Overview of PHP

by

Shashank Shetty
Assistant Professor
Dept of CSE
NMAMIT, Nitte

What is PHP?

- PHP stands for **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Oracle, Sybase, MS SQL, Generic ODBC, etc.)
- PHP is an open source software, is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Basic PHP Syntax

- A PHP scripting block always starts with **<?php** and ends with **?>**
- A PHP scripting block can be placed anywhere in the document

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Refer /srinivas/p1.php

- There are two basic statements to output text with PHP: ***echo*** and ***print***

Comments in PHP…

- In PHP, we use // to make a single-line comment or /* and */ to make a large comment block

Example:

```
<html>
<body>
<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

Variables in PHP…

- All variables in PHP start with a $ sign symbol

Format:

```
$var_name = value;
```

Example:

```
<?php
    $txt="Hello World!";
    $x=16;
?>
```

PHP is a Loosely Typed Language…

- In PHP, a variable does not need to be declared before adding a value to it
- PHP automatically converts the variable to the correct data type, depending on its value
- In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it
- In PHP, the variable is declared automatically when you use it

String Variables in PHP…

- String variables are used for values that contain characters

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

The Concatenation Operator…

- The concatenation operator (.) is used to put two string values together

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

Output: Hello World! What a nice day!

The strlen() function…

- The ***strlen()*** function is used to return the length of a string

