOOP

The For statement is used when you know how many times you want to execute a statement or a list of statements. The For statement has three parameters. The first parameter initializes variables, the second parameter holds the condition, and the third parameter contains the increments required to implement the loop. If more than one variable is included in the initialization or the increment parameter, they should be separated by commas. The condition must evaluate to true or false.

```
for(initialization; condition; inc/dec)
 do this;
```

```
for(initialization; condition; inc/dec)
{
     do this;
     and do this;
}
```

```
for (initialization; condition; inc/dec)
{
    do this;
    for (initialization; condition; inc/dec)
          do this if second for condition is true
}
```

```
<?php
      for ($i = 1;$i > 10 ;$i++){
            echo $i;
      }
?>
```

FOREACHLOOP

The foreach statement is used to loop through arrays. For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop, you'll be looking at the next element.PHP 4 introduced a foreach construct, much like Perl and some other languages. This simply gives an easy way to iterate over arrays. foreach works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable.

```
foreach(array_expression as $value)
   statements;
```

```
foreach(array_expression as  $key => $value)
   statements;
```

```
<?php
      $arr = array(1, 2, 3, 4);
      foreach ($arr as $value) {
            echo $value." ";
      }
```

```
?>
```

```
<?php
	$arr = array(1, 2, 3, 4);
	foreach ($arr as $value) {
		$value *=2;
		echo $value." ";
	}
?>
```

```
<?php
	$arr = array("one", "two", "three");
	foreach ($arr as $key => $value) {
		echo "Key: $key; Value: $value<br>";
	}
?>
```

```
<?php
	foreach (array(1, 2, 3, 4, 5) as $v)
	echo "$v<br>";
?>
```

```
<?php
	$arr = array(1, 2, 3, 4);
	foreach ($arr as $k=>$v)
		$a[$k]=$v*2;
	echo "<pre>";	print_r($a);	echo "</pre>";
?>
```

CONTROL STRUCTURE (UNCONDITIONAL CONTROL)

BREAK

Break ends execution of the current for, foreach, while, do-while or switch structure. Break accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
<?php
	for ($i = 1; ; $i++) {
		if ($i==5)
			break;
		echo $i;
	}
?>
```

```
<?php
```

```
      $i = 0;
      while (++$i) {
      switch ($i) {
      case 5:
      echo "At 5<br />\n";
      break 1;  /* Exit only the switch. */
      case 10:
      echo "At 10; quitting<br />\n";
      break 2;  /* Exit the switch and the while. */
      default:
      break;
          }
      }
?>
```

CONTINUE

Continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the beginning of the next iteration.

```
<?php
      for ($i = 0; $i < 5; ++$i) {
            if ($i == 2)
                  continue;        print $i;
      }
?>
```

```
<?php
      $i = 0;
      while ($i++ < 5) {
      echo "Outer<br />\n";
      while (1) {
      echo "  Middle<br />\n";
      while (1) {
      echo "  Inner<br />\n";
                        continue 3;
              }
      echo "This never gets output.<br />\n";
          }
      echo "Neither does this.<br />\n";
      }
?>
```

ARRAY

An array is a collection of variables indexed and bundled together into a single, easily referenced super-variable that offers an easy way to pass multiple values between lines of code, function, and even pages. There are three kinds of array.

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables it's easy to define an array of 100 lengths.
There are three different kinds of arrays and each array value is accessed using an ID c which is called array index.

- Numeric Array: These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
- Associative Array: The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.
- Multidimensional Arrays: A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple indexes.

COMPARE TO C LANGUAGE

- Array key/index can start with any number.
- Array key can be string, character, and positive/negative integer/float number.
- Values can be added or remove even after declaration of array so size of array is not fixed like C Language.
- Array Key can also be skipped. Memory will be occupied for stored values only (no null or garbage concept).
- Array key can be set automatically to incremented.

Values of the keys are to be passed by the equal to greater then sign (=>). Key name could be string, integer values or both. If the key is the integer value it could start with the any of the integer number not necessary to be start with 0 always. Key name can also be negative value.

```
<?php
      $arr = array(4,5,6);
      print_r($arr);
?>
```

```
<?php
      $arr = array(4,5,6);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

```
<?php
      $arr = array(4,5,6);
      echo $arr[0]." ";
      echo $arr[1]." ";
      echo $arr[2]." ";
?>
```

print_r is special function which can be used to display the array and the ID key of the value. To display the individual key value 'echo' can also be used. For better format '<pre>' tag of Html can also be used.

NEGATIVE INTEGER KEY

```
<?php
      $arr = array(-2=>4,1=>45,5=>00,1,2);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

```
<?php
      $arr = array(-2=>4,2=>45,1,2,6=>3);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

STRING - NEGATIVE INTEGER – POSITIVE INTEGER KEY

```
<?php
      $arr = array(-2=>4,'a'=>45,1,2,6=>3);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

```
<?php
      $arr = array('a'=>4,'z'=>45);
      echo $arr['z'];     echo $arr['a'];
?>
```

```
<?php
      $arr=array(-5=>4,3=>5,'b'=>6,4,'c'=>6,7);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

If in the array [] brackets are left blank then auto key will be assigned. i.e. considering highest positive integer key from existing element, and +1 to that [if highest key is 4 then 5 will auto allocated]. If brackets are left blank then 0 will be allocated if no element exists.

AUTO INCREMENT KEY

```
<?php
```

```
      $arr = array();
      $arr[] = 10;    $arr[3] = 10;    $arr[] = 10;
      $arr['a'] = 10;    $arr[6] = 10; $arr[] = 10;
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

```
<?php
      $arr=array(-2=>4,5,6,-7=>4,6,7);
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

MULTIDIMENSIONAL ARRAY

```
<?php
      $arr = array("a" => array(6 => 5, 13 => 9, "a" => 42));
      echo "<pre>";
      echo $arr["a"][6];   echo $arr["a"][13];
      echo $arr["a"]["a"];
      echo "</pre>";
?>
```

Multi nesting dimensional array

```
<?php
      $arr  =  array("a"  =>  array(6  =>  5,  13  =>  9,  "b"  =>
array(12,3,4,5)));
      echo "<pre>";    print_r($arr);    echo "</pre>";
?>
```

OVERWRITING THE VALUES, string ID as integer string with '0' and mismatch values.

```
<?php
      $switching = array (10, 5 => 6, 3=>TRUE,'a' => 4,11, '8' => 2,
'02' => 77, 0 => 12);
      echo "<pre>";
      print_r($switching);
      echo $switching[0]." ";     echo $switching[08]." ";
      echo $switching[8]." ";     echo
$switching['02']." ";
      echo $switching[2]." ";
      echo "</pre>";
?>
```

UDF – USER DEFINE FUNCTION

UDF the 'User Define Function' same as the 'C Language' known as the function in PHP. A function is a block of code that can be executed whenever we need it.

CREATING PHP FUNCTIONS:

- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Other rules will be same as 'C Language'
- PHP also provides all 4 kinds (types) of function.
- Any valid PHP code may appear inside a function, even other functions and class definitions.
- In PHP 3, functions must be defined before they are referenced. No such requirement exists since PHP 4, except when a function is conditionally.

```
<?php
      //without argument without return
      function test()  {
            $a=5;       echo $a;
      }
      test();
?>
```

```
<?php
      //without argument with return
      function test()  {
            $a=5;
            return $a;
      }
      $b=test();
      echo $b;
?>
```

```
<?php
      //with argument without return
      function test($a){
            echo $a;
      }
      $a="Mohammad"
      test($a);
?>
```

```
<?php
      //with argument with return
      function test($val){
            $a=5;
            $b=$val;
            echo $b."<br>";
            return $a;
```

```
      }
      $b=test(10);
      echo $b;
?>
```

DEFAULT ARGUMENTS

By default, function arguments are passed by value (so that if you change the value of the argument within the function, it does not get changed outside of the function). If you wish to allow a function to modify its arguments, you must pass them by reference.

```
<?php
      function test(&$string){
         $string .= 'and this will be added to main string .';
      }
      $s = "This is a main string to which another will be add,";
      test($s);  echo $s;
?>
```

A function may define C++-style default values as arguments.

```
<?php
      function test($a = "abc"){
      return $a;
      }
      echo test();     echo test("xyz");
?>
```

Incorrect usage of default function arguments

```
<?php
      function test($a = "abc", $b){
      return $a."<Br>".$b;
      }
      echo test("xyz");
?>
```

Use of error operator in such kinds of warning

```
<?php
      function test($a = "abc", $b){
      return $a."<Br>".$b;
      }
      echo @test("xyz");
?>
```

VARIABLE FUNCTION

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates

to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

```
<?php
      function test1(){
      echo "In test1()<br>";
      }
      function test2($arg = ' '){
      echo "In test2(); argument was '$arg'.<br>";
      }
      functionechoit($string){
      echo "In test2(); argument was '$string'.<br>";
      }

      $func = 'test1';
      $func();        // This calls test1()

      $func = 'test2';
      $func('test');  // This calls test2()

      $func = 'print this';
      $func('testing');  // This calls echoit()
?>
```

VARIABLE LENGTH ARGUMENT FUNCTION

FUNC_NUM_ARGS

Returns the number of arguments passed to the function. func_num_args() will generate a warning if called from outside of a user-defined function. This function cannot be used directly as a function parameter. Instead, its result may be assigned to a variable, which can then be passed to the function.

```
<?php
      function test() {
         $totargs = func_num_args();
      echo "Number of arguments: $totargs";
      }
      $a=10;
      test(1,$a, 2,'b');    // Prints 'Number of arguments: 4'
?>
```

FUNC_GET_ARG

Return an item from the argument list. If arg_num is greater than the number of arguments actually passed, a warning will be generated and func_get_arg() will return FALSE.

```
<?php
      function test() {
      $totargs = func_num_args();
```

```
      echo "Number of arguments: $totargs<br>";
      echo "First agrument is :: ".func_get_arg(0)."<Br>";
      echo "Second agrument is :: ".func_get_arg(1)."<Br>";
      echo "Third agrument is :: ".func_get_arg(2)."<Br>";
      echo "Fourth agrument is :: ".func_get_arg(3)."<Br>";
      }
      $a=10;
      test(1,$a, 2,'b');    // Prints 'Number of arguments: 4'
?>
```

FUNC_GET_ARGS

Returns an array in which each element is the corresponding member of the current user-defined function's argument list. func_get_args() will generate a warning if called from outside of a function definition. This function cannot be used directly as a function parameter. Instead, its result may be assigned to a variable, which can then be passed to the function.

```
<?php
      function test() {
            $totarg = func_num_args();
            echo "Number of arguments: $totarg<br />";
            $args = func_get_args();
            foreach($args as $k => $v){
                  echo "Key is :: $k and Value is :: $v<br />";
            }
      }
      test(1, 2, 3);
?>
```

ARRAY FUNCTIONS

ARRAY_DIFF ( )

Compare arrays for unique value differences

The array_diff() PHP array function will find unique values in a target array when you compare other arrays against it. Only unique values are placed into the output array. This function only evaluates array item values and does not evaluate any keys for those values. To evaluate both keys and values in this way use the array_diff_assoc() function instead of this one.

It will take as many arrays as parameters that you wish to compare with the target array. Parameter one is the target array and all arrays to be evaluated for differences against the target are to be used as parameters following the first and separated by commas.

array_diff() will return an associative array containing any unique values(and their keys) found after it processes.

array_diff(array1,array2,array3...);

array1 Required. The array to compare from
array2 Required. An array to compare against
array3,... Optional. More arrays to compare against

Return Value: Returns an array containing the entries from array1 that are not present in any of the other arrays

```
<?php
	$arr1=array("a"=>"Monkey","b"=>"Lion","c"=>"Tiger","d"=>"Cat")
;
	$arr2=array("e"=>"Dog","f"=>"Donkey","g"=>"Horse","h"=>"Monkey
");

	$result=array_diff($arr1,$arr2);
	print_r($result);

	$arr3 = array("CE","EE","TE");
	$arr4 = array("EE","BE","ME");
	$arr5 = array("IM","TE","ME");
	$result=array_diff($arr3,$arr4,$arr5);
	print_r($result);
?>
```

ARRAY_KEY_EXISTS ( )

Check to see if a key exists as an index in an array. The array_key_exists() array function in PHP will return either "true" or "false" when it runs. It takes two parameters to operate. The first parameter is the key that you wish to find out if it exists, and the second parameter is the target array to search in for that key or index.

array_key_exists(key,array)

key Required. Specifies the key

array Required. Specifies an array

Return Value: Returns TRUE if the key exists and FALSE if the key does not exist

```
<?php
	$targetArray = array("a" => "orange", "b" => "apple");
	$keyToFind = "b";
	if (array_key_exists($keyToFind, $targetArray)) {
		echo "<u>$find</u> is found as a key in the array";
	} else {
		echo "<u>$find</u> is not found as a key in the array";
	}
?>
```

ARRAY_KEYS ( )

Use keys in a target array to form the values in a new array. The array_keys() array function in PHP will return an array using keys from a target array. The keys will become values in the new array and they will have a default numeric index starting at 0. It can take up to three parameters. The first parameter is the target array that you wish to run through the function. The second optional parameter is for specifying a value in which only those keys with that value are placed into the output array. And the third optional parameter is to specify strict comparison in the search using "true" or "false".

array_keys(array,value,strict)
array Required. Specifies an array
value Optional. You can specify a value, then only the keys with this value are returned
strict Optional. Used with the value parameter. Possible values:
true - Returns the keys with the specified value, depending on type: the number 5 is not the same as the string "5".
false - Default value. Not depending on type, the number 5 is the same as the string "5".
Return Value: Returns an array containing the keys

```
<?php
      $targetArray = array("a" => "orange", "b" => "apple", "c" =>
"grape");
      $newArray = array_keys($targetArray);
      foreach ($newArray as $key => $value)
                  echo "$key - <strong>$value</strong><br />";
?>
```

ARRAY_MERGE ( )
Merge multiple arrays. The array_merge() array function in PHP will merge multiple arrays. This function takes a list of arrays separated by commas as its parameters. If your arrays are associative arrays the output array will be a multidimensional array as depicted in code Example 2 below.
array_merge(array1,array2,array3...)
array1 Required. Specifies an array
array2 Optional. Specifies an array
array3,... Optional. Specifies an array
Return Value: Returns the merged array

```
<?php
      $array1 = array("orange", "apple", "grape");
      $array2 = array("peach", 88, "plumb");
      $array3 = array("lemon", 342);
      $newArray = array_merge($array1, $array2, $array3);
      foreach ($newArray as $key => $value)
            echo "$key - <strong>$value</strong><br />";
?>
```

ARRAY_MERGE_RECURSIVE ( )
Recursively merge multiple arrays
The array_merge_recursive() array function in PHP will recursively merge multiple arrays. This function takes a list of arrays separated by commas as its parameters. If your arrays are associative arrays, the output array will be a multidimensional array.
array_merge_recursive(array1,array2,array3...)
array1 Required. Specifies an array
array2 Optional. Specifies an array
array3,... Optional. Specifies an array
Return Value: Returns the merged array

```
<?php
      $array1 = array("orange", "apple", "grape", "banana");
      $array2 = array("peach", "apricot", "pineapple");
      $array3 = array("blueberry", "cherry");
      $newArray = array_merge_recursive($array1, $array2, $array3);
      foreach ($newArray as $key => $value)
                  echo "$key - <strong>$value</strong><br />";
?>
```

ARRAY_SHIFT()

The array_shift() function removes the first element from an array, and returns the value of the removed element. If the keys are numeric, all elements will get new keys, starting from 0 and increases by 1

array_shift(*array*)

array Required. Specifies an array

Return Value: Returns the value of the removed element from an array, or NULL if the array is empty

```
<?php
      $hobbies = array("Acting", "Drawing", "Music", "Films",
"Photography");
      // Deleting first array item
      $removed = array_shift($hobbies);
      print_r($hobbies);
      echo "<br>";
      var_dump($removed);
?>
```

ARRAY_SLICE()

The array_slice() function returns selected parts of an array. If the array have string keys, the returned array will always preserve the keys

array_slice(array,start,length,preserve)

array Required. Specifies an array

start Required. Numeric value. Specifies where the function will start the slice. 0 = the first element. If this value is set to a negative number, the function will start slicing that far from the last element. -2 means start at the second last element of the array.

length Optional. Numeric value. Specifies the length of the returned array. If this value is set to a negative number, the function will stop slicing that far from the last element. If this value is not set, the function will return all elements, starting from the position set by the start-parameter.

preserve Optional. Specifies if the function should preserve or reset the keys. Possible values:

true - Preserve keys

false - Default. Reset keys

Return Value: Returns selected parts of an array

```
<?php
      $myArray = array("John","William","Henry","Tom","Peter");
      $slice = array_slice($myArray, 1, 2);

      foreach($slice as $key => $value) {
            echo "$key - $value <br />";
      }
?>
```

ARRAY_REVERSE()

The array_reverse() function returns an array in the reverse order.

array_reverse(array,preserve)

array Required. Specifies an array

preserve Optional. Specifies if the function should preserve the keys of the array or not. Possible values:true or false

Return Value: Returns the reversed array

```
<?php
      $input  = array("php", 4.0, array("green", "red"));
      $reversed = array_reverse($input);
      $preserved = array_reverse($input, true);

      print_r($input);
      print_r($reversed);
      print_r($preserved);
?>
```

ARRAY_UNIQUE()

The array_unique() function removes duplicate values from an array. If two or more array values are the same, the first appearance will be kept and the other will be removed. The returned array will keep the first array item's key type.

array_unique(array)

array Required. Specifying an array

Return Value: Returns the filtered array

```
<?php
      $myArray = array("cat","dog","bird","cat","rabbit");
      $newArray = array_unique($myArray);

      foreach($newArray as $key => $value) {
            echo "$key - $value <br />";
      }
?>
```

ARRAY_UNSHIFT()

The array_unshift() function inserts new elements to an array. The new array values will be inserted in the beginning of the array. Numeric keys will start at 0 and increase by 1. String keys will remain the same.

array_unshift(array,value1,value2,value3...)

array Required. Specifying an array
value1 Required. Specifies a value to insert
value2 Optional. Specifies a value to insert
value3 Optional. Specifies a value to insert
Return Value: Returns the new number of elements in the array

```
<?php
	$myArray = array("cat","dog","bird");
	array_unshift($myArray, "rabbit");

	foreach($myArray as $key => $value) {
		echo "$key - $value <br />";
	}
?>
```

ARRAY_SEARCH()

The array_search() function search an array for a value and returns the key.

array_search(value,array,strict)

value Required. Specifies the value to search for
array Required. Specifies the array to search in
strict Optional. If this parameter is set to TRUE, then this function will search for identical elements in the array. Possible values: true or false - Default. When set to true, the number 5 is not the same as the string 5
Return Value: Returns the key of a value if it is found in the array, and FALSE otherwise. If the value is found in the array more than once, the first matching key is returned.

```
<?php
	$targetArray = array("Sara","Cindy","Julie","Megan");
	$find = "Julie";
	$keyFound = array_search($find, $targetArray);
	if ($keyFound) {
		echo "The key where $find was found is: $keyFound";
	} else {
		echo "$find was NOT FOUND in the array";
	}
?>
```

ARRAY_MULTISORT ()

Sort multiple arrays according to the sort order of the first. The array_multisort() array function in PHP will sort multiple and multidimensional arrays. It takes a list of arrays separated by commas as

parameters and uses the first array as the master array. All other arrays are sorted according to the way in which the first array has been sorted. So it sorts the first array according to values, and then sorts the following arrays according to the way the first was sorted.

It may seem to you that the function is not sorting your following arrays correctly but this function is not meant to sort many arrays individually, it is meant to sort multiple arrays according to the sort order of the master array.

array_multisort(array1,sortingorder,array2,array3...)

array1 Required. Specifies an array

sortingorder Optional. Specifies the sorting order. Possible values:

SORT_ASC - Default. Sort in ascending order (A-Z)

SORT_DESC - Sort in descending order (Z-A)

array2 Optional. Specifies an array

array3 Optional. Specifies an array

Return Value: Returns TRUE on success or FALSE on failure

```
<?php
	$array1 = array(50,90,20,70,10,80);
	$array2 = array("fifty","ninety","twenty","seventy","ten","eighty");
	array_multisort($array1,$array2);
//array_multisort($array1,SORT_DESC ,$array2);
	foreach ($array1 as $key => $value) {
		echo "$key - <strong>$value</strong><br />";
	}
	echo "<hr />";
	foreach ($array2 as $key => $value) {
		echo "$key - <strong>$value</strong><br />";
	}
?>
```

ARRAY_POP ( )

Pop the last element off the end of an array. The array_pop() array function in PHP will remove the last element in an array and the result is the adjusted array with the last element removed. You can easily access the last value removed from array as well as the resulting array, as shown below in the code example.

array_pop(array)

array Required. Specifies an array

Return Value: Returns the last value of array. If array is empty, or is not an array, NULL will be returned.

```
<?php
	$targetArray = array("Sara","Cindy","Julie","Megan");
	array_pop($targetArray);
	foreach ($targetArray as $key => $value)
		echo "$key - <strong>$value</strong><br />";
?>
```

ARRAY_PUSH()

The array_push() function inserts one or more elements to the end of an array.You can add one value, or as many as you like.Even if your array has string keys, your added elements will always have numeric keys (See example below).

array_push(array,value1,value2...)

array Required. Specifies an array
value1 Required. Specifies the value to add
value2 Optional. Specifies the value to add
Return Value: Returns the new number of elements in the array

```
<?php
      $b=array("c"=>"Cherry","b"=>"Strawberry");
      array_push($b, "Orange", "Guava");
      print_r($b);
?>
```

COUNT ( )

Count the number of elements in an array

The count() array function in PHP will count the number of elements in an array. Sometimes you will have to see how many elements are inside of an array to perform certain evaluations in your scripts of you deal with arrays.

count(array,mode);

array Required. Specifies the array
mode Optional. Specifies the mode. Possible values:
0 - Default. Does not count all elements of multidimensional arrays
1 - Counts the array recursively (counts all the elements of multidimensional arrays)

```
<?php
      $a[0] = 1;
      $a[1] = 3;
      $a[2] = 5;
      $result = count($a);
      print($result);
?>
```

CURRENT ( )

Return the value of the current element in an array. The current () array function in PHP will return the value of the current element where the internal pointer is in an array. By default all arrays point to their first element unless the internal pointer is moved by the script.Tip: This function does not move the arrays internal pointer.

current(array)

array Required. Specifies the array to use

```
<?php
      $transport = array('foot', 'bike', 'car', 'plane');
      $mode = current($transport); // $mode = 'foot';
      $mode = next($transport);    // $mode = 'bike';
```

```
      $mode = current($transport); // $mode = 'bike';
      $mode = prev($transport);    // $mode = 'foot';
      $mode = end($transport);     // $mode = 'plane';
      $mode = current($transport); // $mode = 'plane';
?>
```

EACH ( )

Get current array element and advance the internal pointer. The each() array function in PHP will return the current array element and then advance the internal pointer in the array to the next element. This function will return a key value pair.

each(array)

array Required. Specifies the array to use

Return Value: Returns the current element key and value. This element key and value is returned in an array with four elements. Two elements (1 and Value) for the element value, and two elements (0 and Key) for the element key. This function returns FALSE if there are no more array elements

```
<?php
      $transport = array('foot', 'bike', 'car', 'plane');
      while(list($key,$val) = each($transport))
      {
            echo "The Key is:$key "; echo "The Value is:$val<br>";
      }
?>
```

END ( )

Get the value of the last element in an array

The end() array function in PHP will return the value of the last element in an array.

