PHP

PHP

- PHP is a powerful server-side scripting language for creating dynamic and interactive websites.
- PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP is perfectly suited for Web development and can be embedded directly into the HTML code.
- The PHP syntax is very similar to Perl and C. PHP is often used together with Apache (web server) on various operating systems. It also supports ISAPI and can be used with Microsoft's IIS on Windows.

Introduction to PHP

- A PHP file may contain text, HTML tags and scripts. Scripts in a PHP file are executed on the server.

What is PHP?

- PHP stands for **P**HP: **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, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS) PHP is free to download and use

What is a PHP File?

- PHP files may contain text, HTML tags and scripts. PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

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

PHP + MySQL

- PHP combined with MySQL are cross-platform (means that 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

Where to Start?

- Install an Apache server on a Windows or Linux machine
- Install PHP on a Windows or Linux machine Install MySQL on a Windows or Linux machine

PHP Installation

- If your server supports PHP - you don't need to do anything! You do not need to compile anything or install any extra tools - just create some .php files in your web directory - and the server will parse them for you. Most web hosts offer PHP support.
- However, if your server does not support PHP, you must install PHP. Below is a link to a good tutorial from PHP.net on how to install PHP5: http://www.php.net/manual/en/install.php

Download PHP

- Download PHP for free here: http://www.php.net/downloads.php **Download MySQL Database** Download MySQL for free here:
- http://www.mysql.com/downloads/index.html

Download Apache Server

- Download Apache for free here: http://httpd.apache.org/download.cgi

PHP

PHP Syntax

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.
- On servers with shorthand support enabled you can start a scripting block with <? And end with ?>.
- However, for maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

<?php
?>

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>
```

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.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have

used the echo statement to output the text "Hello World".

Comments in PHP

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

```
<html>
<body>
```

PHP

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

PHP Variables

- Variables are used for storing values, such as numbers, strings or function results, so that they can be used many times in a script.

Variables in PHP

- All variables in PHP start with a $ sign symbol. Variables may contain strings, numbers, or arrays.
- Below, the PHP script assigns the string "Hello World" to a variable called $txt:

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

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

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

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

The output of the script above will be: "Hello World 1234".

Variable Naming Rules

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- A variable name should not contain spaces.
- If a variable name should be more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)

PHP Operators

- Operators are used to operate on values.

PHP Operators

- This section lists the different operators used in PHP.

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

PHP

++	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

Comparison Operators

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

PHP

>=	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

PHP If...Else Statements

- The if, elseif and else statements in PHP are used to perform different actions based on different conditions.

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.

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 several condition are true

Syntax:

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

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

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

The Elseif Statement

If you want to execute some code if one of several conditions are true use the elseif statement.

Syntax :

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;

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<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>
```

PHP Switch Statement

The Switch statement in PHP is used to perform one of several different actions based on one of several different conditions.

The Switch Statement

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.

Syntax

```
switch (expression)
{
case label1:
```

```
        code to be  executed    if expression
= label1;
   break;
case label2:
        code to be  executed        if expression
= label2;
   break;
default:
   code to be executed
if expression is different  from both label1 and
label2;

}
```

This is how it works:

A single expression (most often a variable) is evaluated once

The value of the expression is compared with the values for each case in the structure

If there is a match, the code associated with that case is executed

After a code is executed, **break** is used to stop the code from running into the next case

The default statement is used if none of the cases are true.

Example:

```
<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

An array can store one or more values in a single variable name.

What is an array?

When working with PHP, sooner or later, you might want to create many similar variables.

Instead of having many similar variables, you can store the data as elements in an array.

Each element in the array has its own ID so that it can be easily accessed.

There are three different kinds of arrays:

Numeric array - An array with a numeric ID key

Associative array - An array where each ID key is associated with a value

Multidimensional array - An array containing one or more arrays

Numeric Arrays

A numeric array stores each element with a numeric ID key.

There are different ways to create a numeric array.

Example 1

In this example the ID key is automatically assigned:

$names = array("Peter","Quagmire","Joe");

Example 2

In this example we assign the ID key manually:

$names[0] = "Peter";

$names[1] = "Quagmire";

$names[2] = "Joe";

The ID keys can be used in a script:

```
<?php
$names[0]
$names[1]
$names[2] = "Joe";
echo $names[1] . " and".
$names[2] .
" are ". $names[0] . "'s neighbours";
?>
```

The code above will output:

Quagmire and Joe are Peter's neighbours

Associative Arrays

An associative array, each ID key is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

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

Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

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

The code above will output:

Peter is 32 years old.

Multidimensional Arrays

In a multidimensional 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.

Example

PHP

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array (
"Griffin"=>array
(
"Peter",
"Lois",
"Megan",
),
"Quagmire"=>array (
"Glenn"
),
"Brown"=>array (
"Cleveland",
"Loretta", "Junior"
)
);
```

The array above would look like this if written to the output:

```
Array (
[Griffin] => Array
(
[0] => Peter
[1] => Lois
[2] => Megan
)
[Quagmire] => Array (
[0] => Glenn
)
[Brown] => Array (
[0] => Cleveland
[1] => Loretta
[2] => Junior
)
)
```

PHP Looping

PHP

Looping statements in PHP are used to execute the same block of code a specified number of times.

Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

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

for - loops through a block of code a specified number of times

foreach - loops through a block of code for each element in an array

The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

Syntax

while (*condition*)

code to be executed;

Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
```

```
{
echo "The number is " . $i . "<br/>";
$i++;
}
?>
</body>
</html>
```

The do...while Statement

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.

Syntax

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

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 5:

```
<html>
<body>
<?php
$i=0;
do{
$i++;
echo "The number is " . $i . "<br/>";
}
while ($i<5);
?>
</body>
</html>
```

The for Statement

The for statement is used when you know how many times you want to execute a statement or a list of statements.

Syntax:

for (*initialization*; *condition*; *increment*)
{
code to be executed;
}

Note: 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.

Example

The following example prints the text "Hello World!" five times:

```
<html>
<body>
<?php
for ($i=1;$i<=5; $i++)
{
echo "Hello World!<br />";
}
?>
</body>
</html>
```

The foreach Statement

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

Syntax

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

Example

The following example demonstrates a loop that will print the values of the given array:

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

PHP Functions

The real power of PHP comes from its functions.

In PHP - there are more than 700 built-in functions available.

PHP Functions

In this tutorial we will show you how to create your own functions.

For a reference and examples of the built-in functions, please visit our PHP Reference.

Create a PHP Function

A function is a block of code that can be executed whenever we need it.

Creating PHP functions:

PHP

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)

Add a "{" - The function code starts after the opening curly brace

Insert the function code

Add a "}" - The function is finished by a closing curly brace

Example:

A simple function that writes my name when it is called:

```
<html>
<body>
<?php
function writeMyName()
{
echo "Kai Jim Refsnes";
}
writeMyName();
?>
</body>
</html>
```

Use a PHP Function

Now we will use the function in a PHP script:

```
<html>
<body>
<?php
function writeMyName()
{
echo "Kai Jim Refsnes";
}
echo "Hello world!<br />";
echo "My name is ";  writeMyName();  echo ".<br />That's right, ";
```

```
writeMyName();
echo " is my name.";
?>
</body>
</html>
```

The output of the code above will be:

Hello world!

My name is Kai Jim Refsnes.

That's right, Kai Jim Refsnes is my name.

PHP Functions - Adding parameters

Our first function (writeMyName()) is a very simple function. It only writes a static string.

To add more functionality to a function, we can add parameters.

A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.

Example 1

The following example will write different first names, but the same last name:

```
<html>
<body>
<?php
function writeMyName($fname)
{
echo $fname .  " Refsnes.<br />";
}
echo "My name is ";  writeMyName("Kai Jim");
echo "My name is ";writeMyName("Hege");
echo "My name is ";writeMyName("Stale");
?>
```

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

The output of the code above will be:

```
My  name  is  Kai Jim Refsnes.
My  name  is  Hege Refsnes.
My  name  is  Stale Refsnes.
```

The following function has two parameters:

```
<html>
<body>
<?php
function writeMyName($fname,$punctuation)
{
echo $fname . " Refsnes" .
$punctuation . "<br />";
}
echo "My name is ";  writeMyName("Kai Jim",".");  echo "My name is ";
writeMyName("Hege","!");
echo "My name is ";  writeMyName("Ståle","...");

?>

</body>

</html>
```

The output of the code above will be:

```
My  name  is  Kai Jim Refsnes.
My  name  is  Hege Refsnes!
My  name  is  Ståle Refsnes...
```

PHP Functions - Return values Functions can also be used to return values.

Example