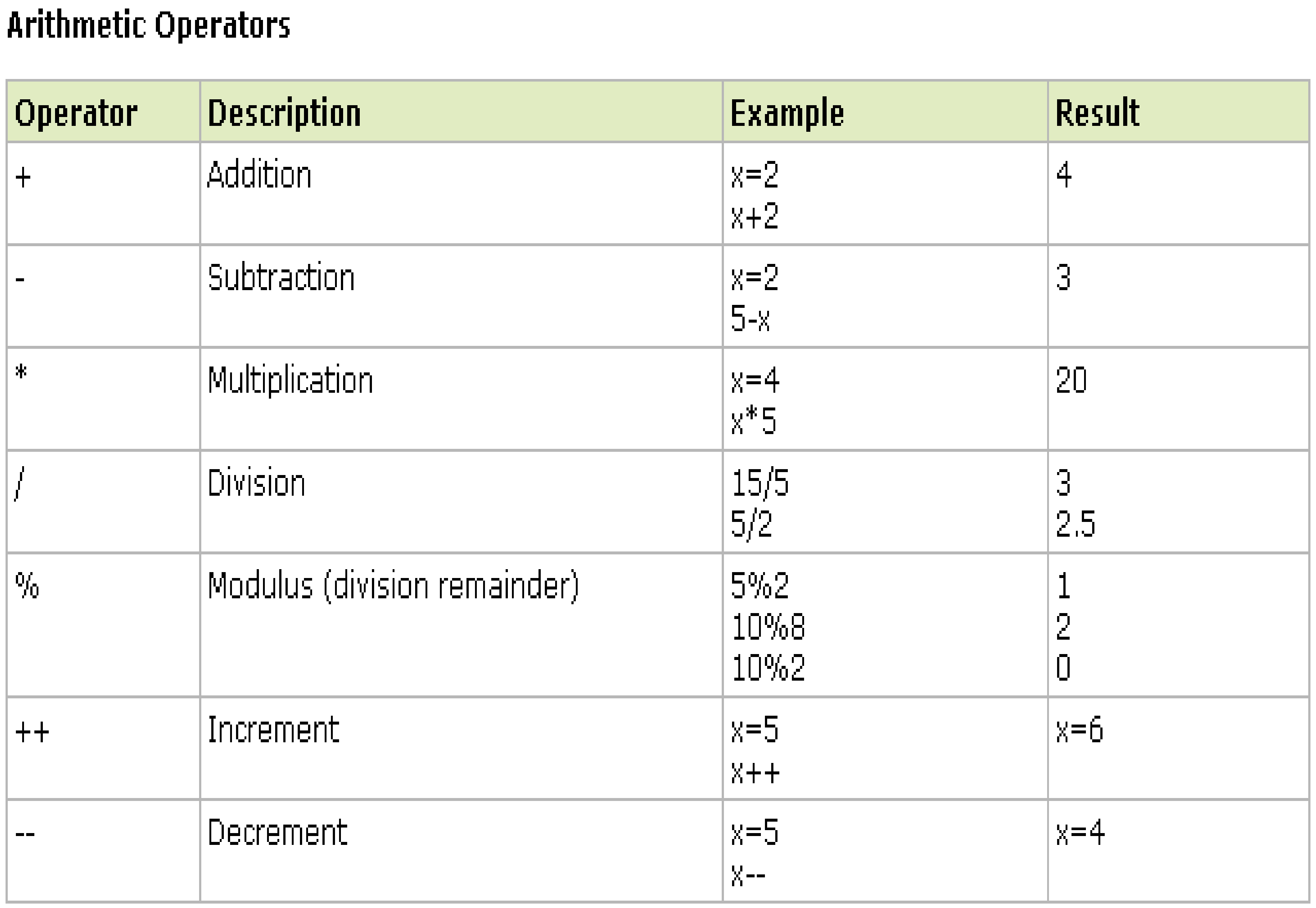
```
<?php
echo strlen("Hello world!");
?>
```

The strpos() function…

- The strpos() function is used to search for character within a string

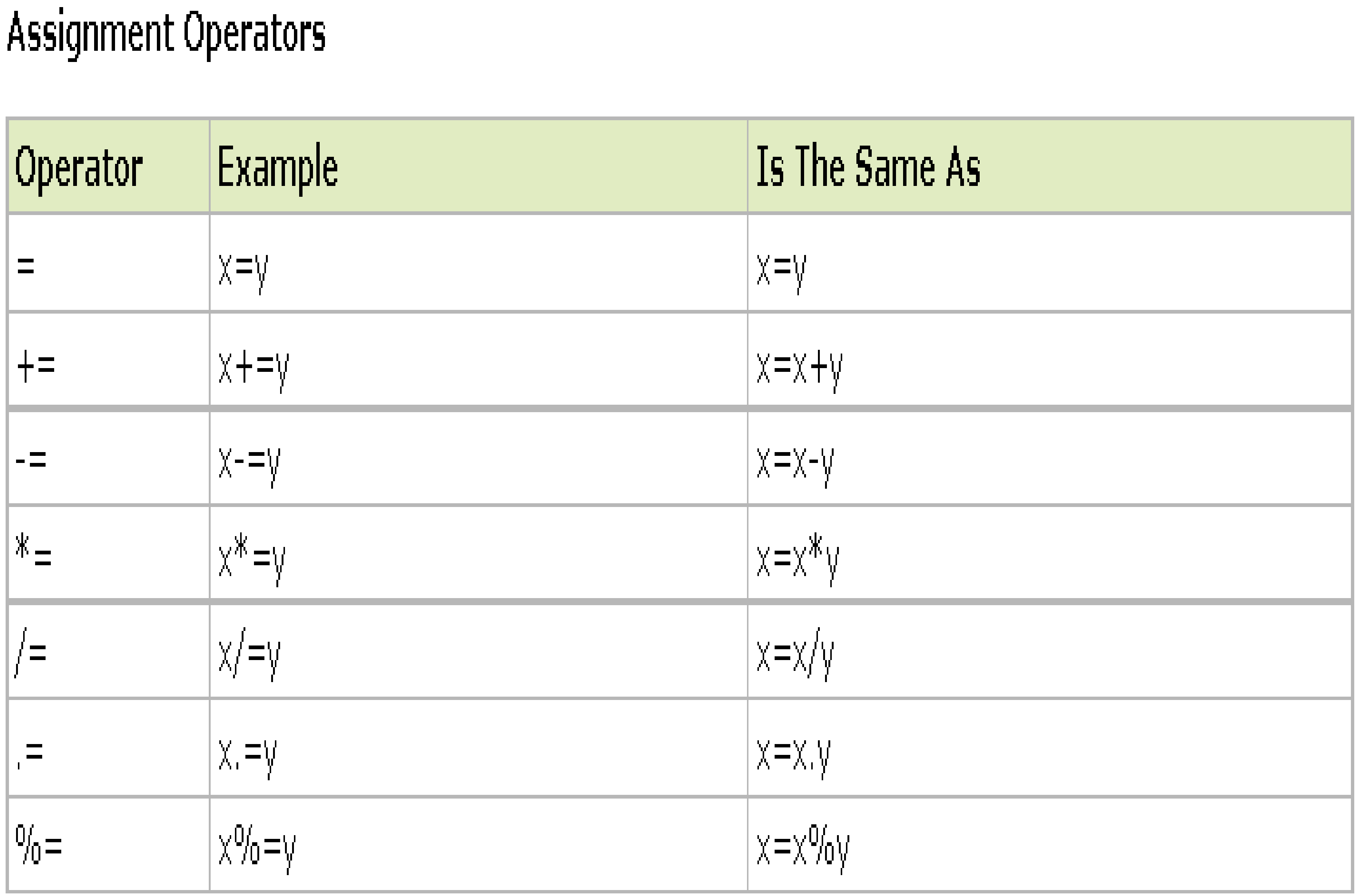
```
<?php
echo strpos("Hello world!","world");
?>
```

Output: 6



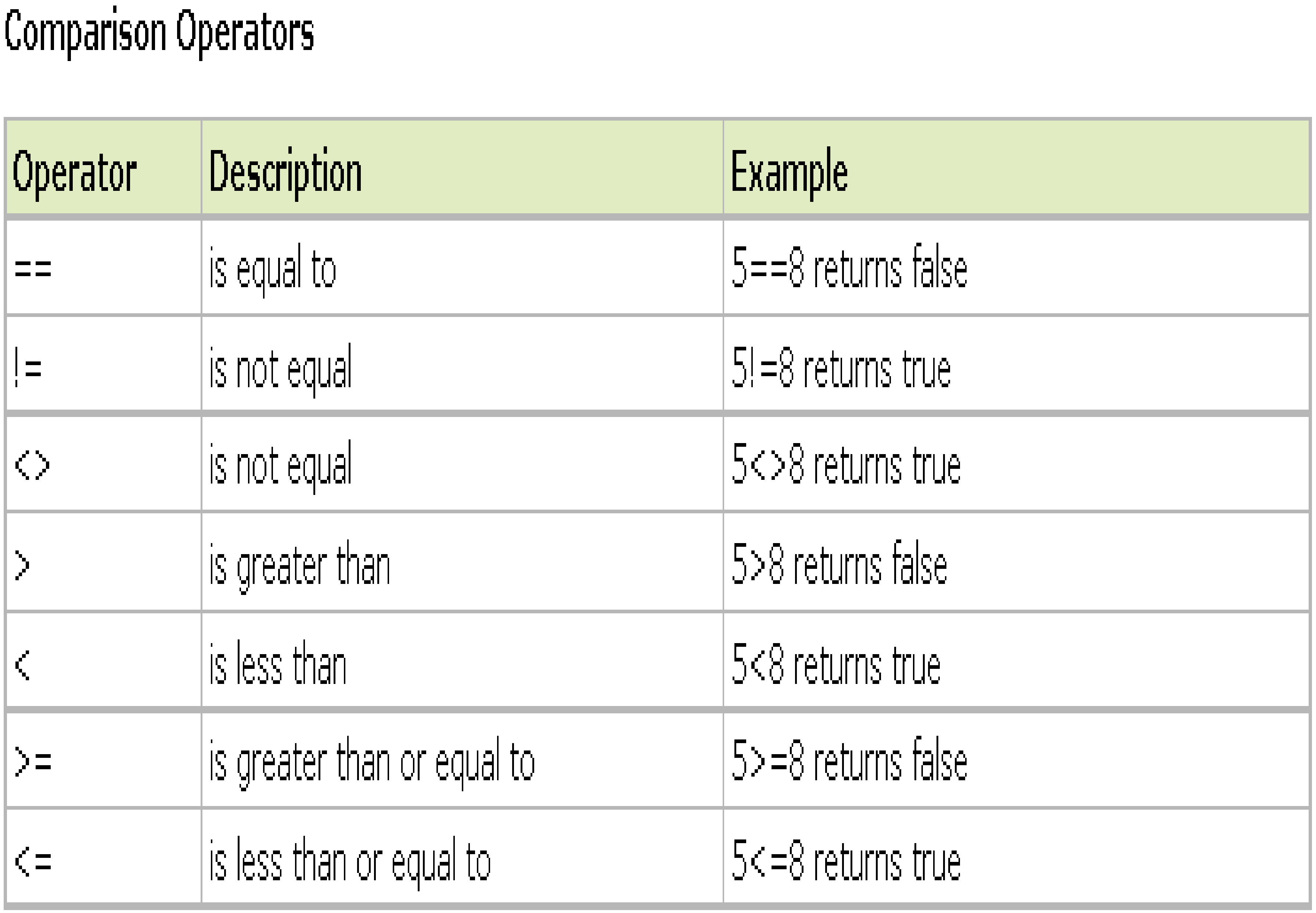
Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4



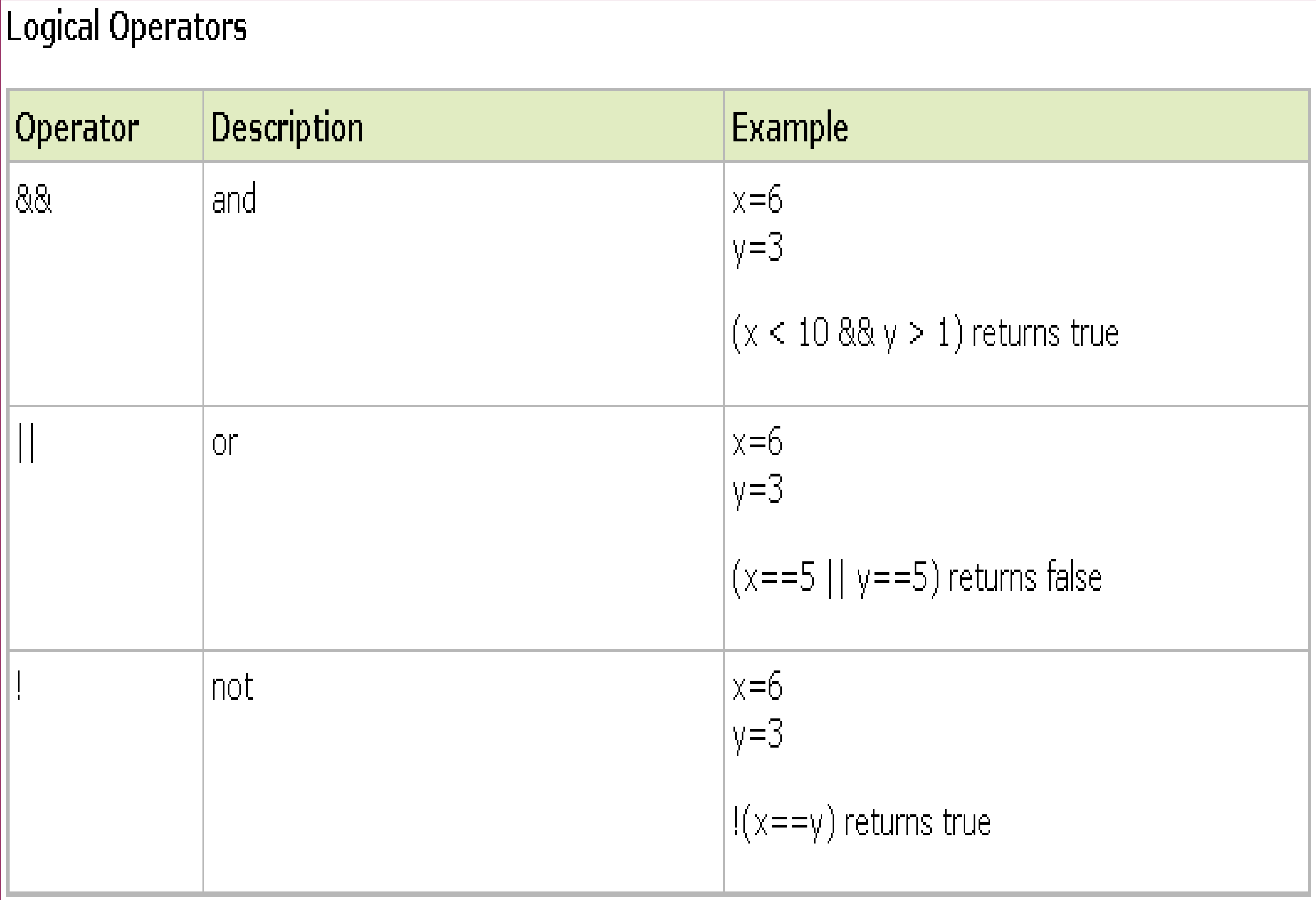
Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y



Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true



Logical Operators

Operator	Description	Example

&&	and	x=6
		y=3
		
		(x < 10 && y > 1) returns true

||	or	x=6
		y=3
		
		(x==5 || y==5) returns false

!	not	x=6
		y=3
		
		!(x==y) returns true

The if Statement…

if (*condition*)

    code to be executed if condition is true;