end(array)

array Required. Specifies the array to use

Return Value: Returns the value of the last element in the array on success, or FALSE if the array is empty

```
<?php
      $transport = array('foot', 'bike', 'car', 'plane');
      $mode = current($transport); // $mode = 'foot';
      $mode = next($transport);    // $mode = 'bike';
      $mode = current($transport); // $mode = 'bike';
      $mode = prev($transport);    // $mode = 'foot';
      $mode = end($transport);     // $mode = 'plane';
      $mode = current($transport); // $mode = 'plane';
?>
```

IN_ARRAY ( )

Look in an array for a value match. The in_array() array function in PHP is used when we wish to look inside of an array to see if a certain value exists. THis function will return either "true" or "false" when it runs.

in_array(search,array,type)

search Required. Specifies the what to search for

array Required. Specifies the array to search

type Optional. If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

Return Value: Returns TRUE if the value is found in the array, or FALSE otherwise

```
<?php
      $os = array("Android", "Windows", "Unix", "Linux");
      if (in_array("Unix", $os))
            echo "Unix found<br>";
      $a = array( array('p', 'r'), array('p', 'h'), 'o' );
      if ( in_array(array('p', 'h'), $a ))
            echo "p,h found";
?>
```

LIST ( )

List values from array elements placing them into variables. The list() array function in PHP will list values from array elements placing them into local variables. It will access the array elements and assign the values to your local variables from the values in the array elements.

list(var1,var2...)

var1 Required. The first variable to assign a value to

var2,... Optional. More variables to assign values to

Return Value: Returns the assigned array

```
<?php
      $w3r1_array = array("php","javascript","asp");
      list($x, $y, $z) = $w3r1_array;
      echo "We have covered $x,  $y and $z.";
      $w3r2_array = array("php","javascript","asp");
      list($x, , $z) = $w3r2_array;
      echo "We have covered $x and $z and so many other topics";
?>
```

NEXT ( )

Move the internal pointer of an array forward by one. The next() array function in PHP will move the internal pointer of an array forward by one. This function has opposite functionality to the prev() function.

next(array)

array Required. Specifies the array to use

Return Value: Returns the value of the next element in the array on success, or FALSE if there are no more elements

```
<?php
      $transport = array('foot', 'bike', 'car', 'plane');
      $mode = current($transport); // $mode = 'foot';
      $mode = next($transport);    // $mode = 'bike';
```

```
      $mode = current($transport); // $mode = 'bike';
      $mode = prev($transport);    // $mode = 'foot';
      $mode = end($transport);     // $mode = 'plane';
      $mode = current($transport); // $mode = 'plane';
?>
```

PREV ( )

Move the internal pointer of an array backward by one. The prev() array function in PHP will move the internal pointer of an array backward by one. This function has opposite functionality to the next() function

prev(array)

array Required. Specifies the array to use

Return Value: Returns the value of the previous element in the array on success, or FALSE if there are no more elements

```
<?php
      $transport = array('foot', 'bike', 'car', 'plane');
      $mode = current($transport); // $mode = 'foot';
      $mode = next($transport);    // $mode = 'bike';
      $mode = current($transport); // $mode = 'bike';
      $mode = prev($transport);    // $mode = 'foot';
      $mode = end($transport);     // $mode = 'plane';
      $mode = current($transport); // $mode = 'plane';
?>
```

RSORT ( )

Sort an array in reverse or descending order. The rsort() array function in PHP will sort an array in reverse or descending order. This function behaves opposite the sort() function.

rsort(array,sortingtype);

array Required. Specifies the array to sort

sortingtype Optional. Specifies how to compare the array elements/items. Possible values:

0 = SORT_REGULAR - Default. Compare items normally (don't change types)

1 = SORT_NUMERIC - Compare items numerically

2 = SORT_STRING - Compare items as strings

Return Value: TRUE on success. FALSE on failure

```
<?php
      $subject = array("Science",  "English", "Math", "History");
      rsort($subject);
      foreach ($subject as $key => $val)
            echo "subject[ " . $key . "] = " . $val . "<br />";
?>
```

ASORT()

The asort() function sorts an associative array in ascending order, according to the value. Use the ksort() function to sort an associative array in ascending order, according to the key.

asort(array)
array Required. Specifies the array to sort
Return Value: TRUE on success. FALSE on failure

```
<?php
      $subject = array("Science",  "English", "Math", "History");
      asort($subject);
      foreach ($subject as $key => $val)
      {
      echo "subject[ " . $key . "] = " . $val . "<br />";
      }
?>
```

ARSORT()

The arsort() function sorts an associative array in descending order, according to the value. Use the krsort() function to sort an associative array in descending order, according to the key.

arsort(array);
array Required. Specifies the array to sort
Return Value: TRUE on success. FALSE on failure

```
<?php
      $subject = array("Science",  "English", "Math", "History");
      arsort($subject);
      foreach ($subject as $key => $val)
      {
            echo "subject[ " . $key . "] = " . $val . "<br />";
      }
?>
```

SORT ( )

Sort array by values into ascending order from lowest to highest. The sort() array function in PHP will sort an array by its values from lowest to highest. It will sort letters and numbers in an ascending fashion.

sort(array,sortingtype);
array Required. Specifies the array to sort
sortingtypeOptional. Specifies how to compare the array elements/items. Possible values:
0 = SORT_REGULAR - Default. Compare items normally (don't change types)
1 = SORT_NUMERIC - Compare items numerically
2 = SORT_STRING - Compare items as strings

```
<?php
      $subject = array("Science",  "English", "Math", "History");
      sort($subject);
      foreach ($subject as $key => $val)
            echo "subject[ " . $key . "] = " . $val . "<br />";
?>
```

DATE FUNCTIONS

CHECKDATE()

This function checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

checkdate (month, day, year);

month Required. The month is between 1 and 12 inclusive.

day Required. The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year Required. The year is between 1 and 32767 inclusive.

Returns TRUE if the date given is valid; otherwise returns FALSE.

```
<?php
      $date = '2011-12-25';
      list($y, $m, $d) = explode('-', $date);
      if(checkdate($m, $d, $y)){
            echo "Valid date";
      }
      else{
            echo "Invalid date";
?>
```

DATE()

Returns a string formatted according to the given format string using the given integer timestamp or the current time if no timestamp is given. In other words, timestamp is optional and defaults to the value of time().

date (string $format [, int $timestamp]);

format Required. Specifies how to return the result:

timestamp Optional. This is an integer Unix timestamp that defaults to the current local time if a timestamp is not given. In other words, it defaults to the value of time().

Character	Description	Example returned values
a / A	Ante meridiem and Post meridiem	am or pm / AM or PM
d	Day of the month, 2 digits with leading zeros	01 to 31
D	A textual representation of a day, three letters	Mon through Sun
F	A full textual representation of a month, such as January or March	January through December
g	12-hour format of an hour without leading zeros	1 through 12
G	24-hour format of an hour without leading zeros	0 through 23
h	12-hour format of an hour with leading zeros	01 through 12
H	24-hour format of an hour with leading zeros	00 through 23
I	Minutes with leading zeros	00 to 59
J	Day of the month without leading zeros	1 to 31
l(small 'L')	A full textual representation of the day of the week	Sunday through Saturday

m	Numeric representation of a month, with leading zeros	01 through 12
M	A short textual representation of a month, three letters	Jan through Dec
N	Numeric representation of a month, without leading zeros	1 through 12
s	Seconds, with leading zeros	00 through 59
S	English ordinal suffix for the day of the month, 2 characters	st, nd, rd or th. Works well with j
T	Number of days in the given month	28 through 31
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	See also time()
W	Numeric representation of the day of the week	0 (for Sunday) through 6 (for Saturday)
Y	A full numeric representation of a year, 4 digits	Examples: 1999 or 2003
y	A two digit representation of a year	Examples: 99 or 03
Z	The day of the year (starting from 0)	0 through 365

```
<?php
      // Prints something like: Monday
      echo date("l");
      echo "<br />";
      // Prints something like: Monday 15th of August 2005 03:12:46
PM
      echo date('l dS \of F Y h:i:s A');
      echo "<br />";
      // Prints: July 1, 2000 is on a Saturday
      echo "July 1, 2000 is on a " . date("l",mktime(0, 0, 0, 7, 1,
2000));
      echo "<br />";
?>
```

DATE_ADD()

The date_add() function adds some days, months, years, hours, minutes, and seconds to a date.

date_add(object,interval);

object Required. Specifies a DateTime object returned by date_create()

interval Required. Specifies a DateInterval object

Return Value: Returns a DateTime object on success. FALSE on failure

```
<?php
      $date=date_create("2013-03-15");
      date_add($date,date_interval_create_from_date_string("40
days"));
      echo date_format($date,"Y-m-d");
?>
```

DATE_CREATE()

The date_format() function returns a date formatted according to the specified format.

date_create(time,timezone);

time Optional. Specifies a date/time string. NULL indicates the current time

timezone Optional. Specifies the timezone of time. Default is the current timezone.

Return Value: Returns a new DateTime object on success. FALSE on failure

```
<?php
      $date=date_create("2013-03-15                              23:40:00",
timezone_open("Asia/Kolkata"));
      echo date_format($date,"Y/m/d H:iP");
?>
```

DATE_FORMAT()

The date_format() function returns a date formatted according to the specified format.

date_format(object,format);

object Required. Specifies a DateTime object returned by date_create()

format Required. Specifies the format for the date

Return Value: Returns the formatted date as a string. FALSE on failure

```
<?php
      $date=date_create("2013-03-15
23:40:00",timezone_open("Asia/Kolkata"));
      echo date_format($date,"Y/m/d H:iP");
?>
```

GETDATE()

The getdate() function returns date/time information of a timestamp or the current local date/time.

getdate(timestamp);

timestamp Optional. Specifies an integer Unix timestamp. Default is the current local time (time())

Return Value: Returns an associative array with information related to the timestamp:

[seconds] - seconds
[minutes] - minutes
[hours] - hours
[mday] - day of the month
[wday] - day of the week
[mon] - month
[year] - year
[yday] - day of the year
[weekday] - name of the weekday

[month] - name of the month
[0] - seconds since Unix Epoch

```
<?php
      $today = getdate();
      print_r($today);
?>
```

GETTIMEOFDAY()

The gettimeofday() function returns the current time.

gettimeofday();

Return Value: Returns an associative array by default, with the following array keys:

[sec] - seconds since the Unix Epoch
[usec] - microseconds
[minuteswest] - minutes west of Greenwich
[dsttime] - type of dst correction

```
<?php
    echo gettimeofday(true)."<br /><br />";
    print_r(gettimeofday());
?>
```

GMDATE()

The gmdate() function formats a GMT/UTC date and time, and returns the formatted date string.

gmdate(format,timestamp);

format Required. Specifies the format of the outputted date string.

timestamp Optional. Specifies an integer Unix timestamp. Default is the current local time (time())

```
<?php
    echo("Result with date():<br />");
    echo(date("l") . "<br />");
    echo(date("l dS \of F Y h:i:s A") . "<br />");
    echo("Result with gmdate():<br />");
    echo(gmdate("l")."<br />");
    echo(gmdate("l dS \of F Y h:i:s A")."<br />");
?>
```

LOCALTIME()

The localtime() function returns the local time.

localtime(timestamp,is_assoc);

timestamp Optional. Specifies a Unix timestamp that defaults to the current local time, time(), if no timestamp is specified

is_assoc Optional. Specifies whether to return an associative or indexed array. FALSE = the array returned is an indexed array. TRUE = the array returned is an associative array. FALSE is default.

Return Value: The keys of the associative array are:

[tm_sec] - seconds
[tm_min] - minutes
[tm_hour] - hour
[tm_mday] - day of the month
[tm_mon] - month of the year (January=0)
[tm_year] - Years since 1900
[tm_wday] - Day of the week (Sunday=0)
[tm_yday] - Day of the year
[tm_isdst] - Is daylight savings time in effect

```
<?php
      echo "<pre>";
      print_r(localtime(time(),true));
      echo '<br />';
      print_r(localtime());
      echo "</pre>";
?>
```

MKTIME()

The mktime() function returns the Unix timestamp for a date.

mktime(hour,minute,second,month,day,year);

hour Optional. Specifies the hour
minute Optional. Specifies the minute
second Optional. Specifies the second
month Optional. Specifies the month
day Optional. Specifies the day
year Optional. Specifies the year

```
<?php
      // Prints: July 1, 2000 is on a Saturday
      echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1,
2000));
      echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
      echo date("M-d-Y", mktime(0, 0, 0, 13, 1, 1997));
      echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
      echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
?>
```

STRFTIME()

The strftime() function formats a local time and/or date according to locale settings.

strftime(format,timestamp);

format Required. Specifies how to return the result:
%a - abbreviated weekday name
%A - full weekday name
%b - abbreviated month name
%B - full month name
%d - day of the month (01 to 31)

%e – day of the month (1 to 31)
%H – hour, using a 24-hour clock (00 to 23)
%I – hour, using a 12-hour clock (01 to 12)
%j – day of the year (001 to 366)
%m – month (01 to 12)
%M – minute
%p – either am or pm according to the given time value
%r – time in a.m. and p.m. notation
%R – time in 24 hour notation
%S – second
%w – day of the week as a decimal, Sunday=0
%y – year without a century (range 00 to 99)
%Y – year including the century

timestamp Optional. Specifies a Unix timestamp that represents the date and/or time to be formatted. Default is the current local time (time())

```
<?php
      echo(strftime("%B %d %Y, %X
%Z",mktime(20,0,0,12,31,98))."<br>");
      setlocale(LC_ALL,"hu_HU.UTF8");
      echo(strftime("%Y. %B %d. %A. %X %Z"));
?>
```

STRPTIME()

The strptime() function parses a time/date generated with strftime().This function is not implemented on Windows platforms!

strptime(date,format);

date Required. The string to parse (e.g. returned from strftime())

format Required. Specifies the format used in the date [format character same as strftime]

```
<?php
      $format="%d/%m/%Y %H:%M:%S";
      $strf=strftime($format);
      echo("$strf");
      print_r(strptime($strf,$format));
?>
```

STRTOTIME()

The strtotime() function parses an English textual datetime into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).If the year is specified in a two-digit format, values between 0-69 are mapped to 2000-2069 and values between 70-100 are mapped to 1970-2000. Be aware of dates in the m/d/y or d-m-y formats; if the separator is a slash (/), then the American m/d/y is assumed. If the separator is a dash (-) or a dot (.), then the European d-m-y format is assumed. To avoid potential errors, you should YYYY-MM-DD dates or date_create_from_format() when possible.

strtotime(time,now);

time Required. Specifies a date/time string
now Optional. Specifies the timestamp used as a base for the calculation of relative dates
Return Value: Returns a timestamp on success. FALSE on failure

```
<?php
      $timestamp = strtotime('April 1, 2009');
      $timestamp = strtotime('next Saturday');
      $timestamp = strtotime('07/05/2008');
      $timestamp = strtotime('April 1, 2009 + 3 days');
      $timestamp = strtotime('07/05/2008 - 1 month');
?>
```

TIME()

The time() function returns the current time in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

time();

Return Value: Returns an integer containing the current time as a Unix timestamp

```
<?php
      $t=time();
      echo($t . "<br>");
      echo(date("Y-m-d",$t));
?>
```

FILE SYSTEM FUNCTIONS

COPY ( )

Create a copy of a file. The copy() file system function in PHP will make a copy of a file that you specify on the server, It will place the new copy where you specify with a name you give it. This function traditionally takes two parameters with the first being the file to copy, and the second parameter is the path and file name of the copy you are creating.

copy(file,to_file)

file Required. Specifies the file to copy
to_file Required. Specifies the file to copy to

```
<?php
      $file = 'example.txt';
      $newfile = 'example.txt.bak';
      if (!copy($file, $newfile)) {
            echo "failed to copy $file...\n";
      }
?>
```

FGETC ( )

Return a single character from an open file. The fgetc() file system function in PHP will return a single character read from a file that is opened using fopen() or fsockopen(). This function is slow and should not be used on large files. If you need to read one character at a time from a large file,

use fgets() to read data one line at a time and then process the line one character at a time with fgetc().

fgetc(file)

file Required. Specifies the file to check

```
<?php
      $handle = fopen("file.txt", "r") or die("can't open file");
      while ($c = fgetc($handle, 4096)) {
      echo $c."<br />";
      }
      fclose($handle);
?>
```

FGETS ( )

Returns a line from a target file handler. The fgets() filesystem function in PHP will return a line from your target file handler. This function can take up to two parameters. The first parameter is your file handler. The second parameter is the character length you wish to read into the line. If no second parameter is supplied it will read until the end of the line.

fgets(file,length)

file Required. Specifies the file to read from

length Optional. Specifies the number of bytes to read. Default is 1024 bytes.

```
<?php
      $handle = fopen("file.txt", "r") or die("can't open file");
      while ($line = fgets($handle, 4096)) {
            echo $line."<br />";
      }
      fclose($handle);
?>
```

FILE ( )

Places file contents into an array. The file() filesystem function in PHP will read the contents of a file into an array.

file(path)

path Required. Specifies the file to read

```
<?php
      $lines = file('file.txt');
      foreach ($lines as $line_num => $line)
            print "<font color=red>Line #{$line_num}</font> : " .
$line . "<br />\n";
?>
```

FILE_GET_CONTENTS ( )

Read entire file contents and place in a string. The file_get_contents() filesystem function in PHP will read the contents of a target file and set the contents into a string.

file_get_contents(path,include_path,context,start,max_length)

path Required. Specifies the file to read

include_path Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.
start Optional. Specifies where in the file to start reading. This parameter was added in PHP 5.1
max_length Optional. Specifies how many bytes to read. This parameter was added in PHP 5.1

```
<?php
      echo file_get_contents("file.txt");
?>
```

FCLOSE()

The fclose() function closes an open file. This function returns TRUE on success or FALSE on failure.

fclose(file)

file Required. Specifies the file to close

```
<?php
      $handle = fopen('somefile.txt', 'r');
      fclose($handle);
?>
```

FILE_EXISTS()

The file_exists() function checks whether or not a file or directory exists. This function returns TRUE if the file or directory exists, otherwise it returns FALSE.

file_exists(path)

path Required. Specifies the path to check

```
<?php
      $filename = 'somefile.txt';
      if (file_exists($filename)) {
            echo "The file $filename exists";
      } else {
            echo "The file $filename does not exist";
      }
?>
```

FILE_PUT_CONTENTS ( )

Write string data into a file. The file_put_contents() filesystem function in PHP will write string data to a file. It works the same as if you were to use fopen(), fwrite(), and fclose() together to write data to a file. The file will be created if the output file does not exist yet, and if it already exists it will be overwritten.

file_put_contents(file,data,mode)

file Required. Specifies the file to write to. If the file does not exist, this function will create one
data Required. The data to write to the file. Can be a string, an array or a data stream

mode Optional. Specifies how to open/write to the file. Possible values: FILE_USE_INCLUDE_PATH, FILE_APPEND

```
<?php
      $file = 'people.txt';
      // Open the file to get existing content
      $current = file_get_contents($file);
      // Append a new person to the file
      $current .= "John Smith\n";
      // Write the contents back to the file
      file_put_contents($file, $current);
      $file = 'people.txt';
      // The new person to add to the file
      $person = "John Smith\n";
      // Write the contents to the file,
      // using the FILE_APPEND flag to append the content to the end
of the file
      // and the LOCK_EX flag to prevent anyone else writing to the
file at the same time
      file_put_contents($file, $person, FILE_APPEND | LOCK_EX);
?>
```

FILESIZE ( )

Returns the byte size of a file . The filesize() filesystem function in PHP will give you a way to read the size of any target file. It will return a number representing the file size in bytes.**filesize(filename)**

filename Required. Specifies the file to check

```
<?php
      $myFile = "testFile.txt";
      $fh = fopen($myFile, 'r');
      $theData = fread($fh, 5);
      fclose($fh);
      echo $theData;

      $size = filesize($myFile);
      $fh = fopen($myFile, 'r');
      $theData = fread($fh, $size);
      fclose($fh);
      echo $theData;
?>
```

FOPEN ( )

Opens a file or a URL. The fopen() filesystem function in PHP will open a file or URL, at which point you can read it, write to it, append to it, and perform similar actions to it once opened.

fopen(filename,mode)

filename Required. Specifies the file or URL to open

mode Required. Specifies the type of access you require to the file/stream.

Possible values:

"r" (Read only. Starts at the beginning of the file)

"a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist)

"w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist)

```
<?php
	$ourFileName = "testFile.txt";
	$ourFileHandle = fopen($ourFileName, 'w') or die("can't open file");
	fclose($ourFileHandle);
?>
```

FPUTCSV()

The fputcsv() function formats a line as CSV and writes it to an open file.This function returns the length of the written string, or FALSE on failure.

fputcsv(file,fields,seperator,enclosure)

file Required. Specifies the open file to write to

fields Required. Specifies which array to get the data from

separator Optional. A character that specifies the field separator. Default is comma (,)

enclosure Optional. A character that specifies the field enclosure character. Default is "

```
<?php
	$list = array (
		array('aaa', 'bbb', 'ccc', 'dddd'),
		array('123', '456', '789'),
		array('"aaa"', '"bbb"')
	);
	$fp = fopen('file.csv', 'w');
	foreach ($list as $fields) {
		fputcsv($fp, $fields);
	}
	fclose($fp);
?>
```

FPUTS()

The fputs() writes to an open file.The function will stop at the end of the file or when it reaches the specified length, whichever comes first.This function returns the number of bytes written on success, or FALSE on failure.The fputs() function is an alias of the fwrite() function.

fputs(file,string,length)

file Required. Specifies the open file to write to

string Required. Specifies the string to write to the open file
length Optional. Specifies the maximum number of bytes to write

```
<?php
      $file = fopen("test.txt","w");
      echo fputs($file,"Hello World. Testing!");
      fclose($file);
?>
```

FREAD ( )

Reads and returns file data up to a specified byte length. The fread() filesystem function in PHP will read a specified byte length into a target file. You can choose to set it to read the full byte amount of a file, or any byte amount you specify. The first parameter of the function is the file handler(fopen() variable), and the second parameter is the byte length you wish to read into the file to access data.

fread(file,length)

file Required. Specifies the open file to read from
length Required. Specifies the maximum number of bytes to read

```
<?php
      $myFile = "testFile.txt";
      $fh = fopen($myFile, 'r');
      $theData = fread($fh, 5);
      fclose($fh);
      echo $theData;
?>
```

FSEEK()

The fseek() function seeks in an open file.This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.This function returns 0 on success, or -1 on failure. Seeking past EOF will not generate an error.

fseek(file,offset)

file Required. Specifies the open file to seek in
offset Required. Specifies the new position (measured in bytes from the beginning of the file)

```
<?php
      $fp = fopen('file.txt', 'r');
      $data = fread($fp, 20);
      echo $data;
      echo "<br />";
      fseek($fp, 10);
      $data = fread($fp, 20);
      echo $data
?>
```

FTELL()

The ftell() function returns the current position in an open file.Returns the current file pointer position, or FALSE on failure.

ftell(file)

file Required. Specifies the open file to check

```
<?php
      $fp = fopen("/etc/passwd", "r");
      $data = fgets($fp, 12);
      echo ftell($fp);
      fclose($fp);
?>
```

FWRITE ( )

Write to a file. The fwrite() file system function in PHP will write to a file. It can take up to three parameters and the third parameter is optional. The first parameter is the fileHandler created by using fopen(). The second parameter is the string data to be written. The third optional parameter is a length you can specify to write only a certain amount of the string to the file.

fwrite() is configured by your file handler. Meaning that if you change the "w" to an "a" in the fopen() function, it will append the new string data to any existing string data when you use the fwrite() function to add the string data. If you write to a file twice within the same fileHandler session it will append by default and not overwrite the data placed by the first write.

It is important to note that any time you write to a file, the target file will be created on server if it does not yet exist. Which makes it operate as a file creation method like the touch() function if you wanted to use it in that way.

