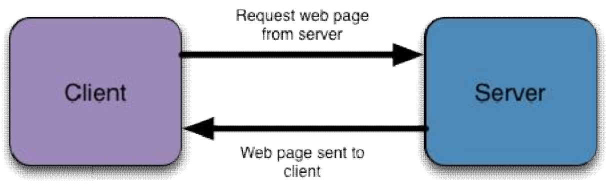
PHP

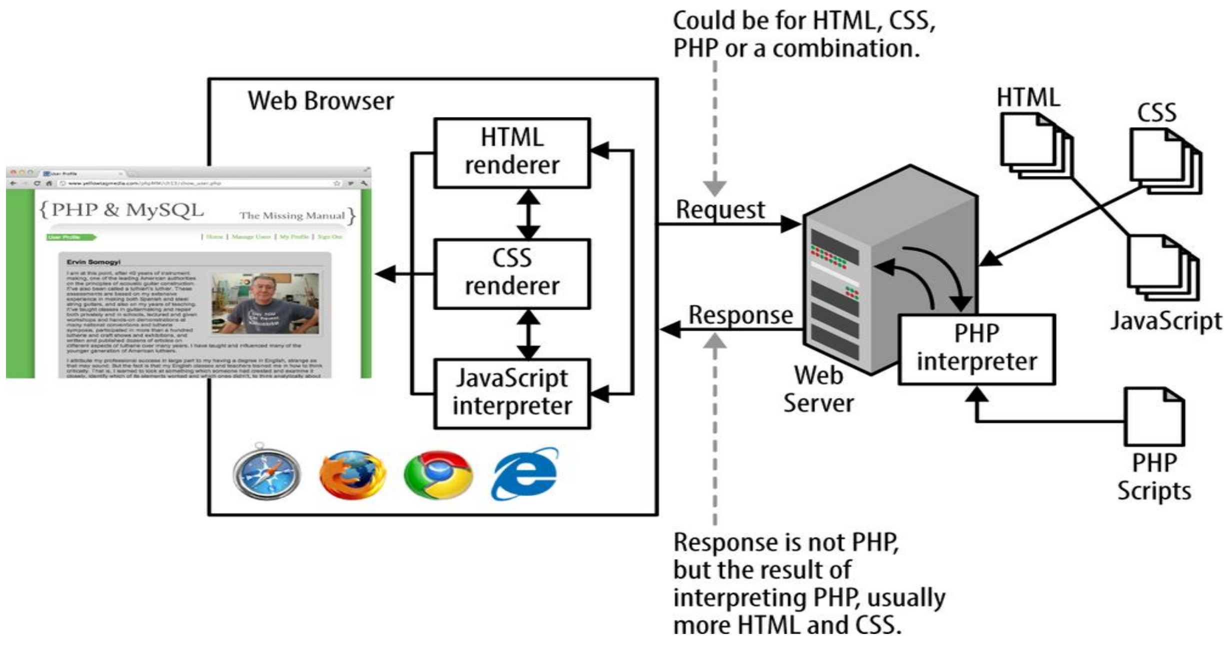
SERVER SIDE SCRIPTING

Most websites on the Internet have dynamic content. This means that the content displayed to the user has not been written directly into the HTML page but rather it has been generated by selecting information from a database that resides on the server. This is advantageous as it means that you can create a single template page and then populate it with different information from the database depending on what page is requested, eg. Mail account, facebook wall, shopping site.

The web is a *client-server system*. A web browser residing on your computer acts as a client which can request web pages from different web server. This can be shown as a diagram



When serving static web pages the server is normally asked for a file that has a *.htm* or *.html* extension e.g. http://mywebsite.com/index.html. If we wish to serve a dynamic page the extension would be different, for example *.php*. If a request comes in with one of these extensions the web server knows to pass the request to the PHP interpreter which then interprets it correctly and executes the script. Once the script has finished executing the Interpreter then passes the output back to the web server to be delivered as a response to the client request. The diagram demonstrates this:


Could be for HTML, CSS, PHP or a combination.
Web Browser
HTML renderer
CSS renderer
JavaScript interpreter
{PHP & MySQL The Missing Manual}
Request
Response
Web Server
PHP interpreter
HTML
CSS
JavaScript
PHP Scripts
Response is not PHP, but the result of interpreting PHP, usually more HTML and CSS.