```
<html>
<body>
<?php
      $d=date("D");
      if ($d=="Fri") echo "Have a nice weekend!";
?>
</body>
</html>
```

The if...else Statement

```
if (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>
</body>
</html>
```

Refer p1a.php

- if (*condition*)
  code to be executed if condition is true;
  elseif (*condition*)
  code to be executed if condition is true;
  else
  code to be executed if condition is false;

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

Refer p1b.php

PHP Switch Statement…

```
switch (n)
{
case label1:
      code to be executed if n=label1;
  break;
case label2:
      code to be executed if n=label2;
  break;
default:
      code to be executed if n is different from both
      label1 and label2;
}
```

```
<html>
<body>

<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

PHP Arrays

- A numeric array stores each array element with a numeric index

```
$cars=array("Saab","Volvo","BMW","Toyota");
                        or
  $cars[0]="Saab";
  $cars[1]="Volvo";
  $cars[2]="BMW";
  $cars[3]="Toyota";
```

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish
cars.";
?>
```

Associative Arrays…

- With associative arrays we can use the values as keys and assign values to them

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

PHP Looping - While Loops…

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

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

refer p1c.php

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



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

```
for (init; condition; increment)
      {
      code to be executed;
      }


    <html>
    <body>
    <?php
          for ($i=1; $i<=5; $i++)
          {
                    echo "The number is " . $i . "<br />";
          }
    ?>

    </body>
    </html>
```

The foreach Loop…

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

```
<html>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
        echo $value . "<br />";
  }
?>
</body>
</html>
```

Refer p1d.php

PHP Functions

syntax:

```
        function functionName()
        {
                code to be executed;
        }
```

```
<html>
<body>
<?php
function writeName()
{
        echo “Raj";
}
echo "My name is ";
writeName();
?>
</body>
</html>
```

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Kumar..<br />";
}

echo "My name is ";
writeName(“Raj");
echo "My sister's name is ";
writeName(“Ajay");
echo "My brother's name is ";
writeName(“Amar");
?>

</body>
</html>
```

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

PHP Functions - Return values…

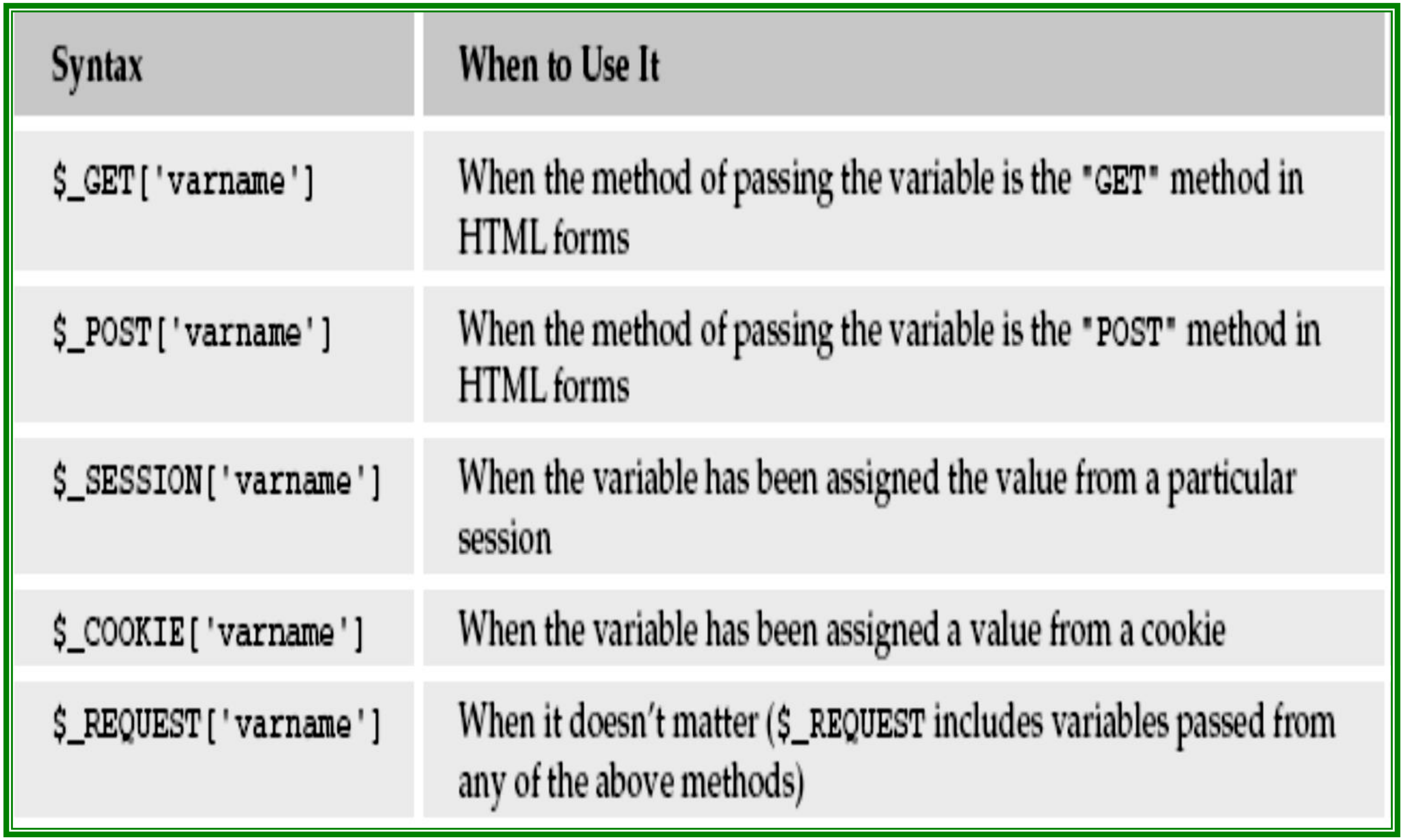
```
<html>
<body>

<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}

echo "1 + 16 = " . add(1,16);
?>

</body>
</html>
```

Passing Variables between Pages…



Syntax	When to Use It
$_GET['varname']	When the method of passing the variable is the "GET" method in HTML forms
$_POST['varname']	When the method of passing the variable is the "POST" method in HTML forms
$_SESSION['varname']	When the variable has been assigned the value from a particular session
$_COOKIE['varname']	When the variable has been assigned a value from a cookie
$_REQUEST['varname']	When it doesn't matter ($_REQUEST includes variables passed from any of the above methods)

Passing Variables through a URL

Example: ***(Query String)***

http://www.mydomain.com/news/articles/showart.php?id=12345

- It requests that the article with the ID number of "12345" be chosen for the showart.php program

- We can also combine variables in a URL by using an ampersand (&)

Example:

http://www.mydomain.com/news/articles/showart.php?id=12345&lang=en

Refer Example ***2_URL1.php and 2_URL2.php***

Passing Variables with Sessions…

- A *session* is basically a temporary set of variables that exists only until the browser has shut down
- Examples of session information include a session ID and whether or not an authorized person has “logged in” to the site
- This information is stored temporarily for your PHP programs to refer back to whenever needed

- Every session is assigned a unique session ID, which keeps all the current information together
- Session ID can either be passed through the URL or through the use of cookies
- To begin a session, use the function ***session_start()***

- First, we need to decide what information will be stored in our session
- Usually, it is information such as username and login information, but it can also be preferences that have been set at some point by the user
- An SID (session ID) will also be stored in the session array of variables

- ***Refer Example***

  3_SID_1.php and ***3_SID_2.php***

Passing Variables with Cookies…

- Cookies are tiny bits of information stored on our Web site visitor’s computer
- The advantage to storing information in a cookie versus a session is longevity
- Sessions alone can’t store information for more than the length of time the browser window is open
- Cookies, on the other hand, can live on a person’s computer until the developer has decided it’s been long enough and they automatically “die.”