When fwrite() executes it will return the amount of bytes written to the file if you wish to access that number by placing it into a variable.

fwrite(file,string,length)

file Required. Specifies the open file to write to
string Required. Specifies the string to write to the open file
length Optional. Specifies the maximum number of bytes to write

```
<?php
      $myFile = "testFile.txt";
      $fh = fopen($myFile, 'w') or die("can't open file");
      $stringData = "Hello world\n";
      fwrite($fh, $stringData);
      $stringData = "Bharmal\n";
      fwrite($fh, $stringData);
      fclose($fh);
?>
```

IS_READABLE()

The is_readable() function checks whether the specified file is readable.This function returns TRUE if the file is readable.

is_readable(file)

file Required. Specifies the file to check

```
<?php
      $filename = 'test.txt';
      if (is_readable($filename)) {
            echo 'The file is readable';
      } else {
            echo 'The file is not readable';
      }
?>
```

IS_WRITABLE()

The is_writable() function checks whether the specified file is writeable.This function returns TRUE if the file is writeable.

is_writable(file)

file Required. Specifies the file to check

```
<?php
      $file_name='test.txt';
      if (is_writable($file_name))
      {
            echo "$file_name is Writable.<br>" ;
            $myFile = "testFile.txt";
            $fh = fopen($myFile, 'w') or die("can't open file");
            $stringData = "Bobby Bopper\n";
            fclose($fh);
      }
      else
      {
      echo "$file_name is not Writable.<br>" ;
      }
?>
```

MOVE_UPLOADED_FILE ( )

Move an uploaded file. The move_uploaded_file() filesystem function in PHP will move an uploaded file, usually from the PHP temp folder to its permanent location on your server. This function takes two parameters. Parameter one is the file in its temporary location in the PHP temp folder, and the second parameter is the new path and the file name. "Path" means which folder you want to place the image into.

move_uploaded_file(file,newloc)

file Required. Specifies the file to be moved

newloc Required. Specifies the new location for the file

```
<!-- The data encoding type, enctype, MUST be specified as below -->
<form          enctype="multipart/form-data"          action="<?php
$_SERVER['PHP_SELF'] ?>" method="POST">
<!-- MAX_FILE_SIZE must precede the file input field -->
<input type="hidden" name="MAX_FILE_SIZE" value="30000" />
<!-- Name of input element determines name in $_FILES array -->
```

```
   Send this file: <input name="userfile" type="file" />
<input type="submit" value="Send File" />
</form>
<?php
     $uploaddir    =    'c:\\';    $uploadfile    =    $uploaddir    .
basename($_FILES['userfile']['name']);
     echo '<pre>';
     if         (move_uploaded_file($_FILES['userfile']['tmp_name'],
$uploadfile))
           echo "File is valid, and was successfully uploaded.\n";
     else
           echo "Possible file upload attack!\n";
     echo 'Here is some more debugging info:';
     print_r($_FILES);
     print "</pre>";
?>
```

RENAME ( )

Rename a file or directory folder. The rename() filesystem function in PHP will rename a file or folder. This function usually uses two parameters. The first parameter we feed it is the target file or directory to be renamed. The second parameter is the new name the target is to have.The rename() function renames a file or directory.This function returns TRUE on success, or FALSE on failure.

rename(oldname,newname)

oldname Required. Specifies the file or directory to be renamed

newname Required. Specifies the new name of the file or directory

```
<?php
     $target = "information.txt";
     $newName = "newinfo.txt";
     $renameResult = rename($target, $newName);
     // Evaluate the value returned from the function if needed
     if ($renameResult == true) {
           echo $target . " is now named " . $newName;
     } else {
            echo "Could not rename that file";
     }
?>
```

REWIND()

The rewind() function "rewinds" the position of the file pointer to the beginning of the file.This function returns TRUE on success, or FALSE on failure.

rewind(file)

file Required. Specifies the open file

```
<?php
     $fp = fopen('file.txt', 'r');
     $data = fread($fp, 20);
     echo $data;echo "<br />";
```

```
      rewind($fp);
      $data = fread($fp, 20);
      echo $data
?>
```

UNLINK()

The unlink() function deletes a file.This function returns TRUE on success, or FALSE on failure.

unlink(filename)

filename Required. Specifies the file to delete

```
<?php
      $myFile = "testFile.txt";
      $fh = fopen($myFile, 'w') or die("can't open file");
      $data = fread($fh,100);
      echo $data;
      fclose($fh);unlink($myFile);
      $myFile = "testFile.txt";
      $fh = fopen($myFile, 'w') or die("can't open file");
      fclose($fh);
      $data = fread($fh,100);
      echo $data;
?>
```

MATHS FUNCTIONS

ABS()

The abs() function returns the absolute value of a number.

abs(x)

x Required. A number. If the number is of type float, the return type is also float, otherwise it is integer

```
<?php
      echo(abs(-6.2));
      print "<br>";
      echo(abs(7));
      print "<br>";
      echo(abs(-3));
?>
```

ACOS()

The acos() function returns the arccosine of a number as a numeric value value between 0 and PI radians.

acos(x)

x Required. Must be a numeric value in the range -1 to 1

```
<?php
      echo(acos(0.64) . "<br />");
      echo(acos(0) . "<br />");
?>
```

ASIN()

The asin() function returns the arcsine of a number as a numeric value between-PI/2 and PI/2 radians.

asin(x)

x Required. Must be a numeric value in the range -1 to 1

```
<?php
      echo(asin(0.64) . "<br />");
      echo(asin(0) . "<br />");
      echo(asin(-1) . "<br />");
?>
```

ATAN() AND ATAN2()

The atan() function returns the arctangent of a number as a numeric value between -PI/2 and PI/2 radians.The atan2() function returns the angle theta of an (x,y) point as a numeric value between -PI and PI radians.

atan(x)

atan2(x,y)

x Required. A number

y Required. A number

```
<?php
      echo(atan(0.50) . "<br />");
      echo(atan(-0.50) . "<br />");
?>
```

BASE_CONVERT()

The base_convert() function converts a number from one base to another.

base_convert(number,frombase,tobase)

number Required. Original value

frombase Required. Original base of number. Frombase has to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, with a meaning 10, b meaning 11 and z meaning 35.

tobase Required. The base to convert to. Tobase has to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, with a meaning 10, b meaning 11 and z meaning 35

```
<?php
      $decimal = "935648124576358";
      $hexa_decimal = base_convert($decimal, 10, 16);
      echo "$decimal decimal = $hexa_decimal hexadecimal.<br/>";

      $oct = "01234567";
      $hexa_decimal = base_convert($oct, 8, 16);
      echo "$oct octal = $hexa_decimal hexadecimal.<br/>";
```

```
      $oct = "01234567";
      $decimal = base_convert($oct, 8, 10);
      echo "$oct octal = $decimal decimal.<br/>";

      $decimal = "342391";
      $oct = base_convert($decimal, 10, 8);
      echo "$decimal decimal = $oct octal.<br/>";

      $oct = "342391";
      $binary = base_convert($oct, 8, 2);
      echo "$oct octal = $binary binary.";
?>
```

BINDEC()

The bindec() function converts a binary number to a decimal number.

bindec(binary_number)

binary_number Required. Specifies the binary number to convert

```
<?php
      echo bindec('110011') . "\n";
      echo bindec('000110011') . "\n";
      echo bindec('111');
?>
```

CEIL()

The ceil() function returns the value of a number rounded UPWARDS to the nearest integer.

ceil(x)

x Required. A number

```
<?php
      $someval = 4.9;
      echo $ceiled = ceil($someval);
      echo $floored = floor($someval);
      $someval = -4.9;
      echo $ceiled = ceil($someval);
      echo $floored = floor($someval);
?>
```

COS()

The cos() function returns the cosine of a number.

cos(x)

x Required. A number

```
<?php
      // Degree
      $degree = 360;
      // Convertion
      print cos(deg2rad($degree));
?>
```

DECBIN()

The decbin() function converts a decimal number to a binary number.

decbin(dec_number)

dec_number Required. Specifies the decimal number to convert

```
<?php
      echo decbin(10) . "<br />";
      echo decbin(5) . "<br />";
      echo decbin(3);
?>
```

DECHEX()

The dechex() function converts a decimal number to a hexadecimal number.

dechex(dec_number)

dec_number Required. Specifies the decimal number to convert

```
<?php
      echo dechex("15") . "<br />";
      echo dechex("10") . "<br />";
?>
```

DEG2RAD()

The deg2rad() function converts a degree to its radian number.

deg2rad(degree_number)

degree_number Required. Specifies the degree to convert

```
<?php
      echo deg2rad(45); // 0.785398163397
      //M_PI_4 = pi/4
      var_dump(deg2rad(45) === M_PI_4); // bool(true)
?>
```

FLOOR()

The floor() function returns the value of a number rounded DOWNWARDS to the nearest integer.

floor(x)

x Required. A number

```
<?php
    $someval = 4.9;
    echo $ceiled = ceil($someval);
    echo $floored = floor($someval);
      $someval = -4.9;
    echo $ceiled = ceil($someval);
    echo $floored = floor($someval);
?>
```

FMOD()

The fmod() function divides x by y and returns the remainder (modulo) of the division.

fmod(x,y)

x Required. A number

y Required.

```
<?php
      $remainder = fmod(7, 2);
      echo "Remainder is :" . $remainder."<br />";
      echo "<br />";
      $remainder = fmod(7.5, 2.3);
      echo "Remainder is :" . $remainder;
?>
```

HEXDEC()

The hexdec() function converts a hexadecimal number to a decimal number.

hexdec(hex_number)

hex_number Required. Specifies the hexadecimal number to convert

```
<?php
      echo hexdec("1e") . "<br />";
      echo hexdec("a") . "<br />";
?>
```

IS_FINITE()

The is_finite() function returns true if the specified value is a finite number, otherwise it returns nothing. The value is finite if it is within the allowed range for a PHP float on this platform.

is_finite(x)

x Required. The value to check

```
<?php
      echo is_finite(3) . "<br />";
      echo is_finite(log(0)) . "<br />";
      echo is_finite(3000);
?>
```

IS_INFINITE()

The is_infinite() function returns true if the specified value is an infinite number, otherwise it returns nothing.The value is infinite if it is too big to fit into a PHP float on this platform.

is_infinite(x)

x Required. The value to check

```
<?php
      echo is_infinite(2) . "<br />";
      echo is_infinite(log(0)) . "<br />";
      echo is_infinite(2000);
?>
```

LOG()

The log() function returns the natural logarithm (base E) of a number.

log(x,base)

x Required. A number

base Optional. If the base parameter is specified, log() returns logbase x. Note: The base parameter became available in PHP 4.

```
<?php
      $returnValue = log(3);
      echo $returnValue;
?>
```

MAX()

The max() function returns the number with the highest value of two specified numbers.

max(x,y)

x Required. A number

y Required. A number

```
<?php
 echo max(2, 4, 6, 8) ;
 // Returns 2

 echo max (array(2, 14, 7)) ;
 //Returns 2

 echo max(0, 'about') ;
 //Returns 0

 echo max ('about', 0) ;
 // Returns about

 // Comparing two arrays
 // so in our example: 5 = 5, but 7 < 9
 $x = max(array(5, 7, 12), array(5, 9, 1)) ;
 //Returns array(5, 7, 12)

 // If both an array and non-array are given, the array is always
seen as the largest, and therefore not returned
 $x = max('about', array(2, 4, 6), 99) ;
 //Returns about, as it is seen as 0
?>
```

MIN()

The min() function returns the number with the lowest value of two specified numbers.

min(x,y)

x Required. A number

y Required. A number

```
<?php
 echo min(2, 4, 6, 8) ;
 // Returns 2

 echo min (array(2, 14, 7)) ;
 //Returns 2
```

```
 echo min(0, 'about') ;
 //Returns 0

 echo min ('about', 0) ;
 // Returns about

 // Comparing two arrays
 // so in our example: 5 = 5, but 7 < 9
 $x = max(array(5, 7, 12), array(5, 9, 1)) ;
 //Returns array(5, 7, 12)

 // If both an array and non-array are given, the array is always
seen as the largest, and therefore not returned
 $x = max('about', array(2, 4, 6), 99) ;
 //Returns about, as it is seen as 0
?>
```

POW()

The pow() function raises the first argument to the power of the second argument, and returns the result.

pow(x,y)

x Required. Specifies the number to be raised

y Required. The power to which to raise the number

```
<?php
      echo pow(7,3) . '<br>';     echo pow(2,3) . '<br>';
      echo pow(-2,3) . '<br>';    echo pow(-2,-3) . '<br>';
      echo pow(-2,-3.2) . '<br>';
?>
```

RAND()

The rand() function generates a random integer. If this function is called without parameters, it returns a random integer between 0 and RAND_MAX. If you want a random number between 10 and 100 (inclusive), use rand (10,100).

rand(min,max)

min,max Optional. Specifies the range the random number should lie within

Note: On some platforms (such as Windows) RAND_MAX is only 32768. So, if you require a range larger than 32768, you can specify min and max, or use the mt_rand() function instead. The mt_rand() function generates a better random value than this function!

```
<?php
      print rand() . "<br>";
      //generates and prints a random number
      print rand(10, 30);
      //generates and prints a random number between 10 and 30 (10
and 30 ARE included)
```

```
      print rand(1, 1000000);
      //generates and prints a random number between on and one
million
?>
```

ROUND()

The round() function rounds a number to the nearest integer.

round(x,prec)

x Required. The number to be round

prec Optional. The number of digits after the decimal point

```
<?php
    echo round(4.9); // 5
    echo round(4.5); // 5
    echo round(4.4999); // 4
    echo round(4.123456, 3); // 4.123
    echo round(4.12345, 4); // 4.1235
    echo round(1000 / 160); // 6
?>
```

SIN()

The sin() function returns the sine of a number.

sin(x)

x Required. A number

```
<?php
      echo sin(deg2rad(60));  //  0.866025403 ...
      echo sin(60);           // -0.304810621 ...
?>
```

SQRT()

The sqrt() function returns the square root of a number.

sqrt(x)

x Required. A number

```
<?php
      print sqrt (9);
      // this would return the value 3
      print sqrt (27.04);
      //this would return the value 5.2
?>
```

TAN()

The tan() function returns a number that represents the tangent of an angle.

tan(x)

x Required. A number

```
<?php
      echo(tan(0.50) . "<br />");
```

```
      echo(tan(-0.50) . "<br />");
?>
```

MISCELLANEOUS FUNCTIONS

CONSTANT()

The constant() function returns the value of a constant.This function also works with class constants.

constant(constant)

constant Required. Specifies the name of the constant to check

```
<?php
      define("MINSIZE", 50);
      echo MINSIZE;
      echo constant("MINSIZE"); // same output of previous line
?>
```

DEFINE()

The define () function defines a constant.Constants are much like variables, except for the following differences:

- A constant's value cannot be changed after it is set
- Constant names do not need a leading dollar sign ($)
- Constants can be accessed regardless of scope
- Constant values can only be strings and numbers

define(name,value,case_insensitive)

name Required. Specifies the name of the constant

value Required. Specifies the value of the constant

case_insensitive Optional. Specifies whether the constant name should be case-insensitive. Possible values:

TRUE - Case-insensitive

FALSE - Default. Case-sensitive

```
<?php
      define("GREETING1","Hello you! How are you today?");
      echo constant("GREETING2");
      define("GREETING2","Hello you! How are you today?",TRUE);
      echo constant("greeting2");
?>
```

DIE()

The die() function prints a message and exits the current script.This function is an alias of the exit() function.

die(message)

message Required. Specifies the message or status number to write before exiting the script. The status number will not be written to the output.

```
<?php
```

```
      for($i = 0; $i < 10; $i++){
            if ($i == 2){
                  die("\n Using die(), We are done");
            }
      }
?>
```

EXIT()

The exit() function prints a message and exits the current script.This function is an alias of the die() function.

exit(message)

message Required. Specifies the message or status number to write before exiting the script. The status number will not be written to the output.

```
<?php
      for($i = 0; $i < 10; $i++)
            if ($i == 2)
                  exit("\n Using exit(), We are done");
?>
```

HEADER()

The header() function sends a raw HTTP header to a client.It is important to notice that header() must be called before any actual output is sent (In PHP 4 and later, you can use output buffering to solve this problem):

header(string,replace)

string Required. Specifies the header string to send

replace Optional. Indicates whether the header should replace previous or add a second header. Default is TRUE (will replace). FALSE (allows multiple headers of the same type)

```
<?php
      header('Location: http://www.example.com/');
      echo "This page will not be displayed";
?>
```

INCLUDE()

include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.

include 'filename'

```
menu.php
<a href="http://www.tutorialspoint.com/index.htm">Home</a> -
```

```
<a href="http://www.tutorialspoint.com/ebxml">ebXML</a> -
<a href="http://www.tutorialspoint.com/ajax">AJAX</a> -
<a href="http://www.tutorialspoint.com/perl">PERL</a><br />
```

```
<html>
      <body>
            <?php include("menu.php"); ?>
            <p>This is an example to show how to include PHP
file!</p>
      </body>
</html>
```

REQUIRE()

The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the require() function generates a fatal error and halt the execution of the script.

So there is no difference in require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.

require 'filename'

```
menu.php
<a href="http://www.tutorialspoint.com/index.htm">Home</a> -
<a href="http://www.tutorialspoint.com/ebxml">ebXML</a> -
<a href="http://www.tutorialspoint.com/ajax">AJAX</a> -
<a href="http://www.tutorialspoint.com/perl">PERL</a><br />
```

```
<html>
      <body>
            <?php require("menu.php"); ?>
            <p>This is an example to show how to include PHP
file!</p>
      </body>
</html>
```

STRING FUNCTIONS

CHR ( )

Returns the character representation of an ASCII value

The chr() string function in PHP will return a character representation of an ASCII value you supply it with.

chr(ascii)

ascii Required. An ASCII value

Return Value: Returns the specified character

```
<?php
      $str = chr(73); echo($str);
```

```
?>
```

ECHO ( ) [PRINT ( )]

Output a string. The echo() string function provides a way for us to output or render data to the user, or to output the data to a receiving application(such as Flash). It has no return values and is not actually a function. To output more than one parameter within the language construct, do not encapsulate the parameters in parenthesis"( )". If only one parameter is being sent through the construct it can be encapsulated by parenthesis.

Your data can be enclosed in single quotes or double quotes, just be careful to use the proper variable render methods in each instance of both single and double quotes enclosures. You must also escape quote marks in your strings that match your quote mark enclosure method.The print() function works in a similar way but they are technically not exactly the same.

echo(strings)

strings Required. One or more strings to be sent to the output

Return Value: No value is returned

```
<?php
 echo "Hello";
 $variable = "Test String";
 echo $variable;
 echo "Multiple things " . $variable . " one line";
?>
```

EXPLODE ( )

Returns an array of substrings from a target string. The explode() string function will break a target string into an array of substrings by taking in 2 parameters. First being the delimiter(character or substring to make break points with in the target string). Second parameter you feed it is the target string. You can use custom unique delimiters when needed(shown at bottom of page).

explode(separator,string,limit)

separator Required. Specifies where to break the string

string Required. The string to split

limit Optional. Specifies the number of array elements to return.

Greater than 0 – Returns an array with a maximum of limit element(s)

Less than 0 – Returns an array except for the last –limit elements()

0 – Returns an array with one element

Return Value: Returns an array of strings

```
<?php
      $str="String as array values";
      print_r(explode(" ",$str));
?>
```

IMPLODE ( ) JOIN ( )

Both combine array elements into a string. The implode() and join() string functions in PHP currently perform the same operation so I am putting them together and we will demonstrate them using implode() only. They are both used to combine elements of an array into a string. join() works just like implode().

implode() (and jojn()) does the opposite of explode(). While implode() combines array elements to make a string, explode() takes a string and makes an array out of it. And very much like explode(), with implode() you get to choose what goes in between each part of the array as it becomes a string.

implode(separator,array)

separator Optional. Specifies what to put between the array elements. Default is "" (an empty string)

array Required. The array to join to a string

Return Value: Returns a string from elements of an array

```
<?php
      $arr = array ('Php','MySql','JQuery','Smarty');
      $space_separated = implode(" ", $arr);
      $comma_separated = implode(" , ", $arr);
      $slash_separated = implode(" / ", $arr);
      $dot_separated = implode(" . ", $arr);
      $hyphen_separated = implode(" - ", $arr);
      echo $space_separated.'<br>';
      echo $comma_separated.'<br>';
      echo $slash_separated.'<br>';
      echo $dot_separated.'<br>';
      echo $hyphen_separated;
?>
```

JOIN()

The join() function returns a string from the elements of an array.The join() function is an alias of the implode() function. The join() function accept its parameters in either order. However, for consistency with explode(), you should use the documented order of arguments.

The separator parameter of join() is optional. However, it is recommended to always use two parameters for backwards compatibility.

join(separator,array)

separator Optional. Specifies what to put between the array elements. Default is "" (an empty string)

array Required. The array to join to a string

Return Value: Returns a string from elements of an array

```
<?php
      $array_name=array('1','2','3','4','....');
      $join_string=join(",", $array_name);
      echo $join_string;
      echo '<br>';
      $array_name=array('A','B','C','D','E');
```

```
      $join_string=join("-", $array_name);
      echo $join_string;
?>
```

ITRIM ( )

Removes characters and whitespace from the front of a string

The Itrim() string function in PHP will remove whitespace and characters from the front of a string. You can pass up to 2 parameters through it. First parameter is the target string. The second parameter is optional and is any specified character (or substring) you want removed from the front of a string.

With only one parameter used this function will remove a space, tab, new line, carriage return, NUL-byte, and vertical tab characters from the beginning of a string.

The ltrim() function will remove whitespaces or other predefined characters from the left side of a string.

ltrim(string,charlist)

string Required. Specifies the string to check

charlist Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed:

- *"\0" - NULL*
- *"\t" - tab*
- *"\n" - new line*
- *"\x0B" - vertical tab*
- *"\r" - carriage return*
- *" " - ordinary white space*

Return Value: Returns the modified string

```
<?php
      $str = "      Mary Had A Little Lamb and She Loved It So";
      echo strlen($str);
      $str = ltrim($str);
      echo strlen($str);
?>
```

MD5 ( )

Returns the MD5 algorithm hash of a target string. The md5() string function in PHP is used to generate the MD5 algorithm hash for encrypting a string.