Thus the server response can be HTML, CSS, JavaScript, depending on the request. These three make up the client side of the code that the web browser interprets, or runs. The server uses a server side scripting language to deal fundamentally with the database, and to response back to client the client code in HTML,CSS and JavaScript.

There are a number of server side scripting language each having pros and cons. Ex PHP, ASP, Perl, Python, JavaScript , Java.

PHP

PHP is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP is an extremely capable language, with a vast array of built-in functions to do everything from tracking user sessions to generating dynamic graphics and even PDF files on the fly! No modules to install, no commercial add-ins to buy… PHP handles everything itself! In fact, just about the only weakness of PHP is that it’s relatively difficult to expand the language to add non-standard functionality that is not handled by its built-in functions.

Installing PHP

PHP works on most of the operating systems like Windows, Linux and Mac OS. PHP can be installed on the development PC to safely create and test a web application without affecting the data or the security on a live server.

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- **Web Server** − PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- **Database** − PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

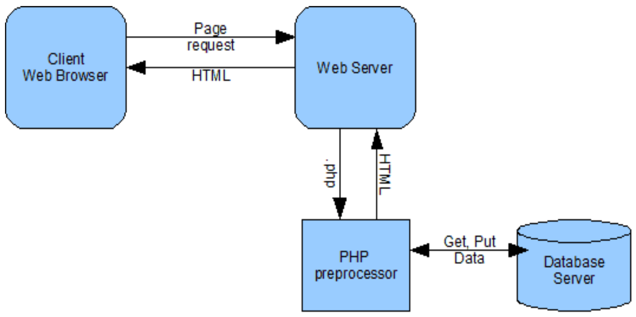
- **PHP Parser/Interpreter** – In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

For windows systems the process is fairly easy as PHP can be installed in two ways.

- **Manual Installation:** PHP interacts on the behalf of the web server. Thus for manually installing PHP we need a local installation of a web server Apache. If the Apache server is working on the PC then PHP installer can be downloaded from the official web site http://www.php.net/downloads.php. besides apache database server like MySQL also needs to be installed on the PC
- **All in One Package:** This is even simpler way. Packages like XAMPP, WAMP, contain Apache, PHP, MySQL and other application in a single installation file. These are fairly easy to configure.

PHP SYNTAX

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.



The default file extension for PHP files is ".php". A PHP script can be placed anywhere in the document. A PHP file normally contains HTML tags, and some PHP scripting code. PHP is case Insensitive language. However all variable names are case-sensitive. A statement in PHP is any expression that is followed by a semicolon (;). Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program.

A PHP script starts with **<?php** and ends with **?>**

```
<?php
// PHP code goes here
?>
```

Simple Program Hello World

```
<html>
<body>
<h1>My first PHP page</h1>
<?php
    echo "Hello World!";
```

```
?>
</body>
</html>
```

This programs simply returns the following code to the browser.

```
<html>
<body>
<h1>My first PHP page</h1>
    Hello World!
</body>
</html>
```

Comments

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code. PHP supports several ways of commenting

```
<?php
   // This is a single-line comment

   # This is also a single-line comment

   /*
     This is a multiple-lines comment block
     that spans over multiple
     lines
   */

?>
```

VARIABLE

The most important things to know about variables in PHP

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary so no declaration of type of variable.

PHP has a total of eight data types

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string operations.'
- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

```
<?php
   $x = 5;
   $y = 10;

   $y = $x + $y;

   echo $y; // outputs 15
?>
```

In PHP, variables can be declared anywhere in the script. The scope of a variable is the part of the script where the variable can be referenced/used.PHP has three different variable scopes:

- **Local :** A variable declared inside a function has local scope and can be accessed within that function only.
- **Global :**A variable declared outside a function has global scope and can be accessed outside a function
- **Static:** Used inside a function to make the value persistent between function calls

Example

```
<?php
   function myTest() {
       static $x = 0; // the value is assigned only first time
       echo $x;
       $x++;
   }
myTest();
myTest();
myTest();
?>
```