- To set a cookie, you use the appropriately named ***setcookie()*** function

setcookie(‘*cookiename*’, ‘*value*’, ‘*expiration time*’, ‘*path*’, ‘*domain*’, ‘*secure connection*’);

- Cookie name (this is mandatory)

- Value of the cookie (such as the person’s username)

- Time in seconds when the cookie will expire

- Path (the directory where the cookie will be saved—the default is usually sufficient; this is optional)
- Domain (domains that may access this cookie—this is optional)
- Whether a cookie must have a secure connection to be set (defaults to 0; to enable this feature set this to 1)

- Refer Example ***4_Cookie_1.php*** and ***4_Cookie_2.php***
- ***Go to Tools -> Page Info -> Security ->View Cookies***

PHP Form Handling…

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

```
<html>
<body>

<form action="greet1.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

“greet1.php" looks like this:

```
<html>
<body>

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

</body>
</html>
```

Refer p2.html and greet1.php

The $_POST variable…

- The built-in $_POST variable is used to collect values from a form sent with method="post“
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send
- However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file)

- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send

The $_GET variable…

- The built-in $_GET variable is used to collect values from a form sent with method="get“
- Information sent from a form with the 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

```
<html>
   <body>

   <form action=“greet2.php" method=“get">
   Name: <input type="text" name="fname" />
   Age: <input type="text" name="age" />
   <input type="submit" />
   </form>

   </body>
   </html>
```

“greet2.php" looks like this:

```
<html>
<body>

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

</body>
</html>
```

Refer p3.html and greet2.php

- When the user clicks the "Submit" button, the URL sent to the server could look something like this:

  http://www.w3schools.com/welcome.php?fname=Peter&age=37

When to use method="get"?

- When using method="get" in HTML forms, all variable names and values are displayed in the URL
- This method should not be used when sending passwords or other sensitive information!
- However, because the variables are displayed in the URL, it is possible to bookmark the page
- This can be useful in some cases.
- The get method is not suitable for very large variable values. It should not be used with values exceeding 2000 characters

The PHP $_REQUEST variable…

- The PHP built-in $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE
- The $_REQUEST function can be used to collect form data sent with both the GET and POST methods

PHP Date() Function…

Syntax:

date(*format*, *timestamp*)

Format – Required; Specifies the format of the timestamp

Timestamp – Optional; Specifies a timestamp Default is the current date and time

- The required *format* parameter in the date() function specifies how to format the date/time
- Here are some characters that can be used:
- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- D – day of a week (Mon, Tue etc)

- Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:

```
<?php
echo date("Y/m/d") . "<br />";
echo date("Y.m.d") . "<br />";
echo date("Y-m-d")
?>
```

Output:

```
2009/05/11
2009.05.11
2009-05-11
```

Refer p4.php

System time…

- h – hours
- i – minutes
- s - seconds

$time_now=mktime(date('h')**+5**,date('i')**+30**,date('s'));

Refer p5.php

Server Side Includes (SSI)…

- We can insert the content of one PHP file into another PHP file before the server executes it, with the ***include()*** or ***require()*** function
- The two functions are identical in every way, except how they handle errors:
- ***include()*** generates a warning, but the script will continue execution
- ***require()*** generates a fatal error, and the script will stop

- Server side includes saves a lot of work
- This means that you can create a standard header, footer, or menu file for all your web pages
- When the header needs to be updated, you can only update the include file, or when you add a new page to your site, you can simply change the menu file (instead of updating the links on all your web pages)

PHP include() Function…

- The include() function takes all the content in a specified file and includes it in the current file

- If an error occurs, the include() function generates a warning, but the script will continue execution

Example

```
<html>
<body>

<?php include("header.php"); ?>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>

</body>
</html>
```

- Assume we have a standard menu file, called "menu.php", that should be used on all pages:

```
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
```

```
<html>
<body>

<?php include("menu.php"); ?>

<h1>Welcome to my home page.</h1>
<p>Some text.</p>

</body>
</html>
```

- If you look at the source code of the page above (in a browser), it will look like this:

```
<html>
<body>
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
</body>
</html>
```

Refer menu.php and p6.php

PHP require() Function…

- The require() function is identical to include(), except that it handles errors differently
- If an error occurs, the include() function generates a warning, but the script will continue execution
- The require() generates a fatal error, and the script will stop

Example…

```
<html>
<body>

<?php
include("wrongFile.php");
echo "Hello World!";
?>

</body>
</html>
```

Error message:

- **Warning:** include(wrongFile.php) [function.include]: failed to open stream:
  No such file or directory in C:\home\website\test.php on line 5

  Warning: include() [function.include]:
  Failed opening 'wrongFile.php' for inclusion (include_path='.;C:\php5\pear') in C:\home\website\test.php on line 5

  Hello World!

Notice that the echo statement is executed! This is because a Warning does not stop the script execution

Refer p8.php

Now, let's run the same example with the require() function…

```
<html>
<body>

<?php
require("wrongFile.php");
echo "Hello World!";
?>

</body>
</html>
```

Error message:

```
Warning: require(wrongFile.php)
[function.require]:
failed to open stream:
No such file or directory in
C:\home\website\test.php on line 5