Since md5() is a one-way hash encryption it can be used to secure passwords and other sensitive data for a level of security. Not the highest level of security... but helpful nevertheless.

md5 ()

md5(string,raw)

string Required. The string to be calculated

raw Optional. Specifies hex or binary output format:

- *TRUE - Raw 16 character binary format*
- *FALSE - Default. 32 character hex number*

Return Value: Returns the calculated MD5 hash on success, or FALSE on failure

```
<?php
      $password="123456";
      echo md5($password);
?>
```

ORD ( )

Returns the ASCII representation of a character. The ord() string function in PHP is used to access the ASCII representation of the first character in a string. This function takes one parameter which is the string.

ord(string)

string Required. The string to get an ASCII value from

Return Value: Returns the ASCII value as an integer

```
<?php
      $ascii = ord('A');
      echo($ascii);
      $ascii = ord('AB');
      echo($ascii);
?>
```

PRINT ( )

String output similar to echo() . The print() string function in PHP is masquerading as a function but is really a language construct that works in a similar way to the echo().

print and echo are both used to output data to browser software or other technologies that intake external data. As you come to view different PHP scripts in your travels online you may notice some authors use echo and some use print. Let us discuss the difference.

echo() is a language construct, so you are not required to use parentheses with it. If you ever need to pass more than one parameter to echo(), the parameters must not be enclosed within parentheses.

print() behaves as a Function, but is not actually a function. print is sometimes mistaken as a function by many programmers since it sets a return value, but it is also a language construct like echo. It simply outputs a string of data. And you are not required to use parentheses with it.

print(strings)

strings Required. One or more strings to be sent to the output

Return Value: Always returns 1

```
<?php
      $str1="Hello world!";
      $str2="What a nice day!";
      print $str1 . " " . $str2;

      $color = "red";
      print "Roses are $color";
```

```
      print "<br>";
      print 'Roses are $color';
?>
```

RTRIM ( )

Trims a specified substring from the end of a string

The rtrim() PHP string function gives us a simple way to trim specified characters, substrings, or whitespace from the end of a string. When using two parameters we can specify a substring or set of character(s) to remove from the end of the target string. If using only one parameter it trims whitespace from the end of a string.

rtrim(string,charlist)

string Required. Specifies the string to check

charlist Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed:

- *"\0" - NULL*
- *"\t" - tab*
- *"\n" - new line*
- *"\x0B" - vertical tab*
- *"\r" - carriage return*
- *" " - ordinary white space*

Return Value: Returns the modified string

```
<?php
      $str = "Mary Had A Little Lamb and She Loved It So    ";
      echo strlen($str);
      $str = rtrim($str);
      echo strlen($str);
?>
```

STR_IREPLACE ( )

Case insensitive search and replace function. The str_ireplace() string function in PHP replaces target substrings with a substring you supply. It will replace all instances found. It is not case sensitive as demonstrated below. For a case sensitive search and replace function use **str_replace().**

str_ireplace(find,replace,string,count)

find Required. Specifies the value to find

replace Required. Specifies the value to replace the value in find

string Required. Specifies the string to be searched

count Optional. A variable that counts the number of replacements

```
<?php
    $string = "An infinite number of monkeys";
    $newstring = str_ireplace("Monkeys", "giraffes", $string);
    print $newstring;
?>
```

STR_REPLACE ( )

Case sensitive search and replace function. The str_replace() string function in PHP is just like the str_ireplace() function except that this function is case sensitive where str_ireplace() is not. It gives us a means to search for substrings within a string and replace them.

str_replace(find,replace,string,count)

find Required. Specifies the value to find
replace Required. Specifies the value to replace the value in find
string Required. Specifies the string to be searched
count Optional. A variable that counts the number of replacements

```
<?php
      $cat = str_replace("kitty", "puppy", "I love my kitty");
      echo $cat;

      $numbers = array("1", "2", "3");
      $words = array("one", "two", "three");
      $phrase = "He have 1 daughter and 3 sons";
      $change = str_replace($numbers, $words, $phrase);
      echo $change;

      $vowels = array("a", "e", "i", "o", "u");
      $no = str_replace($vowels, "", "My name is Joe Smith");
      echo $no;
?>
```

STR_SHUFFLE ( )

Randomly shuffle a string. The str_shuffle() string function in PHP will randomly shuffle a string or characters you pass through it. Each time this function executes it will randomly re-shuffle the string.

str_shuffle(string)

string Required. Specifies the string to shuffle

```
<?php
      $string = "United States";
      $string = str_shuffle($string);
      echo $string;
?>
```

STR_SPLIT ( )

Splits a string into a chunked array. The str_split() PHP string function will split a string into a chunked array. It will make an array out of a string you pass through it. If only fed one parameter(the string) it will split every character and space into an array element. If you pass str_split() a second parameter it will allow you to specify the chunk size of the split, so you can split the string every 5 characters if you like.

str_split(string,length)

string Required. Specifies the string to split

length Optional. Specifies the length of each array element. Default is 1

Return Value: If length is less than 1, the str_split() function will return FALSE. If length is larger than the length of string, the entire string will be returned as the only element of the array.

```
<?php
      $myString = "Programmers are dorks!";
      // Will split every character and space into array elements
      $myArray = str_split($myString);
      echo $myArray[0]."<br />";
      echo $myArray[1]."<br />";
      echo $myArray[2]."<br />";
      echo $myArray[3]."<br />";
      echo "etc...";
      echo "<hr />";
      // Will split a specified chuck of characters into array
elements
      $myArray = str_split($myString, 6);
      echo $myArray[0]."<br />";
      echo $myArray[1]."<br />";
      echo $myArray[2]."<br />";
      echo $myArray[3]."<br />";
?>
```

STRCASECMP ( )

Case insensitive string comparison function. The strcasecmp() function in PHP will help you compare two strings in a case-insensitive comparison. You will get a "0" as a result if the strings match in a case insensitive comparison. If string 2 is greater than string 1 you will get a negative value. If string 1 is greater than string 2 you will get a positive value.

strcasecmp(string1,string2)

string1 Required. Specifies the first string to compare

string2 Required. Specifies the second string to compare

Return Value: This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

```
<?php
      echo strcasecmp("Hello","HELLO");
      echo "<br>";
      echo strcasecmp("Hello","hELLo");
?>
```

STRCMP ( )

Case sensitive string comparison function. The strcmp() function in PHP will help you compare two strings in a case sensitive comparison. You will get a "0" as a result if the strings match in a case

insensitive comparison. If string 2 is greater than string 1 you will get a negative value. If string 1 is greater than string 2 you will get a positive value.

strcmp(string1,string2)

string1 Required. Specifies the first string to compare

string2 Required. Specifies the second string to compare

Return Value: This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

```
<?php
      $str1 = "b"; //asci 98
      $str2 = "t"; //ascii 116
      $str1 = "bear";
      $str2 = "tear";
      $str3 = "";
      echo strcmp($str1, $str2); // -18
      echo "<br/>";
      echo strcmp($str2, $str1); //18
      echo "<br/>";
      echo strcmp($str2, $str2); //0
      echo "<br/>";
      echo strcmp($str2, $str3); //4
      echo "<br/>";
      echo strcmp($str3, $str2); //-4
      echo "<br/>";
      echo strcmp($str3, $str3); // 0
?>
```

STRCSPN()

The strcspn() function returns the number of characters (including whitespaces) found in a string before any part of the specified characters are found. Use the strspn() function to the number of characters found in the string that contains only characters from a specified character list.

strcspn(string,char,start,length)

string Required. Specifies the string to search

char Required. Specifies the characters to search for

start Optional. Specifies where in string to start

length Optional. Specifies the length of the string (how much of the string to search)

```
<?php
      $a = strcspn('abcd',  'apple');
      $b = strcspn('abcd',  'banana');
      $c = strcspn('hello', 'l');
      $d = strcspn('hello', 'world');
      echo($a);  echo($b);
      echo($c);  echo($d);
?>
```

STRISTR ( )

Find the first occurrence of a substring in a string. The stristr() string function in PHP will return the entire end of a string where the first occurrence of a specified case insensitive substring is found. To use the case sensitive version of this function simply trade out stristr() for strstr() in the code.

stristr(string,search,before_search)

string Required. Specifies the string to search

search Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number

before_search Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the search parameter.

```
<?php
      $string = 'Hello World!';
      echo stristr($string, 'earth');
      echo stristr($string, 'World');
      $string = 'APPLE';
      echo stristr($string, 97);
?>
```

STRLEN ( )

Returns the length of a string. The strlen() PHP string function will return a length count for the amount of characters with a string. It will also count spaces as a character, and if need be one could use the str_ireplace() function on the string to remove all spaces before creating the count of the string's length.

strlen(string)

string Required. Specifies the string to check

Return Value: Returns the length of a string on success, and 0 if the string is empty

```
<?php
      $str = "Mary Had A Little Lamb and She Loved It So";
      $str = strlen($str);
      echo $str;
?>
```

STRPBRK()

The strpbrk() function searches a string for any of the specified characters. This function is case-sensitive.This function returns the rest of the string from where it found the first occurrence of a specified character, otherwise it returns FALSE.

strpbrk(string,charlist)

string Required. Specifies the string to search

charlist Required. Specifies the characters to find

Return Value: Returns the string starting from the character found, otherwise it returns FALSE

```
<?php
      echo "W will output: " . strpbrk("Hello world!","W");
      echo "<br>";
      echo "w will output: " . strpbrk("Hello world!","w");
?>
```

STRIPOS ( )

Find the position of the first found substring occurrence. The stripos() string function in PHP will return a number representing the position of the first occurrence of a specified case insensitive substring within a string. If the substring is found you will get a numeric result representing where the substring has first occurred in the string. If the substring is not found PHP will return a NULL value for $result.

stripos(string,find,start)

string Required. Specifies the string to search
find Required. Specifies the string to find
start Optional. Specifies where to begin the search

```
<?php
      $findme    = 'a';
      $mystring1 = 'xyz';
      $mystring2 = 'ABC';

      $pos1 = stripos($mystring1, $findme);
      $pos2 = stripos($mystring2, $findme);

      // Nope, 'a' is certainly not in 'xyz'
      if ($pos1 === false) {
            echo "The string '$findme' was not found in the string
'$mystring1'";
      }

      // Note our use of ===.  Simply == would not work as expected
      // because the position of 'a' is the 0th (first) character.
      if ($pos2 !== false) {
            echo "We found '$findme' in '$mystring2' at position
$pos2";
      }
?>
```

STRPOS()

The strpos() function finds the position of the first occurrence of a string inside another string.The strpos() function is case-sensitive.

strpos(string,find,start)

string Required. Specifies the string to search
find Required. Specifies the string to find
start Optional. Specifies where to begin the search

```
<?php
      $numberedString = "1234567890";
      $fivePos = strpos($numberedString, "5");
      echo "The position of 5 in our string was $fivePos";
?>
```

STRREV()

The strrev() function reverses a string.

strrev(string)

string Required. Specifies the string to reverse

Return Value: Returns the reversed string

```
<?php
      $st="This is a test";
      echo "The string = ".$st;
      $st=strrev($st);
      echo "The new string after applying the strrev function =
".$st;
?>
```

STRRPOS()

The strrpos() function finds the position of the last occurrence of a string inside another string. The strrpos() function is case-sensitive.

strrpos(string,find,start)

string Required. Specifies the string to search

find Required. Specifies the string to find

start Optional. Specifies where to begin the search

Return Value: Returns the position of the last occurrence of a string inside another string, or FALSE if the string is not found. Note: String positions start at 0, and not 1.

```
<?php
      echo strrpos("Good Morning","o");
?>
```

STRRIPOS()

The strripos() function finds the position of the last occurrence of a string inside another string. The strripos() function is case-insensitive.

strripos(string,find,start)

string Required. Specifies the string to search

find Required. Specifies the string to find

start Optional. Specifies where to begin the search

Return Value: Returns the position of the last occurrence of a string inside another string, or FALSE if the string is not found. Note: String positions start at 0, and not 1.

```
<?php
      $haystack = 'ababcd';
      $needle   = 'aB';
```

```
      $pos      = strripos($haystack, $needle);
      if ($pos === false) {
            echo "Sorry, we did not find ($needle) in ($haystack)";
      } else {
            echo "Congratulations!\n";
            echo "We found the last ($needle) in ($haystack) at
position ($pos)";
      }
?>
```

STRSTR()

The strstr() function searches for the first occurrence of a string inside another string.

strstr(string,search,before_search)

string Required. Specifies the string to search

search Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number

before_search Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the search parameter.

```
<?php
      $email  = 'mohammad@test.com';
      $domain = strstr($email, '@');
      echo $domain;

      $user = strstr($email, '@', true);
      echo $user;
?>
```

STRTOLOWER ( )

Convert all capital letters to lowercase. The strtolower() string function in PHP will convert any capital letters found in your string to lowercase. It works the opposite of strtoupper().

strtolower(string)

string Required. Specifies the string to convert

Return Value: Returns the the lowercased string

```
<?php
      $str = "Mary Had A Little Lamb and She Loved It So";
      $str = strtolower($str);
      echo $str; // Prints mary had a little lamb and she loved it
so
?>
```

STRTOUPPER ( )

Convert all lowercase letters to capitals. The strtoupper() string function in PHP will convert any lowercase letters found in your string to uppercase. It works the opposite of strtolower().

strtoupper(string)

string Required. Specifies the string to convert
Return Value: Returns the the uppercased string

```
<?php
      $str = "Mary Had A Little Lamb and She Loved It So";
      $str = strtoupper($str);
      echo $str; // Prints mary had a little lamb and she loved it
so
?>
```

SUBSTR ( )

Return a specified portion of a string. The substr() string function in PHP will return a part of a string according to the parameters you feed it. It can take up to three parameters. The first parameter is the string. The second parameter is the starting position in the string(can be a negative number to count from the end of the string if you like). The third parameter is the length of how many characters in the string that you wish to select and make a substring out of.

substr(string,start,length)

string Required. Specifies the string to return a part of
start Required. Specifies where to start in the string
A positive number - Start at a specified position in the string
A negative number - Start at a specified position from the end of the string
0 - Start at the first character in string.
length Optional. Specifies the length of the returned string. *Default is to the end of the string.*
A positive number - The length to be returned from the start parameter
Negative number - The length to be returned from the end of the string
Return Value: Returns the extracted part of a string, or FALSE on failure, or an empty string

```
<?php
      $str = substr("abcdef", -1);    // returns "f"
      echo $str;
      $str = substr("abcdef", -2);    // returns "ef"
      echo $str;
      $str = substr("abcdef", -3, 1); // returns "d"
      echo $str;
?>
```

SUBSTR_COUNT ( )

Count the number of times a substring is found. The substr_count() string function in PHP will count how many times a substring is found in of a string. The substring you specify is case sensitive to this function.

substr_count(string,substring,start,length)

string Required. Specifies the string to check
substring Required. Specifies the string to search for
start Optional. Specifies where in string to start searching
length Optional. Specifies the length of the search

```
<?php
      $text = 'This is a test';
      echo strlen($text); // 14
      echo substr_count($text, 'is'); // 2
      // the string is reduced to 's is a test', so it prints 1
      echo substr_count($text, 'is', 3);
      // the text is reduced to 's i', so it prints 0
      echo substr_count($text, 'is', 3, 3);
      // generates a warning because 5+10 > 14
      echo substr_count($text, 'is', 5, 10);
?>
```

SUBSTR_COMPARE()

The substr_compare() function compares two strings from a specified start position.This function is binary-safe and optionally case-sensitive.

substr_compare(string1,string2,startpos,length,case)

string1 Required. Specifies the first string to compare
string2 Required. Specifies the second string to compare
startpos Required. Specifies where to start comparing in string1. If negative, it starts counting from the end of the string
length Optional. Specifies how much of string1 to compare
case Optional. A boolean value that specifies whether or not to perform a case-sensitive compare:
FALSE - Default. Case-sensitive
TRUE - Case-insensitive

```
<?php
      echo substr_compare("abcde", "bc", 1, 2); // 0
      echo substr_compare("abcde", "bcg", 1, 2); // 0
      echo substr_compare("abcde", "BC", 1, 2, true); // 0
      echo substr_compare("abcde", "bc", 1, 3); // 1
      echo substr_compare("abcde", "cd", 1, 2); // -1
      echo substr_compare("abcde", "abc", 5, 1); // warning
?>
```

TRIM ( )

Remove whitespace(or chars) from both ends of a string. The trim() string function in PHP will remove whitespace or specified characters from the end and beginning of a string. This function can take up to two parameters. If only fed one parameter it will remove whitespace from both ends. If fed a second parameter(a specified character list), it will remove the specified characters from both ends of the string.

trim(string,charlist)

string Required. Specifies the string to check
charlist Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed:

- *"\0" - NULL*
- *"\t" - tab*
- *"\n" - new line*
- *"\x0B" - vertical tab*
- *"\r" - carriage return*
- *" " - ordinary white space*

```
<?php
      $str = "    Mary Had A Little Lamb and She Loved It So    ";
      echo strlen($str);
      $str = trim($str);
      echo strlen($str);
?>
```

UCFIRST ( )

Capitalize the first letter of a string. The ucfirst() string function in PHP will capitalize the first letter in a string.

ucfirst(string)
string Required. Specifies the string to convert
Return Value: Returns the converted string

```
<?php
            $string = "i like to program in PHP";
            $a = strtoupper($string);
            $b = strtolower($string);
            $c = ucfirst($string);
            $d = ucwords($string);
            $e = ucwords(strtolower($string));
            echo $a."<br />";
            echo $b."<br />";
            echo $c."<br />";
            echo $d."<br />";
            echo $e;
?>
```

UCWORDS ( )

Capitalize the first letter of each word in a string. The ucwords() string function in PHP will capitalize the first letter of each word in a string if the first character is indeed a letter.

ucwords(string)
string Required. Specifies the string to convert
Return Value: Returns the converted string

```
<?php
            $string = "i like to program in PHP";
            $a = strtoupper($string);
            $b = strtolower($string);
```

```
            $c = ucfirst($string);
            $d = ucwords($string);
            $e = ucwords(strtolower($string));
            echo $a."<br />";
            echo $b."<br />";
            echo $c."<br />";
            echo $d."<br />";
            echo $e;
?>
```

VARIABLE FUNCTIONS

The gettype() function is used to get the type of a variable.

gettype(var_name)

var_name Required. The name of the variable. Mixed* : Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

Return value : Possible values of the returned string : (boolean, integer, double, string, array,object,resource,NULL,unknown type)

```
<?php
      $data = array(1, 1.1, NULL, true, 'foo');
      foreach ($data as $value) {
            echo gettype($value), "<br />";
      }
?>
```

SETTYPE()

The settype() function is used to set the type of a variable.

settype(var_name, var_type)

var_name Required. The variable being converted.Mixed* : Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

var_type Type of the variable. Possible values are : boolean, integer, float, string, array, object, null.

Return value : TRUE on success or FALSE on failure.

```
<?php
      $var1 = "5var2"; // string
      $var2 = true;   // boolean
      settype($var1, "integer"); // $var1 is now 5   (integer)
      settype($var2, "string");  // $var2 is now "1" (string)
?>
```

ISSET()

The isset () function is used to check whether a variable is set or not. The isset() function return false if testing variable contains a NULL value.

isset(variable1, variable2.....)

variable1 The variable being checked Required Mixed* : Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

Return value : TRUE if variable (variable1,variable2..) exists and has value not equal to NULL, FALSE otherwise.

```
<?php
      $var1 = 'test';
      if(isset($var1))
            echo "var1 Variable exists";
      else
            echo "var1 does not exists";
      if(isset($var2))
            echo "var2 Variable exists";
      else
            echo "var2 does not exists";
?>
```

UNSET()

The unset() function destroys a given variable.

unset (var1, var2....)

var1 The variable to be unset. Required Mixed*

var2 Another variable to be unset Optional Mixed*

*Mixed : Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

Return value : No value is returned.

```
<?php
      $var='http://www.youtube.com';
      echo 'Before using unset() the value of var is : '.
$var.'<br>';
      unset($var);
      echo 'After using unset() the value of var is : '. $var;
?>
```

STRVAL()

The strval() is used to convert a value of a variable to a string.

strval(var_name)

var_name The variable name. Required Mixed* : Mixed indicates that a parameter may accept multiple (but not necessarily all) types.

Return value : The string value of var_name.

```
<?php
      $value = "1.223";
      print strval ($value);
?>
```

FLOATVAL()

The floatval() function is used to convert a value to a float.

floatval (var1)

var1 Required . May be any scalar type. You cannot use floatval() on arrays or objects. Mixed*

Return value : The float value of the given variable.

```
<?php
      $value = "1.223";
      print floatval ($value);
?>
```

INTVAL()

The intval() function is used to get the integer value of a variable.

intval(var_name, base)

var_name Required The scalar value being converted to an integer. Mixed*

base Optional The base for the conversion. (default is base 10)

Return value : The integer value of var on success, or 0 on failure

```
<?php
      $value = "1.223";
      print intval ($value);
?>
```

PRINT_R()

The print_r() function is used to print human-readable information about a variable.

print_r(var_name)

var_name The variable being printed. Required String

Return value : If the variable is an integer or a float or a string the function returns value of the variable. If the variable is an array the function returns keys and elements, a similarly notation is used for the object.

```
<?php
       $names = array ('a' => 'Angela', 'b' => 'Bradley', 'c' =>
array ('Cade', 'Caleb'));
       print_r ($names);
       echo "<pre>";
       $number = 10.5;
       print_r($number);
       echo "</pre>";
?>
```

HANDLING FORM, SESSION TRACKING, PHP COMPONENTS& AJAX

PHP FORM HANDLING/PROCESSING

The PHP superglobals $_GET and $_POST are used to collect form-data.

The example below contains a simple HTML form with two input fields and a submit button:

```
<html>
	<body>
		<form action="welcome.php" method="post">
			Name: <input type="text" name="fname"><br>
			Age: <input type="text" name="age"><br>
			<input type="submit">
		</form>
	</body>
</html>
```

When a user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with method="post".

"welcome.php" looks like this:

```
<html>
	<body>
		Welcome <?php echo $_POST["fname"]; ?>!<br>
		You are <?php echo $_POST["age"]; ?> years old.
	</body>
</html>
```

The output could be something like this:

Welcome John!

You are 28 years old.

The same result could also be achieved using method="get":

```
<html>
	<body>
		<form action="welcome_get.php" method="get">
			Name: <input type="text" name="fname"><br>
			Age: <input type="text" name="age"><br>
			<input type="submit">
		</form>
	</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
      <body>
            Welcome <?php echo $_GET["fname"]; ?>!<br>
            You are <?php echo $_GET["age"]; ?> years old.
      </body>
</html>
```

GET vs. POST

	Get	Post
History	Parameters remain in browser history because they are part of the URL	Parameters are not saved in browser history.
Bookmarked	Can be bookmarked.	Can not be bookmarked.
BACK button/re-submit behavior	GET requests are re-executed but may not be re-submitted to server if the HTML is stored in the browser cache.	The browser usually alerts the user that data will need to be re-submitted.
Encoding type (enctype attribute)	application/x-www-form-urlencoded	multipart/form-data or application/x-www-form-urlencoded Use multipart encoding for binary data.
Parameters	can send but the parameter data is limited to what we can stuff into the request line (URL). Safest to use less than 2K of parameters, some servers handle up to 64K	Can send parameters, including uploading files, to the server.
Hacked	Easier to hack for script kiddies	More difficult to hack
Restrictions on form data type	Yes, only ASCII characters allowed.	No restrictions. Binary data is also allowed.
Security	GET is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in plaintext.	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs.
Restrictions on form data length	Yes, since form data is in the URL and URL length is restricted. A safe URL length limit is often 2048 characters but varies by browser and web server.	No restrictions
Usability	GET method should not be used when sending passwords or other sensitive information.	POST method used when sending passwords or other sensitive information.
Visibility	GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.	POST method variables are not displayed in the URL.
Cached	Can be cached	Not cached

COOKIES

Cookies are text files stored on the client computer that can each contain around 4000 characters. Up to 20 cookies can be stored for each website, and the client computer can store a maximum of 300 cookies in total. Cookie files can be opened with any text editor so sensitive information should be encrypted. They are useful to retain user preferences, shopping cart selection, and other snippets of data. For instance, rather than directly store a user's password in a cookie it is better to store a unique identification string that references a database entry containing the password on the server.

SETCOOKIE

Defines a cookie to be sent along with the rest of the HTTP headers.