```
<html>
<body>
<?php
```

```
Function add($x,$y)
{
$total = $x + $y;
Return $total;
}
echo "1 + 16 = " . add(1,16) ;
?>
</body>
</html>
```

The output of the code above will be:

1 + 16 = 17

PHP Forms and User Input

The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.

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.

Form example:

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

The example HTML page above contains two input fields and a submit button. When the user fills in this form and click on the submit button, the form data is sent to the "welcome.php" file.

The "welcome.php" file looks like this:

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

A sample output of the above script may be:

Welcome John.

You are 28 years old.

The PHP $_GET and $_POST variables will be explained in the next chapters.

Form Validation

User input should be validated on the browser whenever possible (by client scripts (JavaScript)). Browser validation is faster and you reduce the server load.

You should consider using server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

PHP $_GET

The $_GET variable is used to collect values from a form with method="get".

The $_GET Variable

The $_GET variable is an array of variable names and values sent by the HTTP GET method.

The $_GET variable is used to collect values from a form 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 it has limits on the amount of information to send (max. 100 characters).

Example

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

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

The "welcome.php" file can now use the $_GET variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_GET array):

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

Why use $_GET?

Note: When using the $_GET variable all variable names and values are displayed in the URL. So 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.

Note: The HTTP GET method is not suitable on large variable values; the value cannot exceed 100 characters.

PHP $_POST

The $_POST variable is used to collect values from a form with method="post".

The $_POST Variable

The $_POST variable is an array of variable names and values sent by the HTTP POST method.

The $_POST variable is used to collect values from a form 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.

Example

<form action="welcome.php" method="post">

Enter your name: <input type="text" name="name" />

Enter your age: <input type="text" name="age" />

<input type="submit" />

</form>

When the user clicks the "Submit" button, the URL will not contain any form data, and will look something like this:

http://www.w3schools.com/welcome.php

The "welcome.php" file can now use the $_POST variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_POST array):

Welcome <?php echo $_POST["name"];

?>.

You are <?php echo $_POST["age"];

?> years old!

Why use $_POST?

Variables sent with HTTP POST are not shown in the URL

Variables have no length limit

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

The $_REQUEST Variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Example

```
Welcome

<?php

Echo $_REQUEST["name"];

?>.<br /> You are

<?php

Echo $_REQUEST["age"]; ?> years old!
```

PHP Date()

The PHP date() function is used to format a time or a date.

The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

```
date(format,timestamp)
```

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time (as a timestamp)

PHP Date - What is a Timestamp?

A timestamp is the number of seconds since January 1, 1970 at 00:00:00 GMT. This is also known as the Unix Timestamp.

PHP Date - Format the Date

The first parameter in the date() function specifies how to format the date/time. It uses letters to represent date and time formats. Here are some of the letters that can be used:

d - The day of the month (01-31)

m - The current month, as a number (01-12)

Y - The current year in four digits

An overview of all the letters that can be used in the format parameter, can be found in our PHP Date reference.

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

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

The output of the code above could be something like this:

2006/07/11

2006.07.11

2006-07-11

PHP Date - Adding a Timestamp

The second parameter in the date() function specifies a timestamp. This parameter is optional. If you do not supply a timestamp, the current time will be used.

In our next example we will use the mktime() function to create a timestamp for tomorrow.

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

Syntax

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

To go one day in the future we simply add one to the day argument of mktime():

```
<?php

$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y") echo "Tomorrow is ".date("Y/m/d/", $tomorrow

?>
```

The output of the code above could be something like this:

Tomorrow is 2006/07/12

PHP Include File

Server Side Includes (SSI) are used to create functions, headers, footers, or elements that will be reused on multiple pages.

Server Side Includes

You can insert the content of a file into a 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. The include() function generates a warning (but the script will continue execution)

while the require() function generates a fatal error (and the script execution will stop after the error).

These two functions are used to create functions, headers, footers, or elements that can be reused on multiple pages.

This can save the developer a considerable amount of time. This means that you can create a standard header or menu file that you want all your web pages to include. When the header needs to be updated, you can only update this one 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 web pages).

The include() Function

The include() function takes all the text in a specified file and copies it into the file that uses the include function.

Example 1

Assume that you have a standard header file, called "header.php". To include the header file in a page, use the include() function, like this:

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

The require() Function

The require() function is identical to include(), they only handle errors differently.

The include() function generates a warning (but the script will continue execution) while the require() function generates a fatal error (and the script execution will stop after the error).

If you include a file with the include() function and an error occurs, you might get an error message like the one below.

PHP

PHP code:

```
<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 still executed! This is because a Warning does not stop the script execution.

in C:\home\website\test.php on line 5

The echo statement was 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 continue executing if files are missing or misnamed.

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>
```

The file may be opened in one of the following modes:

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

Note: If the fopen() function is unable to open the specified file, it returns 0 (false).

Example

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.

Note: You cannot read from files opened in w, a, and x mode!

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.

Note: After a call to this function the file pointer has moved to the next line.

Example

The example below reads a file line by line, until the end of file is reached:

```
<?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.

Note: After a call to this function the file pointer moves to the next character.

Example

The example below reads a file character by character, until the end of file is reached:

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

PHP File Upload

With PHP, it is possible to upload files to the server.