Fatal error: require() [function.require]:
Failed opening required 'wrongFile.php'
(include_path='.;C:\php5\pear')
in C:\home\website\test.php on line 5
```

- The echo statement is not executed, because the script execution stopped after the fatal error

- It is recommended to use the require() function instead of include(), because scripts should not execute after error

Refer p9.php

PHP File Handling…

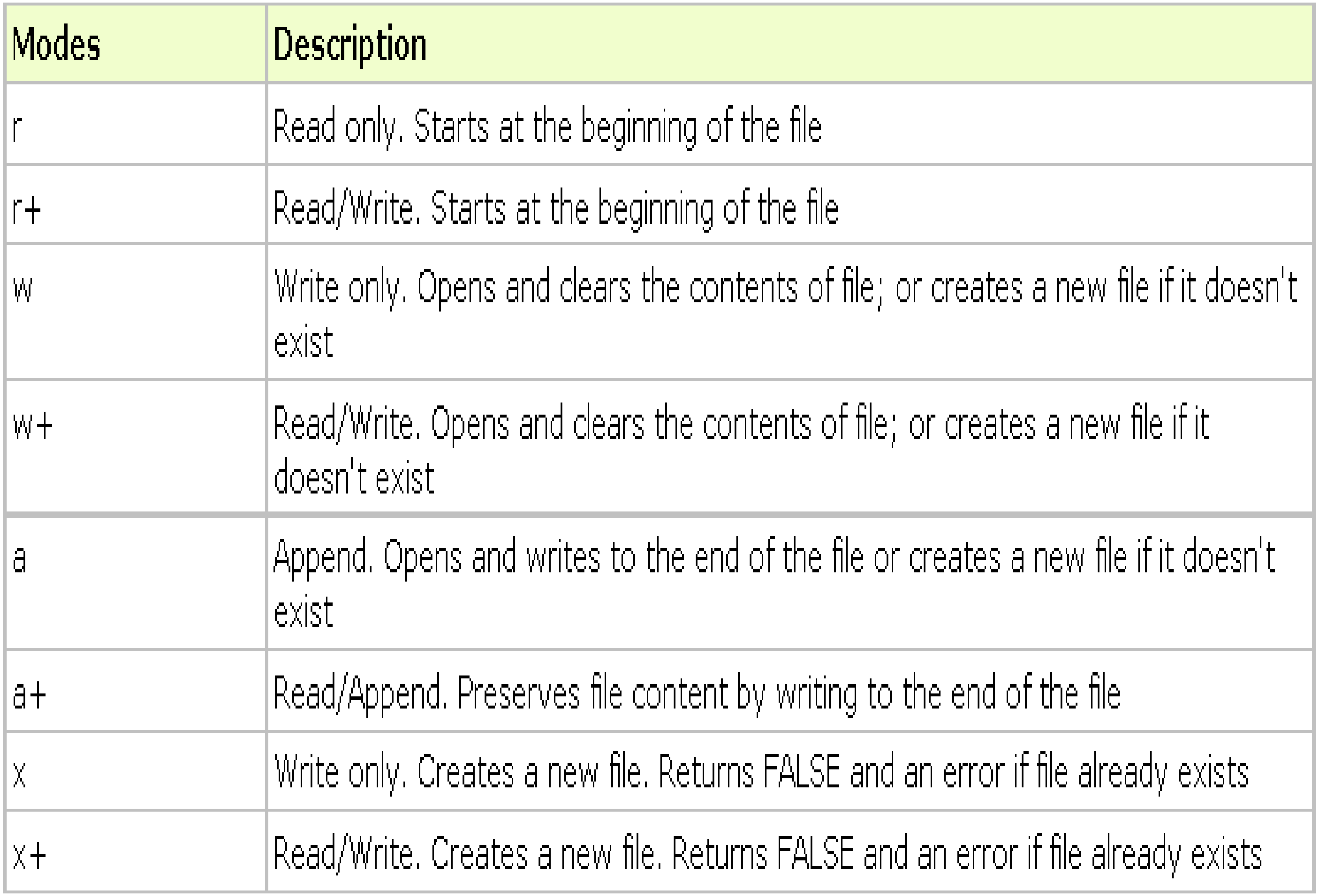
Opening a File…

- The fopen() function is used to open files in PHP
- The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened

```
<html>
<body>

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

</body>
</html>
```



Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

- If the fopen() function is unable to open the specified file, it returns 0 (false)
- The following example generates a message if the fopen() function is unable to open the specified file:

```
<html>
<body>

<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
?>

</body>
</html>
```

Closing a File…

The fclose() function is used to close an open file:

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

//some code to be executed

fclose($file);
?>
```

Check End-of-file…

- The ***feof()*** function checks if the "end-of-file" (EOF) has been reached
- The ***feof()*** function is useful for looping through data of unknown length

Example:

```
if (feof($file)) echo "End of file";
```

Reading a File Line by Line…

- The fgets() function is used to read a single line from a file
- After a call to this function the file pointer will be moved to the next line

Example…

```
<?php
$file = fopen("welcome.txt", "r") or exit("Unable to open
file!");
//Output a line of the file until the end is reached
while(!feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

Reading a File Character by Character…

- The fgetc() function is used to read a single character from a file
- After a call to this function the file pointer moves to the next character

```
<?php
$file=fopen("welcome.txt","r") or exit("Unable to
open file!");
while (!feof($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

Refer p10.php and welcome.txt

PHP File Upload…

- With PHP, it is possible to upload files to the server
- To allow users to upload files from a form can be very useful

```
<html>
<body>

<form action="upload_file.php" method="post“
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>

</body>
</html>
```

Refer p7.html and create a folder /upload inside srinivas

- The enctype attribute of the <form> tag specifies which content-type to use when submitting the form

- "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded

- The type="file" attribute of the <input> tag specifies that the input should be processed as a file

- For example, when viewed in a browser, there will be a browse-button next to the input field

Note:

- Allowing users to upload files is a big security risk
- Only permit trusted users to perform file uploads

- The "upload_file.php" file contains the code for uploading a file:

```
<?php
if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  echo "Type: " . $_FILES[“file"]["type"] . "<br />";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
  echo "Stored in: " . $_FILES["file“]["tmp_name"];
  }
?>
```

- By using the global PHP $_FILES array you can upload files from a client computer to the remote server
- The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp_name" or "error". Like this:
- $_FILES["file"]["name"] - the name of the uploaded file
- $_FILES["file"]["type"] - the type of the uploaded file
- $_FILES["file"]["size"] - the size in bytes of the uploaded file
- $_FILES["file"]["tmp_name"] - the name of the temporary copy of the file stored on the server
- $_FILES["file"]["error"] - the error code resulting from the file upload

- For security reasons, you should add restrictions on what the user is allowed to upload

- In this script we add some restrictions to the file upload

- The user may only upload .gif or .jpeg files and the file size must be under 20 kb:

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
    }
  }
else
  {
  echo "Invalid file";
  }
?>
```

Saving the Uploaded File…

- The examples above create a temporary copy of the uploaded files in the PHP temp folder on the server
- The temporary copied files disappears when the script ends
- To store the uploaded file we need to copy it to a different location

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";
if (file_exists("upload/" . $_FILES["file"]["name"]))
         {
         echo $_FILES["file"]["name"] . " already exists. ";
         }
       else
         {
         move_uploaded_file($_FILES["file"]["tmp_name"],
         "upload/" . $_FILES["file"]["name"]);

     echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
         }
       }
     }
   else
     {
     echo "Invalid file";
     }
   ?>
```

- The script above checks if the file already exists, if it does not, it copies the file to the specified folder

- This example saves the file to a new folder called "upload“

Refer p7.html and upload_file1.php

PHP MySQL Connect to a Database…

- The free MySQL database is very often used with PHP

Create a Connection to a MySQL Database…

- Before you can access data in a database, you must create a connection to the database

- In PHP, this is done with the ***mysql_connect()*** function

Syntax

mysql_connect(servername,username,password);

- servername – Optional - Specifies the server to connect to. Default value is "localhost"
- username - Optional. Specifies the username to log in with. Default value is the name of the user that owns the server process
- password - Optional. Specifies the password to log in with. Default is ""

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code
?>
```

Closing a Connection…

- The connection will be closed automatically when the script ends
- To close the connection before, use the mysql_close() function:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code

mysql_close($con);
?>
```

Insert Data Into a Database Table…

```
<?php
    $con = mysql_connect("localhost","root","");
    if (!$con)
    {
      die("Could not connect: " . mysql_error());
    }
    mysql_select_db("database1", $con);
    mysql_query("INSERT INTO rosh VALUES ('1234', 'Ajay')") or
    die(mysql_error());

    mysql_close($con);
?>
```

Refer p11.php

Insert Data From a Form Into a Database…

```
<html>
<body>

<form action="p13.php" method="post">
Reg No: <input type="text" name="regno" />
Name: <input type="text" name="nm" />

<input type="submit" />
</form>

</body>
</html>
```

Refer p12.html and p13.php

p13.php

```
<html>
<?php
$r = $_POST['regno'];
$n = $_POST['nm'];
$con = mysql_connect("localhost","root","");
if (!$con)
{
    die("Could not connect: " . mysql_error());
}
mysql_select_db("database1", $con);
mysql_query("INSERT INTO rosh VALUES ('$r', '$n')") or
    die(mysql_error());
echo "The data is succesfully inserted!!!";
mysql_close($con);
?>
<br /><a href="p12.html">Go Back</a>
</html>
```

Select Data From a Database Table…

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
{
  die('Could not connect: ' . mysql_error());
}
mysql_select_db("database1", $con);
$result = mysql_query("SELECT * FROM rosh");
while($row = mysql_fetch_array($result))
{
     echo $row['regno'] . " " . $row['name'];
     echo "<br />";
}

mysql_close($con);
?>
```

Refer p14.php

- The example above stores the data returned by the ***mysql_query()*** function in the $result variable

- Next, we use the ***mysql_fetch_array()*** function to return the first row from the recordset as an array

- Each call to mysql_fetch_array() returns the next row in the recordset

- The while loop loops through all the records in the recordset

- To print the value of each row, we use the PHP ***$row*** variable ($row[‘regno'] and $row[‘name']).

Display the Result in an HTML Table…

```
<html>
<?php
    $con = mysql_connect("localhost","root","");
    if (!$con)
    {
    die('Could not connect: ' . mysql_error());
    }
    mysql_select_db("database1", $con);
    $result = mysql_query("SELECT * FROM rosh");
?>
<table border=1 bgcolor="blue" cellspacing = "2" width="30%">
<tr>
<th>Reg No</th>
<th>Name</th>
</tr>
```

```
<?php
while($row = mysql_fetch_array($result))
{
    echo "<tr align=center>";
    echo "<td>" . $row['regno'] . "</td>";
    echo "<td>" . $row['name'] . "</td>";
    echo "</tr>";
}
mysql_close($con);
?>
</table>
</html>
```

Refer p15.php

The WHERE clause…

- The WHERE clause is used to extract only those records that fulfill a specified criterion

Example:

```
<html>
<body>
<form action="p17.php" method="post">
Reg No: <input type="text" name="regno" />
<input type="Submit" />
</form>
</body>
</html>
```

```
<html>
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
{
  die('Could not connect: ' . mysql_error());
}
mysql_select_db("database1", $con);
$r = $_POST['regno'];
$result = mysql_query("SELECT * FROM rosh where regno = '$r'");
if(!($row=mysql_fetch_array($result) ) )
   echo "No record found";
else
   {
?>
<table border=1 bgcolor="blue" cellspacing = "2" width="30%">
<tr>
<th>Reg No</th>
<th>Name</th>
</tr>
<?php
    echo "<tr align=center>";
    echo "<td>" . $row['regno'] . "</td>";
    echo "<td>" . $row['name'] . "</td>";
    echo "</tr>";
    while($row = mysql_fetch_array($result))
    {
            echo "<tr align=center>";
            echo "<td>" . $row['regno'] . "</td>";
            echo "<td>" . $row['name'] . "</td>";
            echo "</tr>";
    }
    mysql_close($con);
}
?>
</table>
<br/><a href="p16.html">Go Back</a>
</html>
```

Refer p16.html and p17.php

Update Data In a Database…

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("UPDATE Persons SET Age = '36'
WHERE FirstName = 'Peter' AND LastName = 'Griffin'");

mysql_close($con);
?>
```

Delete Data In a Database…

- The DELETE FROM statement is used to delete records from a database table

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("DELETE FROM Persons WHERE
LastName='Griffin'");

mysql_close($con);
?>
```

How to connect to MS SQL Server database? (Using DSN)

```
<?php
$myServer = "localhost";
$myUser = "your_name";
$myPass = "your_password";
$myDB = "examples";

//connection to the database

$dbhandle = mssql_connect($myServer, $myUser,
$myPass)  or die("Couldn't connect to SQL Server on
$myServer");
```