`setcookie ( string name [, string value [, int expire]] )`

Parameter	Description	Examples
name	The name of the cookie.	'cookiename' is called as $_COOKIE['cookiename']
value	The value of the cookie. This value is stored on the client's computer; do not store sensitive information.	Assuming the name is 'cookiename', this value is retrieved through $_COOKIE['cookiename']
expire	The time on which cookie has to expires. This is a Unix timestamp so is in number of seconds since the epoch. In other words, you'll most likely set this with the time() function plus the number of seconds before you want it to expire. Or you might use mktime().	Time () +60*60*24*30 will set the cookie to expire in 30 days. If not set, the cookie will expire at the end of the session (when the browser closes).

```
<?php
      $value = 'The data for the cookie';
      setcookie("cookie1", $value);
      setcookie("cookie2", $value, time()+86400);  /* expire after 1
day */
?>
```

```
<?php
      $cookie_value = "mohammad";
      setcookie("name", $cookie_value, time()+3600);
      if (isset($_COOKIE['name']))
            echo $_COOKIE['name'];
?>
```

```
<?php
      // set the cookies
      setcookie("cookie[three]", "cookiethree");
      setcookie("cookie[two]", "cookietwo");
      setcookie("cookie[one]", "cookieone");
```

```
      // after the page reloads, print them out
      if (isset($_COOKIE['cookie'])) {
         foreach ($_COOKIE['cookie'] as $name => $value) {
             echo "$name : $value <br />\n";
         }
      }
?>
```

```
<?php
/* These are our valid username and passwords */
      $user = 'bharmal';
      $pass = 'mohammad';

      if (isset($_POST['username']) && isset($_POST['password')) {

            if (($_POST['username'] == $user) && ($_POST['password']
== $pass)) {

                  if (isset($_POST['rememberme'])) {
                        /* Set cookie to last 1 year */
                        setcookie('username',      $_POST['username'],
time()+60*60*24*365, '/account', 'www.example.com');
                        setcookie('password', md5($_POST['password']),
time()+60*60*24*365, '/account', 'www.example.com');

                  } else {
                        /* Cookie expires when browser closes */
                        setcookie('username',      $_POST['username'],
false, '/account', 'www.example.com');
                        setcookie('password', md5($_POST['password']),
false, '/account', 'www.example.com');
                  }
                  header('Location: index.php');

            } else {
                  echo 'Username/Password Invalid';
            }

      } else {
            echo 'You must supply a username and password.';
      }
?>
<html>
      <head>
            <title>User Logon</title>
```

```
      </head>
      <body>
            <h2>User Login </h2>
            <form name="login" method="post">
               Username: <input type="text" name="username"><br>
               Password: <input type="password" name="password"><br>
               Remember Me: <input type="checkbox" name="rememberme"
value="1"><br>
                  <input type="submit" name="submit" value="Login!">
            </form>
      </body>
</html>
```

```
<?php
      $expire=time()+60*60*24*30;
      setcookie("user", "Alex Porter", $expire); //create Cookie
      setcookie("user", "", time()-3600); //Destroy cookie
?>
```

SESSION

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.The location of the temporary file is determined by a setting in the php.ini file called session.save_path. Bore using any session variable make sure you have setup this path.

When a session is started following things happen:

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

SESSION_START()

A PHP session is easily started by making a call to the **session_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

```
<?php
      // page1.php
      session_start();
      echo 'Welcome to page #1';
      $_SESSION['favcolor'] = 'green';
      $_SESSION['animal']   = 'cat';
      $_SESSION['time']     = time();

      // Works if session cookie was accepted
      echo '<br /><a href="page2.php">page 2</a>';

      // Or maybe pass along the session id, if needed
      echo  '<br  /><a  href="page2.php?'  .  session_id()  .  '">page
2</a>';
?>
```

SESSION_DESTROY

Destroys all data registered to a session.

```
<?php      session_start();      ?>
<!DOCTYPE html>
<html>
      <body>
      <?php
            // remove all session variables
            session_unset();
            // destroy the session
            session_destroy();
      ?>
      </body>
</html>
```

```
//extra example
<?php
      session_start();
      if( isset( $_SESSION['counter'] ) ) {
            $_SESSION['counter'] += 1;
      }else {
            $_SESSION['counter'] = 1;
      }
      $msg = "You have visited this page ".  $_SESSION['counter'];
      $msg .= "in this session.";
?>
<html>
      <head>
            <title>Setting up a PHP session</title>
      </head>
```

```
      <body>
            <?php  echo ( $msg ); ?>
      </body>
</html>
```

SESSIONS WITHOUT COOKIES

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

```
<?php
   session_start();
   if (isset($_SESSION['counter'])) {
      $_SESSION['counter'] = 1;
   }else {
      $_SESSION['counter']++;
   }
   $msg = "You have visited this page ".  $_SESSION['counter'];
   $msg .= "in this session.";
   echo ( $msg );
?>
<p>
   To continue  click following link <br />
<a  href = "nextpage.php?<?php echo htmlspecialchars(SID); ?>">
</p>
```

SERVER VARIABLE

$_SERVER

$_SERVER is introduced in PHP version 4.1.0 and was known as $HTTP_SERVER_VARS. As the $_SERVER is an array containing the all information the server, different environment, all sort of path, server software, port of server, ip of server and various other details. The details stored in the variable are not always same and number of details is also not same, it may vary from system to system. The $_SERVER variable is predefined, array, superglobal variable so it can be accessed in any script at any number of times. You don't need to declare it as global with global keyword.

```
<?php
      echo "<pre>"; print_r($_SERVER); echo "</pre>";
?>
```

Element	Description
'PHP_SELF'	The filename of the currently executing script, relative to the document root. For instance, $_SERVER['PHP_SELF'] in a script at the address http://example.com/test.php would be /test.php.
'argv'	Array of arguments passed to the script. When the script is run on the command line, this gives C-style access to the command line parameters. When called via the GET method, this will contain the query string.
'argc'	Contains the number of command line parameters passed to the script (if run on the command line).
'SERVER_NAME'	The name of the server host under which the current script is executing. If the script is running on a virtual host, this will be the value defined for that virtual host.
'SERVER_SOFTWARE'	Server identification string, given in the headers when responding to requests.
'SERVER_PROTOCOL'	Name and revision of the information protocol via which the page was requested; i.e. 'HTTP/1.0';
'REQUEST_METHOD'	Which request method was used to access the page; i.e. 'GET', 'HEAD', 'POST', 'PUT'.
'DOCUMENT_ROOT'	The document root directory under which the current script is executing, as defined in the server's configuration file.
'HTTP_ACCEPT_LANGUAGE'	Contents of the Accept-Language: header from the current request, if there is one. Example: 'en'.
'HTTP_USER_AGENT'	Contents of the User-Agent: header from the current request, if there is one. This is a string denoting the user agent being which is accessing the page. A typical example is: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Among other things, you can use this value with get_browser() to tailor your page's output to the capabilities of the user agent.
'REMOTE_ADDR'	The IP address from which the user is viewing the current page.
'REMOTE_HOST'	The Host name from which the user is viewing the current page. The reverse dns lookup is based off the REMOTE_ADDR of the user. Your web server must be configured to create this variable. For example in Apache you'll need HostnameLookups On inside httpd.conf for it to exist. See also gethostbyaddr().
'REMOTE_PORT'	The port being used on the user's machine to communicate with the web server.
'SCRIPT_FILENAME'	The absolute pathname of the currently executing script.

'SERVER_ADMIN'	The value given to the SERVER_ADMIN (for Apache) directive in the web server configuration file. If the script is running on a virtual host, this will be the value defined for that virtual host.
'SERVER_PORT'	The port on the server machine being used by the web server for communication. For default setups, this will be '80'; using SSL, for instance, will change this to whatever your defined secure HTTP port is.
'SERVER_SIGNATURE'	String containing the server version and virtual host name which are added to server-generated pages, if enabled.
'SCRIPT_NAME'	Contains the current script's path. This is useful for pages which need to point to themselves. The __FILE__ constant contains the full path and filename of the current (i.e. included) file.
'REQUEST_URL'	The URI which was given in order to access this page; for instance, '/index.html'.

$_FILES

It is an associative array of the items which are uploaded [send to server] by the client [by the user] via the HTTP POST method. The variable is predefined, array, superglobal variable so it can be accessed in any script at any number of times. You don't need to declare it as global with global keyword. It store the all required information of the files which are required to transfer the file from the client machine to server. The information would be name of file, type of file, size of file, etc as the sub associative array of the element by which the user has send the file. Suppose the file is received as 'userfile' from the client it store information in the following manner.

Elements	Descriptions
$_FILES['userfile']['name']	The original name of the file on the client machine.
$_FILES['userfile']['type']	The mime type of the file, if the browser provided this information. An example would be "image/gif".
$_FILES['userfile']['size']	The size, in bytes, of the uploaded file.
$_FILES['userfile']['tmp_name']	The temporary filename of the file in which the uploaded file was stored on the server.
$_FILES['userfile']['error']	The error code associated with this file upload. This element was added in PHP 4.2.0

Error Code	Descriptions
UPLOAD_ERR_OK	Value: 0; There is no error, the file uploaded with success.
UPLOAD_ERR_INI_SIZE	Value: 1; The uploaded file exceeds the upload_max_filesize directive in php.ini.

UPLOAD_ERR_FORM_SIZE	Value: 2; The uploaded file exceeds the *MAX_FILE_SIZE*directive that was specified in the HTML form.
UPLOAD_ERR_PARTIAL	Value: 3; The uploaded file was only partially uploaded.
UPLOAD_ERR_NO_FILE	Value: 4; No file was uploaded.
UPLOAD_ERR_NO_TMP_DIR	Value: 6; Missing a temporary folder. Introduced in PHP 4.3.10 and PHP 5.0.3.

Sending [Uploading] the multiple file

```
<form action="<?php $_SERVER['PHP_SELF'] ?>" method="post"
enctype="multipart/form-data">
 Send these files:<br />
<input name="userfile[]" type="file" /><br />
<input name="userfile[]" type="file" /><br />
<input name="userfile[]" type="file" /><br />
<input name="userfile[]" type="file" /><br />
<input type="submit" value="Send files" />
</form>
<?php
      $len = count($_FILES['userfile']['name']);
      echo $len;
      $uploaddir = 'c:\\';
      for($x=0;$x<$len;$x++)
      {
            echo $x;
            $uploadfile = $uploaddir .
basename($_FILES['userfile']['name'][$x]);
            echo '<pre>';
            if
(move_uploaded_file($_FILES['userfile']['tmp_name'][$x],
$uploadfile)) {
                  echo "File is valid, and was successfully
uploaded.\n";
            }
            else {
                  echo "Possible file upload attack!\n";
            }
            echo 'Here is some more debugging info:';
            print_r($_FILES);
            print "</pre>";
      }
?>
```

$_GET

It is an associative array of the values passed to the current script via the HTTP GET method. The variable is predefined, array, superglobal variable so it can be accessed in any script at any number of times. You don't need to declare it as global with global keyword.

```
<form method="get" action="<?php $_SERVER['PHP_SELF'] ?>">
      <input type="text" name="fst"><br>
      <input type="text" name="scd"><br>
      <input                  type="radio"                  name="ch"
value="add">Add    <input type="radio" name="ch"
value="sub">Subtract    <br>
      <input                  type="radio"                  name="ch"
value="mul">Multiple    <input      type="radio"
name="ch" value="div">Division<br>
      <input type="submit" name="check" value="Calculate">
</form>
<?php
      if(isset($_GET['check']))
      {
            $ch=$_GET['ch'];
            $fst=$_GET['fst'];
            $scd=$_GET['scd'];
            switch($ch)
            {
                  case "add":
                        echo $fst+$scd;
                        break;
                  case "sub":
                        echo $fst-$scd;
                        break;
                  case "div":
                        echo $fst/$scd;
                        break;
                  case "mul":
                        echo $fst*$scd;
                        break;
            }
      }
      else
            echo "Select some choice";
?>
```

$GLOBALS

$GLOBALS is a php super global variable which can be used instead of 'global' keyword to access variables from global scope, i.e. the variables which can be accessed from anywhere in a php script even within functions or methods.

```
<?php
	$s = 25;
	$t = 50;
	function subtraction()
	{
		$GLOBALS['v'] = $GLOBALS['t'] - $GLOBALS['s'];
	}
	subtraction();
	echo $v;
?>
```

$_REQUEST

This variable contains the content of both $_GET, $_COOKIE, and $_POST.This can get the values passed from the form with both the GET and POST methods.

```
<form action="myphp.php" method="POST">
	Firstname: <input type="text" name="firstname" />
	Lastname: <input type="text" name="lastname" />
	<input type="submit" />
</form>
<?php
	echo "Firstname: " $_REQUEST["firstname"]; . <br />;
	echo "Lastname: " $_REQUEST["lastname"];
?>
```

OR

```
<form action="myphp.php" method="GET">
	Firstname: <input type="text" name="firstname" />
	Lastname: <input type="text" name="lastname" />
	<input type="submit" />
</form>

<?php
	echo "Firstname: " . $_REQUEST["firstname"] . "<br />";
	echo "Lastname: " . $_REQUEST["lastname"] ;
?>
```

$_POST

It is an associative array of the values passed to the current script via the HTTP POST method. The variable is predefined, array, superglobal variable so it can be accessed in any script at any number of times. You don't need to declare it as global with global keyword.

```
<form method="POST" action="<?php echo $_SERVER['PHP_SELF'] ?>">
	<input type="text" name="fst"><br>
	<input type="text" name="scd"><br>
```

```
      <input                    type="radio"                    name="ch"
value="add">Add    <input type="radio" name="ch"
value="sub">Subtract    <br>
      <input                    type="radio"                    name="ch"
value="mul">Multiple    <input      type="radio"
name="ch" value="div">Division<br>
      <input type="submit" name="check" value="Calculate">
</form>
<?php
      if(isset($_POST['check']))
      {
            $ch=$_POST['ch'];
            $fst=$_POST['fst'];
            $scd=$_POST['scd'];
            switch($ch){
                  case "add":
                        echo $fst+$scd;
                        break;
                  case "sub":
                        echo $fst-$scd;
                        break;
                  case "div":
                        echo $fst/$scd;
                        break;
                  case "mul":
                        echo $fst*$scd;
                        break;
            }
      }
      else
            echo "Select some choice";
?>
```

$_COOKIE

An associative array of variables passed to the current script via HTTP cookies. Automatically global in any scope. This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a global $_COOKIE; to access it within functions or methods, as you do with $HTTP_COOKIE_VARS. $HTTP_COOKIE_VARS contains the same initial information, but is not an autoglobal.

$_SESSION

An associative array containing session variables available to the current script. This simply means that it is available in all scopes throughout a script. You don't need to do a global $_SESSION; to access it within functions or methods, as you do with $HTTP_SESSION_VARS. $HTTP_SESSION_VARS contains the same information, but is not an autoglobal.

PHP REGULAR EXPRESSION

POSIX REGULAR EXPRESSIONS:

The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions. The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

[0-9] *It matches any decimal digit from 0 through 9.*
[a-z] *It matches any character from lowercase a through lowercase z.*
[A-Z] *It matches any character from uppercase A through uppercase Z.*
[a-Z] *It matches any character from lowercase a through uppercase Z.*

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

Quantifiers, The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character having a specific connotation. The +, *, ?, {int. range}, and $ flags all follow a character sequence.

p+ *It matches any string containing at least one p.*
p* *It matches any string containing zero or more p's.*
p? *It matches any string containing zero or more p's. This is just an alternative way to use p*.*
p{N} *It matches any string containing a sequence of N p's*
p{2,3} *It matches any string containing a sequence of two or three p's.*
p{2, } *It matches any string containing a sequence of at least two p's.*
p$ *It matches any string with p at the end of it.*
^p *It matches any string with p at the beginning of it.*
[^a-zA-Z] *It matches any string not containing any of the characters ranging from a through z and A through Z.*
p.p *It matches any string containing p, followed by any character, in turn followed by another p.*
^.{2}$ *It matches any string containing exactly two characters.*
(.*) *It matches any string enclosed within and .*
p(hp)* *It matches any string containing a p followed by zero or more instances of the sequence hp.*

PHP's Regexp POSIX Functions

PHP currently offers seven functions for searching strings using POSIX-style regular expressions:

ereg() *The ereg() function searches a string specified by string for a string specified by pattern, returning true if the pattern is found, and false otherwise.*

ereg_replace() *The ereg_replace() function searches for string specified by pattern and replaces pattern with replacement if found.*

eregi() *The eregi() function searches throughout a string specified by pattern for a string specified by string. The search is not case sensitive.*

eregi_replace() *The eregi_replace() function operates exactly like ereg_replace(), except that the search for pattern in string is not case sensitive.*

split() *The split() function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.*

spliti() *The spliti() function operates exactly in the same manner as its sibling split(), except that it is not case sensitive.*

sql_regcase() *The sql_regcase() function can be thought of as a utility function, converting each character in the input parameter string into a bracketed expression containing two characters.*

PERL STYLE REGULAR EXPRESSIONS:

Perl-style regular expressions are similar to their POSIX counterparts. The POSIX syntax can be used almost interchangeably with the Perl-style regular expression functions. In fact, you can use any of the quantifiers introduced in the previous POSIX section. Let's give explanation for few concepts being used in PERL regular expressions. After that I will introduce you with regular expression related functions.

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning. For instance, you can search for large money sums using the '\d' metacharacter: /([\d]+)000/, Here \d will search for any string of numerical character. Following is the list of metacharacters which can be used in PERL Style Regular Expressions.

. *a single character*
\s *a whitespace character (space, tab, newline)*
\S *non-whitespace character*
\d *a digit (0-9)*
\D *a non-digit*
\w *word character (a-z, A-Z, 0-9, _)*
\W *a non-word character*
[aeiou] *matches a single character in the given set*
[^aeiou] *matches a single character outside the given set*
(foo|bar|baz) *matches any of the alternatives specified*

Several modifiers are available that can make your work with regexps much easier, like casesensitivity, searching in multiple lines etc.

i *Makes the match case insensitive*
o *Evaluates the expression only once*
s *Allows use of . to match a newline character*
x *Allows you to use white space in the expression for clarity*
g *Globally finds all matches*
cg *Allows a search to continue even after a global match fails*

PHP'S REGEXP PERL COMPATIBLE FUNCTIONS

PHP offers following functions for searching strings using Perl-compatible regular expressions:

preg_match() *The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.*

preg_replace() *The preg_replace() function operates just like ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.*

preg_split() *The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.*

preg_grep() *The preg_grep() function searches all elements of input_array, returning all elements matching the regexp pattern.*

preg_match()

The performs pattern matching on a string. preg_match() takes two basic and three optional parameters. These parameters are, in order, a regular expression string, a source string, an array variable which stores matches, a flag argument and an offset parameter that can be used to specify the alternate place from which to start the search: preg_match (pattern, subject [, matches]). The preg_match() function returns 1 if a match is found and 0 otherwise.

```
<?php
      if (preg_match("/ell/", "Hello World!", $matches))
      {
        echo "Match was found 'ell'<br />";
        echo $matches[0];
      }
?>
```

```
<?php
      if   (preg_match("/ll.*/",   "The   History   of   Halloween",
$matches)) {
        echo "Match was found <br />";   echo $matches[0];
      }
?>
```

The example below checks if the password is "strong", i.e. the password must be at least 8 characters and must contain at least one lower case letter, one upper case letter and one digit:

```
<?php
      $password = "Fyfjk34sdfjfsjq7";
      if        (preg_match("/^.*(?=.{8,})(?=.*\d)(?=.*[a-z])(?=.*[A-
Z]).*$/", $password)) {
          echo "Your passwords is strong.";
      } else {
          echo "Your password is weak.";
      }
?>
```

The ^ and $ are looking for something at the start and the end of the string. The ".*" combination is used at both the start and the end. As mentioned above, the .(dot) metacharacter means any alphanumeric character, and * metacharacter means "zero or more". Between are groupings in parentheses. The "?=" combination means "the next text must be like this". This construct doesn't capture the text. In this example, instead of specifying the order that things should appear, it's saying that it must appear but we're not worried about the order.

The first grouping is (?=.*{8,}). This checks if there are at least 8 characters in the string. The next grouping (?=.*[0-9]) means "any alphanumeric character can happen zero or more times, then any digit can happen". So this checks if there is at least one number in the string. But since the string isn't captured, that one digit can appear anywhere in the string. The next groupings (?=.*[a-z]) and (?=.*[A-Z]) are looking for the lower case and upper case letter accordingly anywhere in the string.

Finally, we will consider regular expression that validates an email address:

```
<?php
      $email = "firstname.lastname@aaa.bbb.com";
      $regexp      =      "/^[^0-9][A-z0-9_]+([.][A-z0-9_]+)*[@][A-z0-
9_]+([.][A-z0-9_]+)*[.][A-z]{2,4}$/";
      if (preg_match($regexp, $email)) {
          echo "Email address is valid.";
      } else {
          echo "Email address is <u>not</u> valid.";
      }
?>
```

VALIDATE DOMAIN NAME

```
$url = "http://testdomain.com/";
if   (preg_match('/^(http|https|ftp):\/\/([A-Z0-9][A-Z0-9_-]*(?:\.[A-
Z0-9][A-Z0-9_-]*)+):?(\d+)?\/?/i', $url)) {
    echo "Your url is ok.";
} else {
    echo "Wrong url.";
}
```

HIGHLIGHT A WORD IN THE CONTENT

```
$text = "Sample sentence, regex has become popular in web
programming. Now we learn regex. According to wikipedia, Regular
expressions (abbreviated as regex or regexp, with plural forms
regexes, regexps, or regexen) are written in a formal language that
can be interpreted by a regular expression processor";
$text = preg_replace("/\b(regex)\b/i", '<span
style="background:#5fc9f6">\1</span>', $text);
echo $text;
```

REMOVE REPEATED WORDS

```
$text = "Keep your your head"
$text = preg_replace("/\s(\w+\s)\1/i", "$1", $text);
```

PHP GD LIBRARY

GD is a graphics library that is bundled with the PHP web scripting language, as of version 4.3.0. GD is an open source project, under the stewardship of Thomas Boutell, whose web site Boutell.com contains downloads and documentation on the latest version (2.0.27 as of writing).

GD allows PHP programmers to developer web applications that facilitate generating dynamic images, in the JPEG and PNG formats. These images may be based on user input, database queries or other variable information, which allows for a very powerful and flexible tool for serving up relevant information to end users in a graphical format, e.g. in bar or pie charts.

The GD library is written in C, and is available for Windows and Unix-derivative systems.

The functions provided by GD vary greatly depending upon which major version of GD that you have installed on your system. Versions before 1.6 only supported the GIF format. Version 1.6 and later support JPEG, PNG and WBMP, but *not* GIF, due to a patent on that graphic format .Version 2.x added several new drawing shapes, and it is that version that I will be using in future tutorials on GD.

To give some understanding of GDs power, here are some of its major features:

- The drawing of primitive shapes such as lines, arcs, ellipses, polygons et cetera with variable parameters provided by your PHP code.
- Support for true color images with an alpha channel that allows for transparency in images.
- Support for merging of two images into one composite.
- Full text support, with built-in fonts and support for anti-aliased TrueType fonts via the FreeType library.
- Support for anti-aliased primitives.
- Optional resampling of resized images.

Apart from the features mentioned above, there are likely to be many more additions to come from this very active project. While there are other tools available for generating images dynamically with PHP, GD is the most widely used and provides the best performance.

IMAGECOLORALLOCATE()

Allocate a color for an image. Returns a color identifier representing the color composed of the given RGB components. imagecolorallocate() must be called to create each color that is to be used in the

image represented by image. The first call to imagecolorallocate() fills the background color in palette-based images - images created using imagecreate().

imagecolorallocate (resource $image , int $red , int $green , int $blue)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

red Value of red component. The parameter is integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

green Value of green component. The parameter is integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

blue Value of blue component. The parameter is integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values A color identifier or FALSE if the allocation failed.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(150, 150);
  imagecolorallocate($png_image, 15, 142, 210);
?>
```

IMAGECOPYRESAMPLED()

Copy and resize part of an image with resampling. imagecopyresampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity.

In other words, imagecopyresampled() will take a rectangular area from src_image of width src_w and height src_h at position (src_x,src_y) and place it in a rectangular area of dst_image of width dst_w and height dst_h at position (dst_x,dst_y).

If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if dst_image is the same as src_image) but if the regions overlap the results will be unpredictable.

imagecopyresampled (resource $dst_image , resource $src_image , int $dst_x , int $dst_y , int $src_x , int $src_y , int $dst_w , int $dst_h , int $src_w , int $src_h)

dst_image Destination image link resource.

src_image Source image link resource.

dst_x x-coordinate of destination point.

dst_y y-coordinate of destination point.

src_x x-coordinate of source point.

src_y y-coordinate of source point.

dst_w Destination width.

dst_h Destination height.

src_w Source width.

src_h Source height.

Return Values Returns TRUE on success or FALSE on failure.