ECHO and PRINT Statement

In PHP there are two basic ways to get output: echo and print. echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

Example

```
<html>
<body>
<?php
   echo "<h2>PHP is Fun!</h2>";
   echo "Hello world!<br>";
   echo "I'm about to learn PHP!<br>";
   echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
</body>
</html>
```

Output to the Browser

```
<html>
<body>
<h2>PHP is Fun!</h2>
Hello world!<br>
I'm about to learn PHP!<br>
This string was made with multiple parameters.
?>
</body>
</html>
```

Browser Displays

PHP is Fun!
Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

OPERATORS

PHP language supports following type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Arithmetic Operator	**Comparison operator**	**Logical operator**	**Assignment operator**	**Conditional operator**
+ Add	== equal to === identical	&& AND	== assignment	? :
- Subtraction	!= not equal to !== not identical	|| OR	+= add assign	
* Multiply	< less than	! NOT	-+ sub assign	
/ Division	> Greater than		*= multiple assign	
% Modulus	< less/equal to		/= divide assign	
++ increment	>greater/equal			
-- decrement				

Example

```
<?php
  $x = 10;
  $y = 6;
  echo $x + $y;
?>
<?php
  $x = 100;
  $y = 50;
  if ($x == 100 && $y == 50) {
      echo "Hello world!";
  }
?>
<?php
  $x = 10;
  $y = 50;
  if ($x < $y) {
      echo "Less Than!";
  }
?>
```

DECISIONS

PHP supports following three decision making statements

- **if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** – is used if you 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

Example

It outputs “have a good day” if current time is less than 20 else “ have a good night”.

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

LOOPS

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** – loops through a block of code a specified number of times.
- **while** – loops through a block of code if and as long as a specified condition is true.
- **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** – loops through a block of code for each element in an array

Syntax

```
while (condition is true) {
    code to be executed;
}

do {
    code to be executed;
} while (condition is true);

for (init counter; test counter; increment counter) {
    code to be executed;
}

foreach ($array as $value) {
    code to be executed;
}
```

Example while loop

```
<html>
<body>
<?php
  $x = 1;

  while($x <= 5) {
     echo "The number is: $x <br>";
     $x++;
   }
?>
</body>
</html>
```

Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Example foreach loop

```
<?php
$colors = array("red", "green", "blue", "yellow");

  foreach ($colors as $value) {
      echo "$value <br>";
```

```
    }
?>
```

Output

```
red
green
blue
yellow
```

PHP String

A sequence of characters is called a string like “Welcome to PHP”. In PHP String can be written in two ways

- **Single Quoted Stings** as ‘Hello’ ‘this is single quoted string’. These are treated literally as they are
- **Double Quoted Strings** as “Hello”, “this is a double quoted string”. These replace the variables with their values and also interpret special character sequences.

Example

```
<?php
   $variable = "name";
   $literally = 'My $variable will not print!\\n';

   print($literally);
   print "<br />";

   $literally = "My $variable will print!\\n";

   print($literally);
?>
```

Output

```
My $variable will not print \\n
My name will print
```

The escape-sequence replacements are –

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

String concatenation operator

PHP has the operator dot (.) to concatenate strings together.

Example

```
<?php
   $string1="Hello World";
   $string2="1234";

   echo $string1 . " " . $string2;
?>
```

Output

```
Hello World 1234
```

String Functions

PHP has a number of inbuilt string functions to perform common tasks associated with the strings. Some of them are listed here.

Function	Description
strlen()	**Returns the length of the string**
strrev()	**Reverses the string**
strpos()	**Searches a specific text within the string**
str_replace()	**Replaces some characters with some other characters in the string**

Examples

```
<?php
  echo strlen("Hello world!");
  echo strrev("Hello world!");
  echo strpos("Hello world!", "world");
  echo str_replace("world", "universe", "Hello world!");

?>
```

Output

```
12
!dlrow olleH
7
Hello universe
```

Array

An array is a special variable, which can hold more than one value at a time. In PHP, the array() function is used to create an array. There are three types of arrays:

- **Indexed /Numeric array:** Arrays with a numeric index
- **Associative arrays :** Arrays with named keys
- **Multidimensional arrays:** Arrays containing one or more arrays

Indexed 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.