```
//select a database to work with

$selected = mssql_select_db($myDB,
$dbhandle)   or die("Couldn't open database
$myDB");

//declare the SQL statement that will query the
database

$query = "SELECT id, name, year ";
$query .= "FROM cars ";
$query .= "WHERE name='BMW'";
```

```
//execute the SQL query and return records
$result = mssql_query($query);
    $numRows = mssql_num_rows($result);
    echo "<h1>" . $numRows . " Row" . ($numRows == 1 ? 
    "" : "s") . " Returned </h1>";

    //display the results
    while($row = mssql_fetch_array($result))
    {
      echo "<li>" . $row["id"] . $row["name"] . $row["year"] . 
    "</li>";
    }
    //close the connection
    mssql_close($dbhandle);
    ?>
```

How to connect to MS SQL Server database? (without using DSN)

```
<?php
$myServer = "localhost";
$myUser = "your_name";
$myPass = "your_password";
$myDB = "examples";

//create an instance of the  ADO connection object
$conn = new COM ("ADODB.Connection")
 or die("Cannot start ADO");
```

//define connection string, specify database driver

$connStr = "PROVIDER=SQLOLEDB;SERVER=".$myServer."; UID=".$myUser.";PWD=".$myPass.";DATABASE=". $myDB;

$conn->open($connStr); //Open the connection to the database

```
//declare the SQL statement that will query the
database

$query = "SELECT * FROM cars";

//execute the SQL statement and return records
$rs = $conn->execute($query);

$num_columns = $rs->Fields->Count();
echo $num_columns . "<br>";

for ($i=0; $i < $num_columns; $i++) {
    $fld[$i] = $rs->Fields($i);
}
```

```
echo "<table>";
while (!$rs->EOF)
//carry on looping through while there are records
  {
      echo "<tr>";
      for ($i=0; $i < $num_columns; $i++) {
          echo "<td>" . $fld[$i]->value . "</td>";
      }
      echo "</tr>";
      $rs->MoveNext(); //move on to the next record
  }
  echo "</table>";
```

```
//close the connection and recordset objects
freeing up resources

$rs->Close();
$conn->Close();

$rs = null;
$conn = null;
?>
```

```
CREATE DATABASE examples;

USE examples;

CREATE TABLE cars(
   id int UNIQUE NOT NULL,
   name varchar(40),
   year varchar(50),
   PRIMARY KEY(id)
);

INSERT INTO cars VALUES(1,'Mercedes','2000');
INSERT INTO cars VALUES(2,'BMW','2004');
INSERT INTO cars VALUES(3,'Audi','2001');
```

PHP and XML…

What is XML?

- XML is used to describe data and to focus on what data is
- An XML file describes the structure of the data
- In XML, no tags are predefined; You must define your own tags

An XML File…

- The XML file below will be used in our example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

PHP XML DOM…

What is DOM?

- The W3C DOM provides a standard set of objects for HTML and XML documents, and a standard interface for accessing and manipulating them

- The W3C DOM is separated into different parts (Core, XML, and HTML) and different levels (DOM Level 1/2/3):

  * Core DOM - defines a standard set of objects for any structured document

  * XML DOM - defines a standard set of objects for XML documents

  * HTML DOM - defines a standard set of objects for HTML documents

XML Parsing

- To read and update - create and manipulate - an XML document, you will need an XML parser
- There are two basic types of XML parsers:
- ***Tree-based parser:*** This parser transforms an XML document into a tree structure. It analyzes the whole document, and provides access to the tree elements
- ***Event-based parser:*** Views an XML document as a series of events. When a specific event occurs, it calls a function to handle it

- The DOM parser is an tree-based parser
- Look at the following XML document fraction:

  <?xml version="1.0" encoding="ISO-8859-1"?>
  <from>Jani</from>

- The XML DOM sees the XML above as a tree structure:

Level 1: XML Document

Level 2: Root element: <from>

Level 3: Text element: "Jani"

- **An XML File**

- The XML file below will be used in our example:
-
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- **Load and Output XML**
- We want to initialize the XML parser, load the xml, and output it:

- **Example**

-
  ```
  <?php
  $xmlDoc = new DOMDocument();
  $xmlDoc->load("note.xml");

  print $xmlDoc->saveXML();
  ?>
  ```
- The output of the code above will be:
- Tove Jani Reminder Don't forget me this weekend!

- **Looping through XML**
- We want to initialize the XML parser, load the XML, and loop through all elements of the <note> element:
- **Example**
-
```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
  {
  print $item->nodeName . " = " . $item->nodeValue .
"<br />";
  }
?>
```

Refer p18.php

- The output of the code above will be:
- #text =
  to = Tove
  #text =
  from = Jani
  #text =
  heading = Reminder
  #text =
  body = Don't forget me this weekend!
  #text = In the example above you see that there are empty text nodes between each element

- When XML generates, it often contains white-spaces between the nodes. The XML DOM parser treats these as ordinary elements, and if you are not aware of them, they sometimes cause problems.

PHP SimpleXML…

What is SimpleXML?

- Elements - Are converted to single attributes of the SimpleXMLElement object. When there's more than one element on one level, they're placed inside an array
- Attributes - Are accessed using associative arrays, where an index corresponds to the attribute name
- Element Data - Text data from elements are converted to strings. If an element has more than one text node, they will be arranged in the order they are found
- SimpleXML is fast and easy to use when performing basic tasks like:
- Reading XML files
- Extracting data from XML strings
- Editing text nodes or attributes
- However, when dealing with advanced XML, like namespaces, you are better off using the Expat parser or the XML DOM.

- **Using SimpleXML**
- Below is an XML file:
- <?xml version="1.0" encoding="ISO-8859-1"?>
  <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>

- We want to output the element names and data from the XML file above.
- Here's what to do:
- Load the XML file
- Get the name of the first element
- Create a loop that will trigger on each child node, using the children() function
- Output the element name and data for each child node

- **Example**
-
```
<?php
$xml = simplexml_load_file("test.xml");

echo $xml->getName() . "<br />";

foreach($xml->children() as $child)
  {
  echo $child->getName() . ": " . $child . "<br />";
  }
?>
```
  The output of the code above will be:

- Refer p19.php

Thank You…

☺