```
<?php
```

```
  header('Content-type: image/jpeg');
  $file = 'image.jpg';

  $new_width = 150;
  $new_height = 150;
  list($old_width, $old_height) = getimagesize($file);

  $new_image = imagecreatetruecolor($new_width, $new_height);
  $old_image = imagecreatefromjpeg($file);

  imagecopyresampled($new_image, $old_image, 0, 0, 0, 0, $new_width,
$new_height, $old_width, $old_height);

  imagejpeg($new_image);
  imagedestroy($old_image);
  imagedestroy($new_image);
?>
```

IMAGECREATE()

Create a new palette based image. Returns an image identifier representing a blank image of specified size. We recommend the use of imagecreatetruecolor().

imagecreate (int $width , int $height)

width The image width.

height The image height.

Return Values: Returns an image resource identifier on success, FALSE on errors.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(150, 150);
?>
```

IMAGEDASHEDLINE()

Draw a dashed line. This function is deprecated. Use combination of imagesetstyle() and imageline() instead.

imagedashedline (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

x1 Upper left x coordinate.

y1 Upper left y coordinate 0, 0 is the top left corner of the image.

x2 Bottom right x coordinate.

y2 Bottom right y coordinate.

color The fill color. A color identifier created with imagecolorallocate().

Return Values Always returns true

```
<?php
```

```
  header('Content-type: image/png');
  $png_image = imagecreate(150, 150);

  imagecolorallocate($png_image, 15, 142, 210);
  imagesetthickness($png_image, 5);
  $black = imagecolorallocate($png_image, 0, 0, 0);
  $x = 0;
  $y = 0;
  $w = imagesx($png_image) - 1;
  $z = imagesy($png_image) - 1;
  imageline($png_image, $x, $y, $x, $y+$z, $black);
  imageline($png_image, $x, $y, $x+$w, $y, $black);
  imageline($png_image, $x+$w, $y, $x+$w, $y+$z, $black);
  imageline($png_image, $x, $y+$z, $x+$w ,$y+$z, $black);
  imagedashedline($png_image, 150, 0, 0, 150, $black);
  imagepng($png_image);
  imagedestroy($png_image);
?>
```

IMAGEDESTROY()

Destroy an image. imagedestroy() frees any memory associated with image image.

imagedestroy (resource $image)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values Returns TRUE on success or FALSE on failure.

```
<?php
      header('Content-type: image/png');
      $png_image = imagecreate(150, 150);
      imagecolorallocate($png_image, 15, 142, 210);
      imagepng($png_image);
      imagedestroy($png_image);
?>
```

IMAGEFILLEDELLIPSE()

Draw a filled ellipse. Draws an ellipse centered at the specified coordinate on the given image.

imagefilledellipse (resource $image , int $cx , int $cy , int $width , int $height , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

cx x-coordinate of the center.

cy y-coordinate of the center.

width The ellipse width.

height The ellipse height.

color The fill color. A color identifier created with imagecolorallocate().

Return Values Returns TRUE on success or FALSE on failure.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(300, 300);
  $grey = imagecolorallocate($png_image, 229, 229, 229);
  $green = imagecolorallocate($png_image, 128, 204, 204);
  imagefilltoborder($png_image, 0, 0, $grey, $grey);
  imagefilledellipse ($png_image, 50, 150, 75, 75, $green);       //
CIRCLE
  imagefilledellipse ($png_image, 200, 150, 150, 75, $green);     //
ELLIPSE
  imagepng($png_image);
  imagedestroy($png_image);
?>
```

IMAGEFILLEDPOLYGON()

Draw a filled polygon. imagefilledpolygon() creates a filled polygon in the given image.

imagefilledpolygon (resource $image , array $points , int $num_points , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

points An array containing the x and y coordinates of the polygons vertices consecutively.

num_points Total number of vertices, which must be at least 3.

color A color identifier created with imagecolorallocate().

Return Values Returns TRUE on success or FALSE on failure.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(300, 300);
  $grey = imagecolorallocate($png_image, 229, 229, 229);
  $green = imagecolorallocate($png_image, 128, 204, 204);
  imagefilltoborder($png_image, 0, 0, $grey, $grey);
  $poly_points = array(150, 200, 100, 280, 200, 280);
  imagefilledpolygon ($png_image, $poly_points, 3, $green);       //
POLYGON
  imagepng($png_image);
  imagedestroy($png_image);
?>
```

IMAGEFILLEDRECTANGLE()

Draw a filled rectangle. Creates a rectangle filled with color in the given image starting at point 1 and ending at point 2. 0, 0 is the top left corner of the image.

imagefilledrectangle (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

x1 x-coordinate for point 1.
y1 y-coordinate for point 1.
x2 x-coordinate for point 2.
y2 y-coordinate for point 2.
color The fill color. A color identifier created with imagecolorallocate().
Return Values Returns TRUE on success or FALSE on failure.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(300, 300);
  $grey = imagecolorallocate($png_image, 229, 229, 229);
  $green = imagecolorallocate($png_image, 128, 204, 204);
  imagefilltoborder($png_image, 0, 0, $grey, $grey);
  imagefilledrectangle ($png_image, 20, 20, 80, 80, $green);      //
SQUARE
  imagefilledrectangle ($png_image, 100, 20, 280, 80, $green);    //
RECTANGLE
  imagepng($png_image);
  imagedestroy($png_image);
?>
```

IMAGEFILLTOBORDER()

Flood fill to specific color. imagefilltoborder() performs a flood fill whose border color is defined by border. The starting point for the fill is x, y (top left is 0, 0) and the region is filled with color color.

imagefilltoborder (resource $image , int $x , int $y , int $border , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().
x x-coordinate of start.
y y-coordinate of start.
border The border color. A color identifier created with imagecolorallocate().
color The fill color. A color identifier created with imagecolorallocate().
Return Values Returns TRUE on success or FALSE on failure.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(300, 300);
  $grey = imagecolorallocate($png_image, 199, 199, 199);
  $green = imagecolorallocate($png_image, 128, 204, 204);
  imagefilltoborder($png_image, 0, 0, $grey, $grey);
  imagepng($png_image);
  imagedestroy($png_image);
```

```
?>
```

IMAGEFILTER()

Applies a filter to an image. imagefilter() applies the given filter filtertype on the image.

imagefilter (resource $image , int $filtertype [, int $arg1 [, int $arg2 [, int $arg3 [, int $arg4]]]])

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

filtertype filtertype can be one of the following:

IMG_FILTER_GRAYSCALE: Converts the image into grayscale.
IMG_FILTER_BRIGHTNESS: Changes the brightness of the image. Use arg1 to set the level of brightness.
IMG_FILTER_CONTRAST: Changes the contrast of the image. Use arg1 to set the level of contrast.
IMG_FILTER_EMBOSS: Embosses the image.

arg1 IMG_FILTER_BRIGHTNESS: Brightness level. IMG_FILTER_CONTRAST: Contrast level. IMG_FILTER_COLORIZE: Value of red component. IMG_FILTER_SMOOTH: Smoothness level. IMG_FILTER_PIXELATE: Block size in pixels.

arg2 IMG_FILTER_COLORIZE: Value of green component. IMG_FILTER_PIXELATE: Whether to use advanced pixelation effect or not (defaults to FALSE).

arg3 IMG_FILTER_COLORIZE: Value of blue component.

arg4 IMG_FILTER_COLORIZE: Alpha channel, A value between 0 and 127. 0 indicates completely opaque while 127 indicates completely transparent.

Return Values Returns TRUE on success or FALSE on failure.

```
<?php
  header('Content-type: image/jpeg');
  $jpg_image = imagecreatefromjpeg('image.jpg');
  imagefilter($jpg_image, IMG_FILTER_GRAYSCALE);
  imagejpeg($jpg_image);
  imagedestroy($jpg_image);
?>
```

IMAGELINE()

Draw a line. Draws a line between the two given points.

imageline (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

x1 x-coordinate for first point.

y1 y-coordinate for first point.

x2 x-coordinate for second point.

y2 y-coordinate for second point.

color The line color. A color identifier created with imagecolorallocate().
Return Values Returns TRUE on success or FALSE on failure.

```
<?php
      header('Content-type: image/png');
      $png_image = imagecreate(150, 150);
      imagecolorallocate($png_image, 15, 142, 210);
      $black = imagecolorallocate($png_image, 0, 0, 0);
      imageline($png_image, 0, 0, 150, 150, $black);
      imagepng($png_image);
      imagedestroy($png_image);
?>
```

IMAGESETTHICKNESS()

Set the thickness for line drawing. imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to thickness pixels.

imagesetthickness (resource $image , int $thickness)
image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().
thickness Thickness, in pixels.
Return Values Returns TRUE on success or FALSE on failure.

```
<?php
      header('Content-type: image/png');
      $png_image = imagecreate(150, 150);
      imagecolorallocate($png_image, 105, 142, 210);
      imagesetthickness($png_image, 5);
      $black = imagecolorallocate($png_image, 0, 0, 0);
      $x = 0;
      $y = 0;
      $w = imagesx($png_image) - 1;
      $z = imagesy($png_image) - 1;
      imageline($png_image, $x, $y, $x, $y+$z, $black);
      imageline($png_image, $x, $y, $x+$w, $y, $black);
      imageline($png_image, $x+$w, $y, $x+$w, $y+$z, $black);
      imageline($png_image, $x, $y+$z, $x+$w ,$y+$z, $black);
      imagepng($png_image);
      imagedestroy($png_image);
?>
```

IMAGETTFTEXT()

Write text to the image using TrueType fonts. Writes the given text into the image using TrueType fonts.

imagettftext (resource $image , float $size , float $angle , int $x , int $y , int $color , string $fontfile , string $text)
image An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

size The font size. Depending on your version of GD, this should be specified as the pixel size (GD1) or point size (GD2).
angle The angle in degrees, with 0 degrees being left-to-right reading text. Higher values represent a counter-clockwise rotation. For example, a value of 90 would result in bottom-to-top reading text.
x The coordinates given by x and y will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where x and y define the upper-left corner of the first character. For example, "top left" is 0, 0.
y The y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.
color The color index. Using the negative of a color index has the effect of turning off antialiasing. See imagecolorallocate().
fontfile The path to the TrueType font you wish to use. Depending on which version of the GD library PHP is using, when fontfile does not begin with a leading / then .ttf will be appended to the filename and the library will attempt to search for that filename along a library-defined font path.
text The text string in UTF-8 encoding.
Return Values Returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is lower left, lower right, upper right, upper left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontally. Returns FALSE on error.

```
<?php
  //Set the Content Type
  header('Content-type: image/jpeg');

  // Create Image From Existing File
  $jpg_image = imagecreatefromjpeg('image.jpg');

  // Allocate A Color For The Text
  $white = imagecolorallocate($jpg_image, 255, 255, 255);

  // Set Path to Font File
  $font_path = 'CALIBRI.TTF';

  // Set Text to Be Printed On Image
  $text = "This is a Image!";

  // Print Text On Image
  imagettftext($jpg_image, 25, 0, 75, 300, $white, $font_path,
$text);
```

```
  // Send Image to Browser
  imagejpeg($jpg_image);

  // Clear Memory
  imagedestroy($jpg_image);
?>
```

UPLOADING FILE

Handling file uploads isn't difficult, but there are a handful of small details that must be correct or else the upload will fail. First, you need to ensure PHP is configured to allow uploads. Check your php.ini file and verify the file_uploads directive is set On.

```
file_uploads = On
```

Uploaded files are first stored in a temporary directory (don't worry… your PHP script can move the files to a more permanent location afterward). By default, the initial location is the system's default temporary directory. You can specify a different directory using the upload_tmp_dir directive in php.ini. Regardless, you should verify the PHP process has the proper privileges to write to whichever directory is used.

```
upload_tmp_dir = "/tmp"
```

After you're certain the configuration allows the server to accept uploaded files, you can focus your attention on the HTML details. As with most other server-side interactions from HTML, uploading files make use of forms. It is imperative that your <form> element uses the POST method and has an enctype attribute set to multipart/form-data.

```
<form action="upload.php" method="post" enctype="multipart/form-
data">
```

Scripting the Upload Process

You can probably guess the workflow file uploads go through based on your own experiences and the requirement checks I've just mentioned.

- A visitor views an HTML page with a form specifically written to support file uploads
- The visitor provides the file he wants to upload and submits the form
- The browser encodes the file and sends it as part of the POST request it makes to the server
- PHP receives the form submission, decodes the file and saves it in a temporary location on the server
- The PHP script responsible for handling the form post verifies the file and processes it in some manner, often moving it from its temporary location to a more permanent home
- Adding support for file uploads requires you to create an HTML form to be presented to the user and a PHP script to take care of the uploaded file on the server.

HTML forms provide the interface through which a user initiates a file upload. Remember, the <form> element must have its method attribute set to post and its enctype attribute set to multipart/form-data. A file <input> element provides the a field used to specify the file that will be upload. Like any other form element, it is important you provide a name attribute so you can reference it in the PHP script that processes the form.

Here's what markup for a basic file upload form looks like:

```
<form action="upload.php" method="post" enctype="multipart/form-
data">
<input type="file" name="myFile">
<br>
<input type="submit" value="Upload">
</form>
```

It's worth noting that different browsers will render the file field differently. IE, Firefox, and Opera display it as a text field with a button next to it labeled "Browse" or "Choose." Safari renders it just as button labeled "Choose File." This isn't a problem most of the time since users are accustomed to how the field renders in their browser of choice and know how to use it. Occasionally, however, you will be faced with a client or designer who is adamant on presenting it a certain way. The amount of CSS and JavaScript that can be applied to a file field is extremely limited because of security reasons imposed by the browsers. Styling the file field can be difficult.

Information about the file upload is made available with the multidimensional $_FILES array. This array is indexed by the names assigned to the file fields in the HTML form, just as how $_GET and $_POST are indexed. Each file's array then contains the following indexes

$_FILES["myFile"]["name"] stores the original filename from the client
$_FILES["myFile"]["type"] stores the file's mime-type
$_FILES["myFile"]["size"] stores the file's size (in bytes)
$_FILES["myFile"]["tmp_name"] stores the name of the temporary file
$_FILES["myFile"]["error"] stores any error code resulting from the transfer

The move_uploaded_file() function moves an uploaded file from its temporary to permanent location. You should always use move_uploaded_file() over functions like copy() and rename() for this purpose because it performs additional checks to ensure the file was indeed uploaded by the HTTP POST request.

If you plan on saving a file with the original filename provided by the user, it's a good idea to make sure it's safe to do so. The filename should not contain any characters that can affect the destination path, such as a slash. The name shouldn't cause the file to overwrite an existing file with the same name, either (unless that's what your application is designed to do). I ensure a safe filename by replacing any characters with an underscore that aren't a letter, number, or a member of a very

restricted set of punctuation, and then append an incrementing number when a file by that name already exists.

```
<?php
      define("UPLOAD_DIR", "/srv/www/uploads/");
      if (!empty($_FILES["myFile"])) {
            $myFile = $_FILES["myFile"];

            if ($myFile["error"] !== UPLOAD_ERR_OK) {
                  echo "<p>An error occurred.</p>";
                  exit;
            }

            // ensure a safe filename
            $name = preg_replace("/[^A-Z0-9._-]/i", "_",
$myFile["name"]);

            // don't overwrite an existing file
            $i = 0;
            $parts = pathinfo($name);
            while (file_exists(UPLOAD_DIR . $name)) {
                  $i++;
                  $name = $parts["filename"] . "-" . $i . "." .
$parts["extension"];
            }

            // preserve file from temporary directory
            $success = move_uploaded_file($myFile["tmp_name"],
                  UPLOAD_DIR . $name);
            if (!$success) {
                  echo "<p>Unable to save file.</p>";
                  exit;
            }

            // set proper permissions on the new file
            chmod(UPLOAD_DIR . $name, 0644);
      }
?>
```

The code first makes sure the file uploaded without any errors. It then determines a safe filename as I just described, and then moves the file to its final directory using move_uploaded_file(). Finally, there is a call to chmod() to make sure sane access permissions are set on the new file.

Security Considerations

Most of us wouldn't let complete strangers store random files on our personal computers, and yet that is exactly what you are doing when you allow file uploads in our application. You may intend for a user to upload a picture of him for a profile page, but what if he tries to upload a specially-crafted, virus-laden executable instead? I'd like to share a few steps that you can take to minimize the security risks inherent in allowing file uploads.

One is to verify the type of the uploaded file is what it should be. Relying on either the value of $_FILES["myFile"]["type"] or on the filename's extension isn't secure because both can easily be spoofed. Rather, use a function like exif_imagetype() to examine the contents of the file and determine if it is indeed a GIF, JPEG, or one of several other supported image formats. If exif_imagetype() isn't available (the function requires the Exif extension to be enabled), then you can use getimagesize(). The array returned by getimagesize() will contain the image type if it is recognized.

```
<?php
      // verify the file is a GIF, JPEG, or PNG
      $fileType = exif_imagetype($_FILES["myFile"]["tmp_name"]);
      $allowed = array(IMAGETYPE_GIF, IMAGETYPE_JPEG,
IMAGETYPE_PNG);
      if (!in_array($fileType, $allowed)) {
            // file type is not permitted
    ...
```

Another step you can take is to impose hard limits on the total size of the POST request and the number of files that can be uploaded. To do so, specify an appropriate value for the upload_max_size, post_max_size, and max_file_uploads directives in php.ini. The upload_max_size directive specifies the maximum size a file upload can be. In addition to the size of the upload, you can limit the size of the entire POST request with the post_max_size directive. max_file_uploads is a newer directive (added in version 5.2.12) which limits the number of file uploads. These three directives help protect your site against attacks that try to disrupt its availability by causing heavy network traffic or system load.

```
post_max_size = 8M
upload_max_size = 2M
max_file_uploads = 20
```

A third step you can take to minimize your risk is to scan uploaded files with a virus scanner. This is vitally important in this day and age of widespread viruses and malware, especially if your site later makes uploaded files available for download by other individuals, such as with attachments in a web-based email client or a (legal) file-sharing site. There is a PHP extension that provides access to ClamAV, but of course you can invoke ClamAV's command-line utility in much the same way I demonstrated for file.

```
<?php
      exec("clamscan --stdout " . $_FILES["myFile"]["tmp_name"], $out, $return);
```

```
if ($return) {
        // file is infected
        ...
```

SENDING MAIL USING MAIL()

mail (string $to , string $subject , string $message [, string $additional_headers])

to Receiver, or receivers of the mail. The formatting of this string must comply with » RFC 2822. Some examples are:

user@example.com
user@example.com, anotheruser@example.com
User <user@example.com>
User <user@example.com>,Another User<anotheruser@example.com>

subject Subject of the email to be sent.

message Message to be sent. Each line should be separated with a CRLF (\r\n). Lines should not be larger than 70 characters.

additional_headers (optional) String to be inserted at the end of the email header. This is typically used to add extra headers (From, Cc, and Bcc). Multiple extra headers should be separated with a CRLF (\r\n).

Return Values Returns TRUE if the mail was successfully accepted for delivery, FALSE otherwise. It is important to note that just because the mail was accepted for delivery, it does NOT mean the mail will actually reach the intended destination.

```
<?php
      $to = "recipient@example.com";
      $subject = "Hi!";
      $body = "Hi,\n\nHow are you?";
      if (mail($to, $subject, $body)) {
            echo("<p>Email successfully sent!</p>");
      } else {
            echo("<p>Email delivery failed…</p>");
      }
?>
```

```
<form name="contactform" method="post" action="send_form_email.php">
      <table width="450px">
            <tr>
                  <td valign="top"><label for="first_name">First Name
*</label></td>
                  <td valign="top"><input  type="text"
name="first_name" maxlength="50" size="30"></td>
            </tr>
            <tr>
```

```
                <td valign="top""><label for="last_name">Last Name
*</label></td>
                <td valign="top"><input  type="text"
name="last_name" maxlength="50" size="30"></td>
          </tr>
          <tr>
                <td valign="top"><label for="email">Email Address
*</label></td>
                <td valign="top"><input  type="text" name="email"
maxlength="80" size="30"></td>
          </tr>
          <tr>
                <td valign="top"><label for="telephone">Telephone
Number</label></td>
                <td valign="top"><input  type="text"
name="telephone" maxlength="30" size="30"></td>
          </tr>
          <tr>
                <td valign="top"><label for="comments">Comments
*</label></td>
                <td valign="top"><textarea  name="comments"
maxlength="1000" cols="25" rows="6"></textarea></td>
          </tr>
          <tr>
                <td colspan="2" style="text-align:center"><input
type="submit" value="Submit"></td>
          </tr>
     </table>
</form>
<?php
     if(isset($_POST['email'])) {
          // EDIT THE 2 LINES BELOW AS REQUIRED
          $email_to = "you@yourdomain.com";
          $email_subject = "Your email subject line";
          function died($error) {
                // your error code can go here
                echo "We are very sorry, but there were error(s)
found with the form you submitted. ";
                echo "These errors appear below.<br /><br />";
                echo $error."<br /><br />";
                echo "Please go back and fix these errors.<br /><br
/>";
                die();
          }
          // validation expected data exists
          if(!isset($_POST['first_name']) ||
                !isset($_POST['last_name']) ||
```

```
                !isset($_POST['email']) ||
                !isset($_POST['telephone']) ||
                !isset($_POST['comments'])) {
                     die('We are sorry, but there appears to be a
problem with the form you submitted.');
           }

           $first_name = $_POST['first_name']; // required
           $last_name = $_POST['last_name']; // required
           $email_from = $_POST['email']; // required
           $telephone = $_POST['telephone']; // not required
           $comments = $_POST['comments']; // required

           $error_message = "";
           $email_exp = '/^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+\.[A-Za-
z]{2,4}$/';
           if(!preg_match($email_exp,$email_from)) {
                $error_message .= 'The Email Address you entered
does not appear to be valid.<br />';
           }
           $string_exp = "/^[A-Za-z .'-]+$/";
           if(!preg_match($string_exp,$first_name)) {
                $error_message .= 'The First Name you entered does
not appear to be valid.<br />';
           }
           if(!preg_match($string_exp,$last_name)) {
                $error_message .= 'The Last Name you entered does
not appear to be valid.<br />';
           }
           if(strlen($comments) < 2) {
                $error_message .= 'The Comments you entered do not
appear to be valid.<br />';
           }
           if(strlen($error_message) > 0) {
                die($error_message);
           }
           $email_message = "Form details below.\n\n";
           function clean_string($string) {
                $bad = array("content-
type","bcc:","to:","cc:","href");
                return str_replace($bad,"",$string);
           }
           $email_message .= "First Name:
".clean_string($first_name)."\n";
           $email_message .= "Last Name:
".clean_string($last_name)."\n";
```

```
            $email_message .= "Email:
".clean_string($email_from)."\n";
            $email_message .= "Telephone:
".clean_string($telephone)."\n";
            $email_message .= "Comments:
".clean_string($comments)."\n";

            // create email headers
                  $headers = 'From: '.$email_from."\r\n".
                  'Reply-To: '.$email_from."\r\n" .
                  'X-Mailer: PHP/' . phpversion();
                  @mail($email_to, $email_subject, $email_message,
$headers);
      }
?>
```

Sending Mail Using SMTP()

Fortunately, overcoming PHP's built-in shortcomings is not difficult either, complicated or painful. For most email uses, the free PEAR Mail package offers all the power and flexibility needed, and it authenticates with your desired outgoing mail server. For enhanced security, encrypted SSL connections are supported for sending mail using PEAR Mail as well.