Example

```
<?php
    // First method to create array
    $numbers = array( 1, 2, 3, 4, 5);

    foreach( $numbers as $value ) {
       echo "Value is $value <br />";
    }

    // Second method to create array
    $numbers[0] = "one";
    $numbers[1] = "two";
    $numbers[2] = "three";
    $numbers[3] = "four";
    $numbers[4] = "five";

    foreach( $numbers as $value ) {
       echo "Value is $value <br />";
    }
?>
```

Output

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

Associative Array

Associative arrays are arrays that use named keys that you assign to them. 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.

Example

```php
<?php
     /* First method to associate create array. */
     $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

     echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
     echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
     echo "Salary of zara is ".  $salaries['zara']. "<br />";

     /* Second method to create array. */
     $salaries['mohammad'] = "high";
     $salaries['qadir'] = "medium";
     $salaries['zara'] = "low";

     echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
     echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
     echo "Salary of zara is ".  $salaries['zara']. "<br />";
  ?>
```

output

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

This array is used to associate key value pairs like employee with salary, mobile number with a person name etc.

Multi-Dimensional Array

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 index. PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. The dimension of an array indicates the number of indices you need to select an element.

GET and POST

There are two ways the browser client can send information to the web server. Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

```
name1=value1&name2=value2&name3=value3
```

- **The GET Method**
  - The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?**character
  - http://www.test.com/index.htm?name1=value1&name2=value2
  - The GET method produces a long string that appears in your server logs
  - Never use GET method if you have password or other sensitive information to be sent to the server.
  - GET can't be used to send binary data, like images or word documents, to the server.
  - The PHP provides **$_GET** associative array to access all the sent information using GET method.

- **The POST Method**
  - The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
  - The POST method does not have any restriction on data size to be sent.
  - The POST method can be used to send ASCII as well as binary data
  - The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
  - The PHP provides **$_POST** associative array to access all the sent information using POST method.

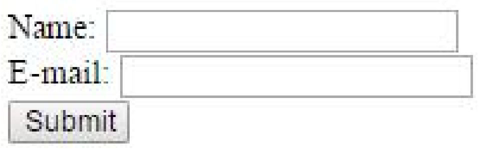
GET Method

Example

```
<html>
<body>
<form action="welcome.php" method="get">
    Name: <input type="text" name="name"><br>
    E-mail: <input type="text" name="email"><br>
    <input type="submit">
```

```
</form>
</body>
</html>
```

Output



When the 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 method used to get the data with the form field is get.

In the welcome.php file we use the $_GET superglobal of the PHP to fetch the data sent by the browser.

Example

```
<html>
<body>

   Welcome <?php echo $_GET["name"]; ?><br>
   Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Output

Welcome <name entered by user>

Your email address is <email entered by user

The same result can be achieved using the post method simply specifying method = post in the form tag. The values can be accessed in the PHP code using $_POST associative array.

Cookie

A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.There are three steps involved in identifying returning users

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user

A cookie is created with the setcookie() function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

- **Name** − This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** − This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** − This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** − This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** − This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** − This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Example setting and retrieve cookie

```
<?php
    $cookie_name = "user";
    $cookie_value = "John Doe";
    setcookie($cookie_name, $cookie_value, time()+(86400 * 30), "/");
// 86400 = 1 day
?>
<?php
   if(!isset($_COOKIE[$cookie_name])) {
      echo "Cookie named '" . $cookie_name . "' is not set!";
   } else {
      echo "Cookie '" . $cookie_name . "' is set!<br>";
      echo "Value is: " . $_COOKIE[$cookie_name];
   }
?>
```

Example deleting cookie

```
<?php
// set the expiration date to one hour ago
    setcookie("user", "", time() - 3600);
?>
```

Session

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer. 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.

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.

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION. Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

```
<?php
   // Start the session
      session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
   // Set session variables
     $_SESSION["favcolor"] = "green";
     $_SESSION["favanimal"] = "cat";
     echo "Session variables are set.";
?>

</body>
</html>
```

Get PHP Session Variable Values

we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

```
<?php
      session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
   // Echo session variables that were set on previous page
      echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
      echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Output

```
Favorite color is green.
Favorite animal is cat.
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

```
<?php
     session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
   // remove all session variables
      session_unset();

   // destroy the session
      session_destroy();
?>

</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

```
<?php
  // to change a session variable, just overwrite it
    $_SESSION["favcolor"] = "yellow";

?>
```

EMAIL

PHP must be configured correctly in the php.ini file with the details of how your system sends email. Windows users should ensure that two directives are supplied. The first is called SMTP that defines your email server address. The second is called sendmail_from which defines your own email address.

PHP makes use of mail() function to send an email. This function requires three mandatory arguments that specify the recipient's email address, the subject of the message and the actual message additionally there are other two optional parameters.

Syntax

```
mail(to, subject, message, headers, parameters);
```

Argument	**Description**
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF
parameters	Optional. Specifies an additional parameter to the send mail program

Send a simple email:

```
<?php
  // the message
    $msg = "First line of text\nSecond line of text";
```

```
        // send email
           mail("someone@example.com","My subject",$msg);
      ?>
```

Sending HTML email:

```
<html>

   <head>
      <title>Sending HTML email using PHP</title>
   </head>

   <body>

      <?php
         $to = "xyz@somedomain.com";
         $subject = "This is subject";

         $message = "<b>This is HTML message.</b>";
         $message .= "<h1>This is headline.</h1>";

         $header = "From:abc@somedomain.com \r\n";
         $header .= "Cc:afgh@somedomain.com \r\n";
         $header .= "MIME-Version: 1.0\r\n";
         $header .= "Content-type: text/html\r\n";

         $retval = mail ($to,$subject,$message,$header);

         if( $retval == true ) {
            echo "Message sent successfully...";
         }else {
            echo "Message could not be sent...";
         }
      ?>

   </body>
</html>
```

MySQL

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

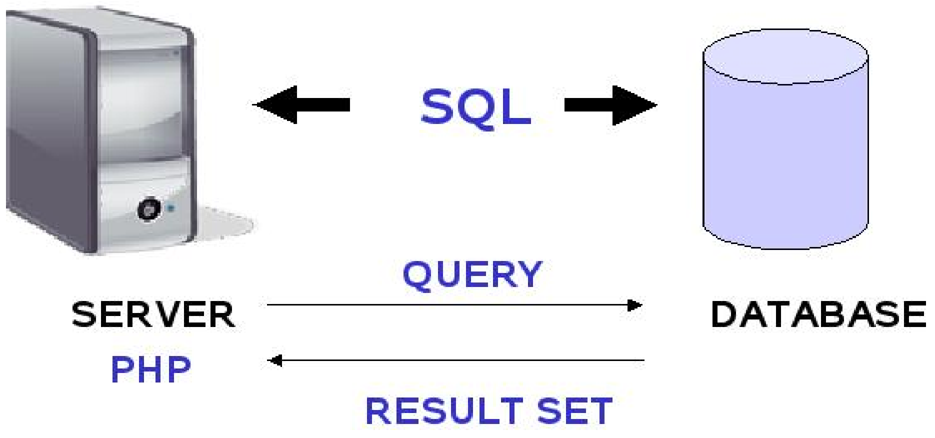
- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

Pre requisite for working with MySQL database

- MySQL must be installed and running.
- Creating a database user with a password

SQL

SQL is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). The database server can be running on the same machine or on other machine. The PHP interacts with the database in the SQL language.



NOTE: SQL is language, MySQL is Database Management Software (DBMS)

PHP can work with a MySQL database using:

- **MySQL extension**
- **MySQLi extension (the "i" stands for improved)**
- **PDO (PHP Data Objects)**

Here MySQL extension are described

PHP connection to MySQL

PHP provides **mysql_connect** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or FALSE on failure.

connection mysql_connect(server,user,passwd,,new_link,client_flag);

```
<?php

   $servername = "localhost";
   $username = "username";
   $password = "password";
// trying to connect
   $conn = mysql_connect($servername,$username, $password);
// failed connection
   if(! $conn ) {
      die('Could not connect: ' . mysql_error());
   }

   echo 'Connected successfully';
   mysql_close($conn);
?>
```

PHP closing connection to MySQL

bool mysql_close(connection);

Selecting a Database

Once a connection has been established with a database server then it is required to select a particular database. This is required because there may be multiple databases residing on a single server and you can do work with a single database at a time.PHP provides function **mysql_select_db** to select a database.It returns TRUE on success or FALSE on failure.