Send Email from a PHP Script Using SMTP Authentication

To connect to an outgoing SMTP server from a PHP script using SMTP authentication and send an email:

- Make sure the PEAR Mail package is installed.Typically, in particular with PHP 4 or later, this will have already been done for you. Just give it a try.
- Adapt the example below for your needs. Make sure you change the following variables at least:

  from: the email address from which you want the message to be sent.
  to: the recipient's email address and name.
  host: your outgoing SMTP server name.
  username: the SMTP user name (typically the same as the user name used to retrieve mail).
  password: the password for SMTP authentication.

```
<?php
      require_once "Mail.php";
      $from = "Sender <sender@example.com>";
      $to = "Recipient <recipient@example.com>";
      $subject = "Hi!";
      $body = "Hi,\n\nHow are you?";
      $host = "mail.example.com";
      $username = "smtp_username";
      $password = "smtp_password";
```

```
      $headers = array ('From' => $from,
            'To' => $to,
            'Subject' => $subject);
            $smtp = Mail::factory('smtp',
            array ('host' => $host,
            'auth' => true,
            'username' => $username,
            'password' => $password));

      $mail = $smtp->send($to, $headers, $body);

      if (PEAR::isError($mail)) {
            echo("<p>" . $mail->getMessage() . "</p>");
      } else {
            echo("<p>Message successfully sent!</p>");
      }
?>
```

Ajax

AJAX is a web development technique (not a new programming language) that allows you to create faster and more user friendly web applications. AJAX stands for Asynchronous JavaScript and XML. It changes the way a web page is processed on client side using server side information.

Practically, AJAX combines many existing web technologies, such as JavaScript, CSS, DOM, PHP and more. JavaScript communicates directly with the server through XMLHttpRequest API and this way the information exchanged between client and server is "invisible", because the content of the page will update for a given user action without the need to reload that page.

The asynchronous behavior of JavaScript resides in the fact that data requested from server is loaded on client side without generation of interferences with the actual page display. The applications made with AJAX will not depend on web server software and are fast because the time needed to exchange certain quantity of information with the server is lower than the time necessary to reload the whole page. As a consequence the application response time to users' requests are decreased with AJAX.

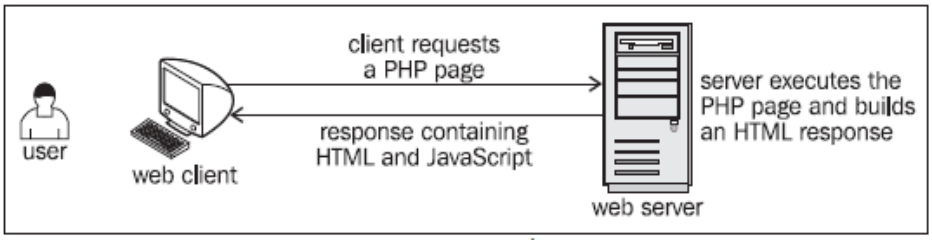
AJAX has many other advantages. It can be employed to create more user friendly forms which could be validated in real time, rich web applications with appearance and functionality similar to desktop applications and more. The applications made with AJAX can be used on any operating system in any modern web browser that provides support for XMLHttpRequest object. Generally the quantity of information exchanged with a web server is small.

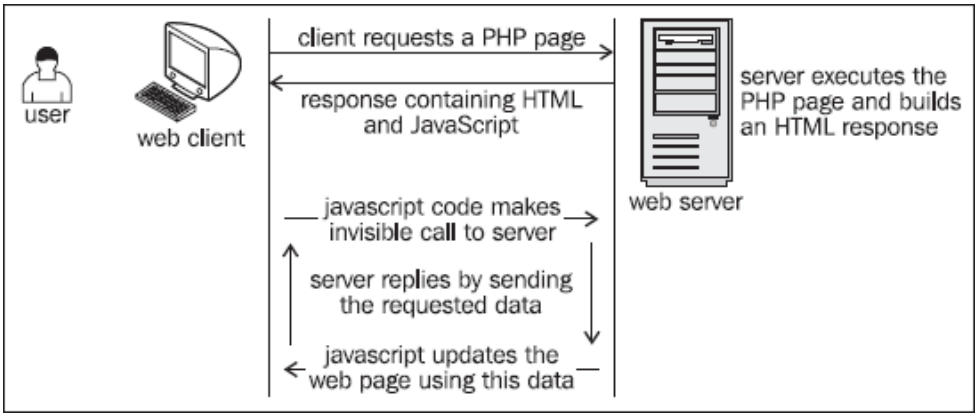
Due to the small quantity of information downloaded from webserver in order to update only a part of a webpage, without reloading it, the bandwidth consumption is reduced while the page loading time is optimized. In order to begin the development of web applications with AJAX, you do not have

to learn new programming languages, because it is based on actual web standard programming languages.

AJAX LIFE CYCLE

1. The webpage is visited by internet users simply by clicking the targeted link or searching an URL. Thereafter the page gets initialized and loaded. In order to handle the user input callbacks are present in the system which acts after this.
2. When a key is pressed it's called an event which is a query or click in common words. On such events browser sends the information request to the server which in turn responds to give back required information as bites.
3. The response of the server is translated through callback process to make viewable to the user.
4. The callback function then updates the DOM objects and JavaScript variables if present.
5. Generally the process of requesting information and giving callbacks are free to server to increase convenience and speed.
6. Some AJAX applications are short term and terminated once the request is completed.





HTTP REQUEST AND RESPONSE FUNDAMENTALS

In order to understand exactly how Ajax concepts are put together, it is important to know how a web site processes a request and receives a response from a web server. The current standard that browsers use to acquire information from a web server is the HTTP (Hypertext Transfer Protocol) method (currently at version HTTP/1.1). This is the means a web browser uses to send out a request from a web site and then receive a response from the web server that is currently in charge of returning the response. Once a request has been received, the server then decides what response to return. There are many different response codes.

Code	Description
200 OK	This response code is returned if the document or file in question is found and served correctly

304 Not Modified	This response code is returned if a browser has indicated that it has a local, cached copy, and the server's copy has not changed from this cached copy.
401 Unauthorized	This response code is generated if the request in question requires authorization to access the requested document
403 Forbidden	This response code is returned if the requested document does not have proper permissions to be accessed by the requestor.
404 Not Found	This response code is sent back if the file that is attempting to be accessed could not be found (e.g., if it doesn't exist).
500 Internal Server Error	This code will be returned if the server that is being contacted has a problem.
503 Service Unavailable	This response code is generated if the server is too overwhelmed to handle the request.

Various forms request methods available. A few of them, like GET and POST, will probably sound quite familiar.

The XMLHttpRequest Object

In order to make a request to the server through Ajax, an object must be created.

XMLHTTPREQUEST METHODS

Once an instance of the XMLHttpRequest object has been created, there are a number of methods available to the user.

Method	Description
open("method","URL","async","username","psw d")	Specifies the different attributes necessary to make a connection tothe server; allows you to make selections such as GET or POST (moreon that later), whether to connect asynchronously, and which URLto connect to
send("content")	Sends the current request
abort()	Cancels the current request

XMLHTTPREQUEST PROPERTIES

Of course, any object has a complete set of properties that can be used and manipulated in order for it works to its fullest.

Method	Description
onreadystatechange	Used as an event handler for events that trigger upon state changes
readyState	Contains the current state of the object (0: uninitialized, 1: loading, 2: loaded, 3: interactive, 4: complete)
responseText	Returns the response in string format
status	Returns the status of the request in numerical format (regular page errors are returned, such as the number 404, which refers to a not found error)
statusText	Returns the status of the request, but in string format (e.g., a 404 error would return the string Not Found)

```
Create a broswer variable for Ajax
<script language="javascript">
      var xmlhttp = false;
```

```
      //Check if we are using IE.
      try {
            //If the Javascript version is greater than 5.
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            alert ("You are using Microsoft Internet Explorer.");
      }
      catch (e) {
            //If not, then use the older active x object.
            try {
                  //If we are using Internet Explorer.
                  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                  alert ("You are using Microsoft Internet Explorer");
            }
            catch (E) {
                  //Else we must be using a non-IE browser.
                  xmlhttp = false;
            }
      }
</script>
```

Calling the content from different file

Main.html

```
<script language="javascript">
      var xmlhttp = false;
      if(window.XMLHttpRequest){
            xmlhttp = new XMLHttpRequest();
      }
      else if(window.ActiveXObject){
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
      }

      function makerequest(serverPage, objID) {
            var obj = document.getElementById(objID);
            xmlhttp.open("GET", serverPage);
            xmlhttp.onreadystatechange = function() {
                  if (xmlhttp.readyState == 4 && xmlhttp.status ==
200) {
                        obj.innerHTML = xmlhttp.responseText;
                  }
            }
            xmlhttp.send(null);
      }
</script>
<body onload="makerequest ('content1.html','hw')">
      <div align="center">
            <h1>My Webpage</h1>
```

```
            <a href="content1.html"
onclick="makerequest('content1.html','hw');return false;"> Page
1</a> |
            <a
href="content2.html"onclick="makerequest('content2.html','hw');retur
n false;">Page 2</a> |
            <a href="content3.html"
onclick="makerequest('content3.html','hw'); return false;">Page
3</a> |
            <a href="content4.html"
onclick="makerequest('content4.html','hw'); return false;">
Page 4</a>
            <div id="hw"></div>
      </div>
      <br><br><br>
      <center><b>The content of main page will not change or
moved</b></center>
</body>
```

Content1.html, Content2.html, Content3.html, Content4.html

```
<html>
      <body>
            The content for page [1][2][3][4]
      </body>
</html>
```

Database program to check user is available or not& strength of Password *[Table name: user | Fields: nm & pass]*

Functions.js

```
var xmlhttp = false;
if(window.XMLHttpRequest){
      xmlhttp = new XMLHttpRequest();
}
else if(window.ActiveXObject){
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
function checkdata(serverPage,objID,val) {
      var obj = document.getElementById(objID);
      serverPage = serverPage + "?val=" + val
      xmlhttp.open("GET", serverPage);
      xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                  obj.innerHTML = xmlhttp.responseText;
            }
      }
            xmlhttp.send(null);
```

```
}
function check(form)
{
      len = form.elements.length;
      for(i=0;i<len;i++){
            if(form.elements[i].value===""){
                  alert("Empty not allowed");
                  form.elements[i].focus();
                  return false;
            }
      }
      if(form.elements[1].value.length<10)  {
            alert("Password should atleast 10 character");
            form.elements[1].focus();
            return false;
      }
}
```

Main.htm

```
<html>
      <head>
            <script type="text/javascript"
src="functions.js"></script>
            <link rel="stylesheet" type="text/css" href="style.css"
/>
      </head>
      <body>
            <div>
                  <form method="post" action="save.php"
onsubmit="return check(this)">
                  Enter user name <input type="text" name="usrnm"
onblur="checkdata('user.php','usr',this.value)"><div
id="usr"></div><br>
                  Password <input type="password" name="pass"
onblur="checkdata('password.php','pss',this.value)"><div
id="pss"></div><br>
                  <input type="submit" value="Save">
                  </form>
            </div>
      </body>
</html>
```

Password.php

```
<?php
      $password = $_GET[val];
      if (preg_match("/^.*(?=.{10,})([a-z|A-Z]{6})([0-9]{4})$/",
$password)) {
```

```
          echo "<font color='green' size='-1'>Your password is
strong</font>.";
      } else {
          echo "<font color='red' size='-1'>Your password is
weak.</font>";
      }
?>
```

Save.php

```
<?php
      $link = mysql_connect("localhost","root") or die("Connection
failed");
      $db_selected = mysql_select_db("mydb");
      if (!$db_selected) {
            die ('Database failed to open : ' . mysql_error());
      }
      $s = "insert into user
values('$_POST[usrnm]','$_POST[pass]')";
      if(mysql_query($s)){
            $dis =  "<font color='red' size='-1'>Record
Saved</font>";
      }
      else{
            $dis =  "<font color='green' size='-1'>Record Not
Saved</font>";
      }
      echo $dis;
?>
```

User.php

```
<?php
      $link = mysql_connect("localhost","root") or die("Connection
failed");
      $db_selected = mysql_select_db("mydb");
      if (!$db_selected) {
            die ('Database failed to open : ' . mysql_error());
      }
      $s = "select * from user where nm='$_GET[val]'";
      $result = mysql_query($s);
      if(mysql_num_rows($result)>0){
            $dis =  "<font color='red' size='-1'>Not
Available</font>";
      }
      else{
            $dis =  "<font color='green' size='-1'>Available</font>";
      }
```

```
      echo $dis;
?>
```

INTRODUCTION OF SQL

The MySQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a trademark of MySQL AB. MySQL AB is the company of the MySQL founders and main developers. MySQL AB was originally established in Sweden by David Axmark, Allan Larsson, and Michael Monty Widenius.The MySQL software is Dual Licensed. Users can choose to use the MySQL software as an Open Source/Free Software product under the terms of the GNU General Public License

FEATURES

- Written in C and C++. Tested with a broad range of different compilers.
- Works on many different platforms.
- APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl.
- Fully multi-threaded using kernel threads. This means it can easily use multiple CPUs if available.
- Transactional and non-transactional storage engines.
- Very fast B-tree disk tables (MyISAM) with index compression.
- Relatively easy to add another storage engine. This is useful if you want to add an SQL interface to an in-house database.
- A very fast thread-based memory allocation system.
- Very fast joins using an optimised one-sweep multi-join.
- In-memory hash tables which are used as temporary tables.
- SQL functions are implemented through a highly optimised class library and should be as fast as possible. Usually there isn't any memory allocation at all after query initialisation.
- Available as client/server or embedded (linked) version

BASIC COMMAND OF MYSQL

- Login in mysql

```
MYSQL –u username –p password
```

- Displaying databases

```
SHOW DATABASES
```

- Selecting databases

```
USE databasename
```

- Displaying tables

```
SHOW TABLES
```

- Displaying table record

```
SELECT * FROM tablename
SELECT field1,field2 FROM tablename
SELECT * FROM tablename WHERE field1=value OR/AND field2=value
SELECT * FROM tablename LIMIT recordindex,recordnumbers
```

- Creating databases

```
CREATE DATABASE databasename
```

- Create table

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name(field1
datatype(size)[PRIMARY KEY],field2 datatype(size),…)
```

- Alter table [adding a field first]

```
ALTER TABLE tablename ADD newfieldname datatype(size) FIRST
```

- Alter table [adding a field last]

```
ALTER TABLE tablename ADD newfield datatype(size)
```

- Alter table [adding a field in between]

```
ALTER TABLE tablename ADD newfield datatype(size) AFTER
currentfieldname
```

- Altering the field

```
ALTER TABLE tablename CHANGE fieldname datatype(size)
```

- Dropping the field

```
ALTER TABLE tablename DROP fieldname;
```

- Dropping the table

```
DROP TABLE tablename
```

- Inserting the records in the table

```
INSERT INTO tablename[fields name] VALUES(field1value,
field2value,….)
```

- Updating the records in table

```
UPDATE tablename SET field1name=newvalue,field2name=newvalue where
uniquefield=value
```

- Delete the records in table

```
DELETE FROM tbl_name  [WHERE where_condition]  [ORDER BY ...]
[LIMIT row_count]
```

MySQL Functions

mysql_affected_rows()

The mysql_affected_rows() function returns the number of affected rows in the previous MySQL operation.This function returns the number of affected rows on success, or -1 if the last operation failed.

mysql_affected_rows(connection)

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
      $link = mysql_connect('localhost', 'root', '');
      if (!$link) {
            die('Could not connect: ' . mysql_error());
      }
      mysql_select_db('mydb');
      mysql_query('DELETE FROM mydb WHERE id < 10');
      echo "Records deleted: ".mysql_affected_rows();
?>
```

mysql_close()

The mysql_close() function closes a non-persistent MySQL connection.This function returns TRUE on success, or FALSE on failure.

mysql_close(connection)

connection Optional. Specifies the MySQL connection to close. If not specified, the last connection opened by mysql_connect() is used.

```
<?php
      $link = mysql_connect('localhost', 'root', '');
      if (!$link) {
            die('Could not connect: ' . mysql_error());
      }
      echo 'Connected successfully';
      mysql_close($link);
?>
```

mysql_connect()

The mysql_connect() function opens a non-persistent MySQL connection.This function returns the connection on success, or FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

mysql_connect(server,user,pwd,newlink)

server Optional. Specifies the server to connect to (can also include a port number. e.g. "hostname:port" or a path to a local socket for the localhost). Default value is "localhost:3306"

user Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process

pwd Optional. Specifies the password to log in with. Default is ""

```
<?php
      $link = mysql_connect('localhost', 'root', '');
      if (!$link) {
            die('Could not connect: ' . mysql_error());
      }
      echo 'Connected successfully';
      mysql_close($link);
?>
```

MYSQL_DATA_SEEK()

The mysql_data_seek() function moves the internal row pointer.The internal row pointer is the current row position in a result returned by the mysql_query() function.This function returns TRUE on success, or FALSE on failure.

mysql_data_seek(data,row)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

row Required. Specifies which record to move to. 0 indicates the first record

```
<?php
      $con = mysql_connect("localhost", "root", "");
      $selecteddb = mysql_select_db("mydb");
      $sql = "select * from student_info";
      $result = mysql_query($sql,$con);
      mysql_data_seek($result,0);
      print_r(mysql_fetch_row($result));
      mysql_close($con);
?>
```

MYSQL_DB_NAME()

The mysql_db_name() function finds the database name from a call to the mysql_list_dbs() function.This function returns the database name on success, or FALSE on failure.

mysql_db_name(list,row,field)

list Required. Specifies result pointer from the mysql_list_dbs() function

row Required. Specifies what row-index in the result pointer to return. Starts at zero.

field Optional. Specifies the field name

```
<?php
      mysql_connect("localhost", "root", "");
      $db_list = mysql_list_dbs();
      $i = 0;
      $db_count = mysql_num_rows($db_list);
      while ($i < $db_count)
      {
```

```
            echo mysql_db_name($db_list, $i) . "<br />";
            $i++;
      }
?>
```

MYSQL_DB_QUERY()

Deprecated. Sends a MySQL query. Use mysql_select_db() and mysql_query() instead

```
<?php
      mysql_connect("localhost", "root", "");
      $query = "select * from mydb";
      $result = mysql_db_query ("bcasy", $query);
      echo mysql_num_rows($result);
?>
```

MYSQL_ERRNO()

The mysql_errno() function returns the error number of the last MySQL operation.This function returns 0 (zero) if no error has occurred.

mysql_errno(connection)

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
      $link = mysql_connect('localhost', 'root', '');
      if (!$link) {
            die('Could not connect: ' .mysql_errno()." : :
".mysql_error());
      }
      echo 'Connected successfully';
      mysql_close($link);
?>
```

MYSQL_ERROR()

The mysql_error() function returns the error description of the last MySQL operation.This function returns an empty string ("") if no error occurs.

mysql_error(connection)

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
mysql_connect('localhost', 'root', '') or die('Could not connect: '
. mysql_error());
      echo 'Connected successfully';
      mysql_close($link);
?>
```

MYSQL_FETCH_ARRAY()

The mysql_fetch_array() function returns a row from a recordset as an associative array and/or a numeric array.This function gets a row from the mysql_query() function and returns an array on success, or FALSE on failure or when there are no more rows.

mysql_fetch_array(data,array_type)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

array_type Optional. Specifies what kind of array to return.

MYSQL_ASSOC - Associative array
MYSQL_NUM - Numeric array
MYSQL_BOTH - Default. Both associative and numeric array

```
<?php
	$connection = mysql_connect("localhost","root","");

	if (!$connection) {
	die('PHP Mysql database connection could not connect : ' .
mysql_error());
	}
	else{
		$db_name = "mydb";

		mysql_select_db($db_name, $connection);
		$sql = "select * from student_info;";
		$records = mysql_query($sql, $connection);
			while($rows                                    =
mysql_fetch_array($records,MYSQL_ASSOC)){
				echo "Student Id : " . $rows['id'] . "<br />";
				echo "Student Name : " . $rows['name'] . "<br
/>";
				echo "Student Email : " . $rows['email'] .
"<br />";
				echo "Student Address : " . $rows['address'] .
"<hr />";
			}
	}
	mysql_close($connection);
?>
```

```
<?php
	$connection = mysql_connect("localhost","root","");

	if (!$connection) {
	die('PHP Mysql database connection could not connect : ' .
mysql_error());
	}
```

```
      else{
            $db_name = "mydb";

            mysql_select_db($db_name, $connection);
            $sql = "select * from student_info;";
            $records = mysql_query($sql, $connection);
                  while($rows                                          =
mysql_fetch_array($records,MYSQL_BOTH)){
                        echo "Student Id : " . $rows['id'] . "<br />";
                        echo "Student Name : " . $rows[1] . "<br />";
                        echo "Student Email : " . $rows[2] . "<br />";
                        echo "Student Address : " . $rows['address'] .
"<hr />";
                  }
      }
      mysql_close($connection);
?>
```

```
<?php
      $connection = mysql_connect("localhost","root","");

      if (!$connection) {
      die('PHP Mysql database connection could not connect : ' .
mysql_error());
      }
      else{
            $db_name = "mydb";

            mysql_select_db($db_name, $connection);
            $sql = "select * from student_info;";
            $records = mysql_query($sql, $connection);
                  while($rows                                          =
mysql_fetch_array($records,MYSQL_NUM)){
                        echo "Student Id : " . $rows[0] . "<br />";
                        echo "Student Name : " . $rows[1] . "<br />";
                        echo "Student Email : " . $rows[2] . "<br />";
                        echo "Student Address : " . $rows[3] . "<hr
/>";
                  }
      }
      mysql_close($connection);
?>
```

MYSQL_FETCH_ASSOC()

The mysql_fetch_assoc() function returns a row from a recordset as an associative array.This function gets a row from the mysql_query() function and returns an array on success, or FALSE on failure or when there are no more rows.

mysql_fetch_assoc(data)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
<?php
      $connection = mysql_connect("localhost","root","");

      if (!$connection) {
      die('PHP Mysql database connection could not connect : ' .
mysql_error());
      }
      else{
            $db_name = "mydb";

            mysql_select_db($db_name, $connection);
            $sql = "select * from student_info;";
            $records = mysql_query($sql, $connection);
                  while($rows = mysql_fetch_assoc($records)){
                        echo "Student Id : " . $rows['id'] . "<br />";
                        echo "Student Name : " . $rows['name'] . "<br
/>";
                        echo "Student Email : " . $rows['email'] .
"<br />";
                        echo "Student Address : " . $rows['address'] .
"<hr />";
                  }
      }
      mysql_close($connection);
?>
```

MYSQL_FETCH_FIELD

The mysql_fetch_field() function returns an object containing information of a field from a recordset.This function gets field data from the mysql_query() function and returns an object on success, or FALSE on failure or when there are no more rows.

mysql_fetch_field(data,field_offset)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

field_offset Optional. Specifies which field to start returning. 0 indicates the first field. If this parameter is not set, it will retrieve the next field

Return Value: name - Field name

table - The table the field belongs to

def - Default value for the field

```
max_length - Maximum field length
not_null - 1 if the field cannot be NULL
primary_key - 1 if the field is a primary key
unique_key - 1 if the field is a unique key
multiple_key - 1 if the field is a non-unique key
numeric - 1 if the field is numeric
blob - 1 if the field is a BLOB
type - Field type
unsigned - 1 if the field is unsigned
zerofill - 1 if the field is zero-filled
```