Syntax

bool mysql_select(database_name, connection);

Database_name	**:database to work with**
Connection	**:connection to the database server**

Query the database

PHP uses **mysql_**query function to query the database. When happen with the data in database depends on the query and the privileges of the user connected to the database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

bool mysql_query(sql , connection) ;

sql	**: query string**
connection	**: connection to the database server**

Creating a Database

To create a database follow the simple steps

- Connect to the database server with user having admin rights using **mysql_connect**
- Query the database with **mysql_query** with query string to create the database.

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysql_connect($servername, $username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysql_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysql_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysql_error($conn);
}

mysql_close($conn);
?>
```

Creating a Table

The CREATE TABLE statement is used to create a table in MySQL. This SQL syntax nothing to do with PHP.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysql_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysql_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP
  )";

if (mysql_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysql_error($conn);
}

mysql_close($conn);
?>
```

Insert Data into MySQL Database

Data can be entered into MySQL tables by executing SQL **INSERT** statement through PHP function **mysql_query**. Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

syntax

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Example

```
<?php
  $servername = "localhost";
  $username = "username";
  $password = "password";
  $dbname = "myDB";
// Create connection
  $conn = mysql_connect($servername, $username, $password, $dbname);
// Check connection
  if (!$conn) {
     die("Connection failed: " . mysql_connect_error());
  }

  $sql = "INSERT INTO MyGuests (firstname, lastname, email)
          VALUES ('John', 'Doe', 'john@example.com')";

  if (mysql_query($conn, $sql)) {
     echo "New record created successfully";
  } else {
     echo "Error: " . $sql . "<br>" . mysql_error($conn);
  }

mysql_close($conn);
?>
```

Getting Data From MySQL Database

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function mysql_query. There are several options to fetch data from MySQL.

mysql_fetch_assoc()

The most frequently used option is to use function **mysql_fetch_assoc()**. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysql_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysql_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row as a row in table
<table>
    echo "<tr> <td>ID</td> <td>NAME</td>  </tr>";
    while($row = mysqli_fetch_assoc($result))
       echo "<tr><td>" . $row["id"]. "</td><td>".
        $row["firstname"]." " . $row["lastname"]. "</td> </tr>" ;
    }

</table>
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

This example displays that data from the database in a table.

PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks. When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks. When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.

die() function

This function is used to deliver messages to the client browser in case of error. While writing your PHP program you should check all possible error condition before going ahead and take appropriate action when required.

Example

```
<?php
    $file=fopen("welcome.txt","r");
?>
```

If the file does not exist then error is generated and sent to the client browser. It is security risk to expose the internal file structure to the client as well as unprofessional

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

Now using die function

```
<?php
  if(!file_exists("welcome.txt")) {
      die("File not found");
  } else {
      $file=fopen("welcome.txt","r");
  }
?>
```

Now if the file does not exist the client get an error like this:

```
File not found
```

Creating custom error handler

In PHP creating a custom error handler is easy. We have to define a function that takes al least 2 parameters (error level and error message) but can accept up to 5 parameters (file , line number, error_context);

Syntax

error_function(error_level,error_message,error_file,error_line,error_context)

Parameter	Description
error_level	**Required: specify error level for the user defined error. Like 2 warning, 8 notice, 256 user error etc**
error_message	**Required: message to be displayed**
error_file	**Optional : filename in which error occurred**
error_line	**Optional : line number where error occurred**
error_context	**Optional : array describing the variables when error occured**

The default error handler for PHP is the built in error handler. So when we define our custom handler we have to set that function to be the error handler using this function

```
set_error_handler("customError");
```

Example

```
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error undefined variable
echo($test);
?>
```

Output

```
Error: [8] Undefined variable: test
```