```
<?php
      $connection = mysql_connect("localhost","root","");

      if (!$connection) {
            die('PHP Mysql database connection could not connect : '
. mysql_error());
      }
      else{
      mysql_select_db("mydb", $connection);
      $sql = "SELECT * from student_info";
      $result = mysql_query($sql,$connection);

      while ($table_property = mysql_fetch_field($result)) {
            echo "Field name: " . $table_property->name . "<br />";
            echo "Table name: " . $table_property->table . "<br />";
            echo "Default value: " . $table_property->def . "<br />";
            echo "Max length: " . $table_property->max_length . "<br
/>";
            echo "Not NULL: " . $table_property->not_null . "<br />";
            echo "Primary Key: " . $table_property->primary_key .
"<br />";
            echo "Unique Key: " . $table_property->unique_key . "<br
/>";
            echo "Field Type: " . $table_property->type . "<br /><br
/>";
            }
      }
      mysql_close($connection);
?>
```

MYSQL_FETCH_OBJECT()

The mysql_fetch_object() function returns a row from a recordset as an object.This function gets a row from the mysql_query() function and returns an object on success, or FALSE on failure or when there are no more rows.

mysql_fetch_object(data)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
<?php
	mysql_connect("localhost", "root", "") or die ("Could not connect");
	mysql_select_db("mydb");
	$query = "SELECT * FROM student_info";
	$result = mysql_query($query);
	while ($row = mysql_fetch_object($result)) {
		echo $row->id;
		echo $row->name;
	}
	mysql_free_result($result);
?>
```

MYSQL_FETCH_ROW()

The mysql_fetch_row() function returns a row from a recordset as a numeric array.This function gets a row from the mysql_query() function and returns an array on success, or FALSE on failure or when there are no more rows.

mysql_fetch_row(data)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
<?php
	$connection = mysql_connect("localhost","root","");

	if (!$connection) {
	die('PHP Mysql database connection could not connect : ' . mysql_error());
	}
	else{
		$db_name = "mydb";

		mysql_select_db($db_name, $connection);
		$sql = "select * from student_info;";
		$records = mysql_query($sql, $connection);
			while($rows = mysql_fetch_row($records)){
				echo "Student Id : " . $rows[0] . "<br />";
				echo "Student Name : " . $rows[1] . "<br />";
				echo "Student Email : " . $rows[2] . "<br />";
				echo "Student Address : " . $rows[3] . "<hr />";
			}
	}
	mysql_close($connection);
?>
```

MYSQL_FIELD_TYPE()

The mysql_field_type() function returns the type of a field in a recordset.Returns the type of the specified field on success, or FALSE on failure.

mysql_field_type(data,field_offset)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

field_offset Required. Specifies which field to start on. 0 indicates the first field

```
<?php
     mysql_connect("localhost", "root", "");
     $result_handle = mysql_list_fields ("mydb","student_info");
     $number_fields = mysql_num_fields ($result_handle);
     echo "Table 'student_info' in database 'mydb' contains
$number_fields fields:<ol>";
     for ($index=0; $index < $number_fields; ++$index) {
        echo '<li>', mysql_field_type ($result_handle, $index),
'</li>';
     }
     echo '</ol>';
?>
```

MYSQL_INSERT_ID()

The mysql_insert_id() function returns the AUTO_INCREMENT ID generated from the previous INSERT operation.This function returns 0 if the previous operation does not generate an AUTO_INCREMENT ID, or FALSE on MySQL connection failure.Note: Be sure to call mysql_insert_id() immediately after a query to get the correct value.

mysql_insert_id(connection)

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
     mysql_connect("localhost", "root", "");
     mysql_select_db("mydb");
     $sql = "INSERT INTO student_info(name,email,address) VALUES
('abc','xyz@test.com','rajkot')";
     $result = mysql_query($sql);
     echo "ID of last inserted record is: " . mysql_insert_id();
     mysql_close();
?>
```

MYSQL_LIST_FIELDS

Deprecated. List MySQL table fields. It is preferable to use mysql_query() to issue an SQL SHOW COLUMNS FROM table [LIKE 'name'] statement instead.Retrieves information about the given table name.

```
<?php
// Included code that connects to a MySQL server and sets a default
database
// See the MySQL Functions chapter introduction for the source code
for the file
     mysql_connect("localhost", "root", "");
     $result_handle = mysql_list_fields ("mydb","student_info");
     $number_fields = mysql_num_fields ($result_handle);
     echo "Table 'student_info' in database 'mydb' contains
$number_fields fields:<ol>";
     for ($index=0; $index < $number_fields; ++$index) {
        echo '<li>', mysql_field_name ($result_handle, $index),
'</li>';
     }
     echo '</ol>';
?>
```

MYSQL_LIST_TABLES

Deprecated. List tables in a MySQL database. It is preferable to use mysql_query() to issue an SQL SHOW TABLES [FROM db_name] [LIKE 'pattern'] statement instead.

```
<?php
     mysql_connect("localhost", "root", "") or die ("Could not
connect");
     $result = mysql_list_tables("mysql");
     $num_rows = mysql_num_rows($result);
     for ($i = 0; $i < $num_rows; $i++)
          echo "Table: ", mysql_tablename($result, $i), "<br>";
     mysql_free_result($result);
?>
```

MYSQL_NUM_FIELDS()

The mysql_num_fields() function returns the number of fields in a recordset.This function returns FALSE on failure.

mysql_num_fields(data)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
<?php
     mysql_connect("localhost", "root", "");
     mysql_select_db("mydb");
     $result = mysql_query("select id,name,email from student_info
where id = 5");
     echo mysql_num_fields($result); // since three fields are
fetched, retutns 3
?>
```

MYSQL_NUM_ROWS()

The mysql_num_rows() function returns the number of rows in a recordset.This function returns FALSE on failure.

mysql_num_rows(data)

data Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

```
<?php
      $connection = mysql_connect("localhost", "root", "");
      if (!$connection) {
            die('PHP Mysql database connection could not connect : '
. mysql_error());
      }
      $db_selected = mysql_select_db("mydb", $connection);
      $sql_query = "SELECT * FROM student_info";
      $result = mysql_query($sql_query, $connection);
      echo "Number of rows : " . mysql_num_rows($result);
      mysql_close($connection);
?>
```

MYSQL_QUERY()

The mysql_query() function executes a query on a MySQL database.This function returns the query handle for SELECT queries, TRUE/FALSE for other queries, or FALSE on failure.

mysql_query(query,connection)

query Required. Specifies the SQL query to send (should not end with a semicolon)

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
      $connection = mysql_connect("localhost","root","");

      if (!$connection) {
      die('PHP Mysql database connection could not connect : ' .
mysql_error());
      }
      else{
            $db_name = "mydb";

            mysql_select_db($db_name, $connection);
            $sql = "select * from student_info;";
            $records = mysql_query($sql, $connection);
                  while($rows = mysql_fetch_array($records)){
                        echo "Student Id : " . $rows['id'] . "<br />";
                        echo "Student Name : " . $rows['name'] . "<br
/>";
```

```
                        echo "Student Email : " . $rows['email'] .
"<br />";
                        echo "Student Address : " . $rows['address'] .
"<hr />";
                  }
      }
      mysql_close($connection);
?>
```

MYSQL_RESULT()

The mysql_result() function returns the value of a field in a recordset.This function returns the field value on success, or FALSE on failure.This function is slower than mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() and mysql_fetch_object().This function should not be used together with mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() or mysql_fetch_object().

mysql_result(data,row,field)

data Required. Specifies which result handle to use. The data pointer is the return from the mysql_query() function

row Required. Specifies which row number to get. Row numbers start at 0

field Optional. Specifies which field to get. Can be field offset, field name or table.fieldname. If this parameter is not defined mysql_result() gets the first field from the specified row

```
<?php
      mysql_connect("localhost", "root", "");
      mysql_select_db("mydb");
      $result = mysql_query("select name from student_info");
      $no_result = mysql_num_rows($result);
      echo "<h2>Here is a list of the Name :</h2>";
      for ($i=0;$i<$no_result;$i++)
      {
            echo "<p>".mysql_result($result,$i,'name')."</p>";
      }
      mysql_close();
?>
```

MYSQL_SELECT_DB()

The mysql_select_db() function sets the active MySQL database.This function returns TRUE on success, or FALSE on failure.

mysql_select_db(database,connection)

database Required. Specifies the database to select.

connection Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

```
<?php
      $connection = mysql_connect("localhost","root","");

      if (!$connection) {
      die('PHP Mysql database connection could not connect : ' .
mysql_error());
      }
      else{
            $db_name = "mydb";

            mysql_select_db($db_name, $connection);
            $sql = "select * from student_info;";
            $records = mysql_query($sql, $connection);
                  while($rows = mysql_fetch_array($records)){
                        echo "Student Id : " . $rows['id'] . "<br />";
                        echo "Student Name : " . $rows['name'] . "<br
/>";
                        echo "Student Email : " . $rows['email'] .
"<br />";
                        echo "Student Address : " . $rows['address'] .
"<hr />";
                  }
      }
      mysql_close($connection);
?>
```

MYSQL_TABLENAME

Deprecated. Get table name of field, It is preferable to use mysql_query() to issue an SQL SHOW TABLES [FROM db_name] [LIKE 'pattern'] statement instead.

```
<?php
      mysql_connect("localhost", "root", "") or die ("Could not
connect");
      $result = mysql_list_tables("mysql");
      $num_rows = mysql_num_rows($result);
      for ($i = 0; $i < $num_rows; $i++) {
            echo "Table: ", mysql_tablename($result, $i), "<br>";
      }
      mysql_free_result($result);
?>
```

JQUERY

jQuery is not a language, but it is a well written JavaScript code. As quoted on official jQuery website, "it is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development".
In order to work with jQuery, you should be aware of the basics of JavaScript, HTML and CSS.It was released in January 2006 at BarCamp NYC by John Resig.

It is free, open source software Dual-licensed under the MIT License and the GNU General Public License. Microsoft has integrated jQuery officially into its IDE Visual Studio 2010 and jQuery intelli-sense is available in Visual Studio 2010 now.

jQuery is very compact and well written JavaScript code that increases the productivity of the developer by enabling them to achieve critical UI functionality by writing very small amount of code.

- It helps to improve the performance of the application
- It helps to develop most browser compatible web page
- It helps to implement UI related critical functionality without writing hundreds of lines of codes
- It is fast
- It is extensible – jQuery can be extended to implement customized behavior
- No need to learn fresh new syntaxes to use jQuery, knowing simple JavaScript syntax is enough

Simple and cleaner code, no need to write several lines of codes to achieve complex functionality

jQuery JavaScript file can be downloaded from jQuery Official website.jQuery usually comes as a single JavaScript file containing everything comes out of the box with jQuery. It can be included within a web page using the following mark-up:

```
<script type="text/javascript" src="jQuery-1.4.1-min.js"></script>
```

Ideally, this markup is kept in under <head></head> tag of your web page, however you are free to keep anywhere you want.
Do I need to refer jQuery file both in Master page/base page/template page and content page?
No, master page/base page/ template page basically helps to create consistent layout for the page in the application. In case you have referred the jQuery file in master page/base page/ template page that causes rendering the file in the browser, you do not need to refer jQuery file in the content page again.
In summary, there should not be more than one <script> tag with jQuery file reference in the source code of the rendered web page in the browser.

Difference between jQuery-x.x.x.js and jQuery.x.x.x min.js?

In terms of functionality, there is no difference between the jQuery-x.x.x.js and jQuery-x.x.x-min.js (also called minified version). However, this can play a vital role in the performance of the web page.jQuery-1.4.4.js file size is 178 KB as against its minified version jQuery-1.4.4-min.js that is only 76.7 KB in size. So when your page loads in the client's browser if you are not using minified version, it loads 178 KB file that takes more time to load than 76.7 KB.

jQUERY SYNTAX

The jQuery syntax is tailor made for selecting HTML elements and perform some action on the element(s).

```
$(selector).action()
```

- A dollar sign to define jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

```
$(this).hide() - hides current element.
$("p").hide() - hides all paragraphs.
$("p.test").hide() - hides all paragraphs with class="test".
$("#test").hide() - hides the element with id="test".
```

jQUERY SELECTORS

jQuery selectors allow you to select and manipulate HTML element(s).jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.All selectors in jQuery start with the dollar sign and parentheses: $().

THE ELEMENT SELECTOR

The jQuery element selector selects elements based on the element name.You can select all <p> elements on a page like this:

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

THE #ID SELECTOR

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$(document).ready(function(){
    $("button").click(function(){
        $("#test").hide();
    });
```

```
});
```

THE .CLASS SELECTOR

The jQuery class selector finds elements with a specific class.To find elements with a specific class, write a period character, followed by the name of the class:

```
$(document).ready(function(){
    $("button").click(function(){
        $(".test").hide();
    });
});
```

All type of selectors in jQuery, start with the dollar sign and parentheses: $().

$("")*	*selects all elements.*
$("p")	*selects all <p> elements.*
$("p.intro")	*selects all <p> elements with class="intro".*
$("p#intro")	*selects the first <p> elements with id="intro".*
$(":animated")	*selects all elements that are currently animated.*
$(":button")	*selects all <button> elements and <input> elements of type="button".*
$(":even")	*selects even elements.*
$(":odd")	*selects odd elements.*
$(this)	*Selects the current HTML element*
$("p#intro:first")	*Selects the first <p> element with id="intro"*
$(".intro")	*Selects all elements with class="intro"*
$("#intro")	*Selects the first element with id="intro"*
$("ul li:first")	*Selects the first element of the first *
$("ul li:first-child")	*Selects the first element of every *
$("[href]")	*Selects all elements with an href attribute*
$("[href$='.jpg']")	*Selects all elements with an href attribute that ends with ".jpg"*
$("[href='#']")	*Selects all elements with an href value equal to "#"*
$("[href!='#']")	*Selects all elements with an href value NOT equal to "#"*
$("div#intro .head")	*Selects all elements with class="head" inside a element with id="intro"*

JQUERY EVENT FUNCTIONS

The jQuery event handling methods are core functions in jQuery.Event handlers are methods that are called when "something happens" in HTML. The term "triggered (or "fired") by an event" is often used.

```
It is common to put jQuery code into event handler methods in the
<head> section:
<html>
      <head>
```

```
      <script type="text/javascript" src="jquery.js"></script>
            <script type="text/javascript">
                  $(document).ready(function(){
                    $("button").click(function(){
                        $("p").hide();
                    });
                  });
            </script>
      </head>
      <body>
            <h2>This is a heading</h2>
            <p>This is a paragraph.</p>
            <p>This is another paragraph.</p>
            <button>Click me</button>
      </body>
</html>
```

In the example above, a function is called when the click event for the button is triggered:

```
$("button").click(function() {..some code... } )
```

The method hides all <p> elements:

```
$("p").hide();
```

jQuery Events

Method / Property	Description
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
focus()	Attaches/Triggers the focus event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
load()	Removed in version 3.0. Attaches an event handler to the load event
ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event
unload()	Removed in version 3.0. Attaches an event handler to the unload event

jQUERY EFFECTS

```
<html>
      <head>
            <script type="text/javascript" src="jquery.js"></script>
            <script type="text/javascript">
```

```
                $(document).ready(function(){
                  $("#hide").click(function(){
                      $("p").hide();
                  });
                  $("#show").click(function(){
                      $("p").show();
                  });
                });
            </script>
      </head>
      <body>
            <p>If you click on the "Hide" button, I will
disappear.</p>
            <button id="hide">Hide</button>
            <button id="show">Show</button>
      </body>
</html>
```

```
<html>
      <head>
            <script type="text/javascript" src="jquery.js"></script>
            <script type="text/javascript">
                  $(document).ready(function(){
                    $("button").click(function(){
                        $("p").toggle();
                    });
                  });
            </script>
      </head>
      <body>
            <button>Toggle</button>
            <p>This is a paragraph with little content.</p>
            <p>This is another small paragraph.</p>
      </body>
</html>
```

```
<html>
      <head>
            <script type="text/javascript" src="jquery.js"></script>
                  <script type="text/javascript">
                  $(document).ready(function(){
                  $(".flip").click(function(){
                        $(".panel").slideToggle("slow");
                    });
                  });
            </script>
```

```
            <style type="text/css">
                  div.panel,p.flip{margin:0px;    padding:5px;    text-
align:center; background:#e5eecc; border:solid 1px #c3c3c3;}
                  div.panel{height:120px;     display:none;}
            </style>
      </head>
      <body>
            <div class="panel">
            <p>Because time is valuable, we deliver quick and easy
learning.</p>
            <p>At W3Schools, you can study everything you need to
learn, in an accessible and handy format.</p>
            </div>
            <p class="flip">Show/Hide Panel</p>
      </body>
</html>
```

AJAX AND JQUERY

jQuery provides a rich set of methods for AJAX web development.With jQuery AJAX, you can request TXT, HTML, XML or JSON data from a remote server using both HTTP Get and HTTP Post.And you can load remote data directly into selected HTML elements of your web page!

```
Calling Page using Post method with argument name and location
$.ajax({
      type: "POST",
      url: "some.php",
      data: { name: "mohammad", location: "Rajkot" }
      }).done(function( msg ) {
      alert( "Data Saved: " + msg );
});
```

```
Appending Html content using Get method
$.ajax({
      type: "GET",
      url: "test.html",
      }).done(function( html ) {
      $("#results").append(html);
});
```

CONTENT MANAGEMENT SYSTEM

A web site's content is a combination of different components like texts, graphics, images, and scripts, embedded files such as flash animations, audio/video streams or downloadable files. All of these may be components of one document (or HTML page in case of a web content management system).
Content management systems (CMS) are computer software systems for organizing, displaying and facilitating collaborative creation of this content.
One principle of many content management systems is, to separate the content from the layout, which makes it easier to preset the same content in different layouts for different media ("cross-media publishing") like web browser and printer. Separating content and layout also enables website designers to concentrate on the presentation, while others attend the content.
This can be achieved by storing the content and the layout in different resources and dynamically merge them together to the final document.

ADVANTAGES OF CONTENT MANAGEMENT SYSTEMS

- CMS facilitate the collaborative creation of websites. People can concentrate on the content while others care for the template to present the content. Also many CMS provide systems to enable users to add or modify content via their web browser
- CMS make it easier to display the same content in different ways, like a normal view for web browsers and a printer friendly view
- CMS make it easier to create new documents as one can concentrate on the content and do not have to care about the layout
- CMS make it easier to modify the layout of a website as one only has to modify the template at a single source instead of having to modify each single page to reflect the change
- CMS often can automatically create additional content like menus, sitemaps etc.
- CMS often provide methods to find content, for example by providing search functionality on the content

DIFFERENT TYPES OF CONTENT MANAGEMENT SYSTEMS

There are a lot of different Content Management Systems available, and beside the main common feature of offering the division between layout and content, the modes of operation can be very different.

FLATFILE VS. DATABASE-DRIVEN CMS

FLATFILE-CMS

On a flatfile-based CMS, the content is (usually) saved in clear text files. Depending on the concept, layout elements can be saved in separate text files as well, dividing layout and content, and having the CMS put those two together on accessing the page (or any other defined action).

DATABASE-MANAGED CMS

With a database-managed CMS, all data is saved in a database and will be requested from there. Using such a system therefore requires a database (including the rights to modify it).

From a technical point of view, using a database has the advantage of a better scaling system on really huge amounts of data - provided that the database is properly accessible. On the other hand, a flatfile-based CMS can be much faster with a moderate amount of data, just because the database communication is not needed. This can be especially important in a shared-host environment (one server, multiple domains hosted), as server load and reduction thereof directly influences the website speed.

WORDPRESS

WordPress is a free and open source blogging tool and a content management system (CMS) based on PHP and MySQL which runs on a web hosting service. Features include a plug-in architecture and a template system. WordPress is used by over 18.9% of the top 10 million websites as of August 2013. WordPress is currently the most popular blogging system being used on the Web, powering over 60 million websites worldwide.

It was first released on May 27, 2003, by founders Matt Mullenweg and Mike Little as a fork of b2/cafelog. As of September 2013, version 3.6 had been downloaded over 6 million times.

INSTALLATION

- Download and unzip the WordPress package if you haven't already.
- Create a database for WordPress on your web server, as well as a MySQL user who has all privileges for accessing and modifying it.
- Upload the WordPress files to the desired location on your web server:
- If you want to integrate WordPress into the root of your domain (e.g. http://example.com/), move or upload all contents of the unzipped WordPress directory (excluding the WordPress directory itself) into the root directory of your web server.
- If you want to have your WordPress installation in its own subdirectory on your web site (e.g. http://example.com/blog/), create the blog directory on your server and upload the contents of the unzipped WordPress package to the directory via FTP.
- Note: If your FTP client has an option to convert file names to lower case, make sure it's disabled.
- Run the WordPress installation script by accessing the URL in a web browser. This should be the URL where you uploaded the WordPress files.
- If you installed WordPress in the root directory, you should visit: http://example.com/
- If you installed WordPress in its own subdirectory called blog, for example, you should visit: http://example.com/blog/
- 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. Basically, you need to make a new MySQL database, username for it and grant that username the appropriate permissions for this database.
- Enter the details for your newly created MySQL database and press the Sumbit button.
- 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.

JOOMLA

Joomla is a free and open-source content management system (CMS) for publishing web content. It is built on a model–view–controller web application framework that can also be used independently.

Joomla is written in PHP, uses object-oriented programming (OOP) techniques (since version 1.5) and software design patterns, stores data in a MySQL, MS SQL (since version 2.5), or PostgreSQL (since version 3.0) database, and includes features such as page caching, RSS feeds, printable versions of pages, news flashes, blogs, polls, search, and support for language internationalization.

As of July 2013, Joomla has been downloaded over 35 million times. Over 6,000 free and commercial extensions are available from the official Joomla! Extension Directory, and more are available from other sources. It is estimated to be the second most used CMS on the Internet after WordPress.

INSTALLATION

- Before we start installing Joomla!, there are a couple prerequisites that need to be met to install successfully:
- Hosting - whether you have a dedicated server, or shared hosting plan, you'll need some sort of web hosting that meets the following requirements:
  - PHP v. 5.2+
  - MySQL 5.04+
  - Apache 1.3
- MySQL Database - you'll need access to a MySQL database, as well as the following credentials.
  - DB Name
  - Host Name
  - Username
  - Password
- FTP Client - for transferring files to your server. If you don't already have an FTP client that you use, we recommend Filezilla (ensure transfers are set to Binary)
- Download Latest Release of Joomla! 2.5.X
- Step 1 is to select your language in the screen above, then press the 'Next' button in the upper right corner.
- Step 2 runs a series of system and server checks to ensure that Joomla will be able to install, and function correctly. The top section of items should all be green and 'Yes'. If any are not (they'll be red and say "No") then you will need to take action to correct them. The bottom section of items are not required, but are recommended to ensure that Joomla can operate smoothly.
- Step 3 is to review the GNU license. This is the software license agreement for Joomla use, and simply requires you to press 'Next' in the top right corner after reviewing.

- Step 4 is configuring the database connection for Joomla. Here you will need to enter the Database type, Hostname, User name, Database name and the Database prefix for the MySQL database you have set up. If your settings are not correct, you might be unable to connect to the database. An additional option is to change the Table prefix. (Note: Joomla 2.5.x now offers a random default prefix) This may be helpful for security purposes.
- Step 5 is the FTP Configuration. This step is unnecessary for most sites and its need is usually identified if you find you cannot upload media or images or install Extensions. The details can be added at a later time directly within the Global Configuration in the Joomla Administration pages. This step is not required on servers installed on a Windows operating system.
- Step 6, and the last to installing Joomla, is the Site Configuration. Here you must add a Site Name, Admin User Name, Admin Password, and Admin e-mail address. If you are new to Joomla you should also CLICK Install the Sample Content, this will help with learning about Joomla.
- Step 7 Is the final step. Joomla is now installed! You must now remove the installation directory, this needs to be removed for security reasons to prevent anyone else from coming along and reinstalling Joomla over your existing site. Click the Remove Installation Directory button.