Create an Upload-File Form

To allow users to upload files from a form can be very useful.

Look at the following HTML form for uploading files:

```
<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>
```

Notice the following about the HTML form above:
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 possessed 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.

Create The Upload Script

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

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

Restrictions on 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")
&& ($_FILES["file"]["size"] < 20000))
{
        if ($_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") &&  ($_FILES["file"]["size"] < 20000))
{
        if ($_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.

PHP Cookies

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests for a page with a browser, it will send the cookie too.

With PHP, you can both create and retrieve cookie values.

How to Create a Cookie?

The setcookie() function is used to set a cookie.

Note: The setcookie() function must appear BEFORE the <html> tag.

Syntax

setcookie(name, value, expire, path, domain);

Example

In the example below, we will create a cookie named "user" and assign the value "Alex also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>
<html>
<body>
</body>
</html>
```

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

How to Retrieve a Cookie Value?

The PHP $_COOKIE variable is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
// Print a cookie
echo $_COOKIE["user"];
// A way to view all cookies print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set:

```
<html>
<body>
<?php
if (isset($_COOKIE["user"]))
echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
```

```
echo "Welcome guest!<br
/>";
?>
</body>
</html>
```

How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is in the past.

Delete example:

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

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your

application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).

asses the user input to "welcome.php" when the user clicks on the "Submit" button:

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

Retrieve the values in the "welcome.php" file like this:

```
<html>
<body>
```

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

PHP Sessions

A PHP session variable is used to store information about, or change settings for a user session.

Session variables hold information about one single user, and are available to all pages in one application.

PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted

after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>
<html>
<body>
</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//retrieve session data echo "Pageviews=".
$_SESSION['views'];
?>
</body>
</html>
```

Output:

Pageviews=1

In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();
if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function.

The unset() function is used to free the specified session variable:

```
<?php unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

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

Note: session_destroy() will reset your session and you will lose all your stored session data.

PHP Sending E-mails

PHP allows you to send e-mails directly from a script.

The PHP mail() Function

The PHP mail() function is used to send emails from inside a script.

Syntax

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

Parameter	Description

to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. Note: 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 (\r\n)
parameters	Optional. Specifies an additional parameter to the sendmail program

Note: For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file. Read more in our PHP Mail reference.

PHP Simple E-Mail

The simplest way to send an email with PHP is to send a text email.

In the example below we first declare the variables ($to, $subject, $message, $from, $headers), then we use the variables in the mail() function to send an e- mail:

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple
email message.";
$from = "someonelse@example.com";
$headers = "From: $from"; mail($to,$subject,$message,$headers);
echo "Mail Sent.";
```

```
?>
```

PHP MySQL Introduction

MySQL is the most popular open source database server.

What is MySQL?

A database defines a structure for storing information.

In a database, there are tables. Just like HTML tables, database tables contain rows, columns, and cells.

Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

Database Tables

A database most often contains one or more tables. Each table has a name (e.g. "Customers" or "Orders"). Each table contains records (rows) with data.

Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

The table above contains three records (one for each person) and four columns (LastName, FirstName, Address, and City).

Queries

A query is a question or a request.

With MySQL, we can query a database for specific information and have a recordset returned.

PHP

Look at the following query:

SELECT LastName FROM Persons

The query above selects all the data in the LastName column in the Persons table, and will return a recordset like this:

LastName
Hansen
Svendson
Pettersen

Download MySQL Database

If you don't have a PHP server with a MySQL Database, you can download MySQL for free here: http://www.mysql.com/downloads/index.html

Facts About MySQL Database

One great thing about MySQL is that it can be scaled down to support embedded database applications.

Perhaps it is because of this reputation that many people believe that MySQL can only handle small to

medium-sized systems.

The truth is that MySQL is the de-facto standard database for web sites that support huge volumes of both data and end users (like Friendster, Yahoo, Google). Look at http://www.mysql.com/customers/ for an overvirew of companies that uses MySQL.

PHP MySQL Connect to a Database

The free MySQL Database is very often used with PHP.

Connecting to a MySQL Database

Before you can access and work with 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);

Parameter	Description
servername	Optional. Specifies the server to connect to. Default value is "localhost:3306"
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 ""

Note: There are more available parameters, but the ones listed above are the most important. Visit our full PHP MySQL Reference for more details.

Create a Table

The CREATE TABLE statement is used to create a database table in MySQL.

Syntax

```
CREATE TABLE table_name (
column_name1 data_type,
column_name2 data_type, column_name3 data_type,
.....
)
```

We must add the CREATE TABLE statement to the mysql_query() function to execute the command.

Example

The following example shows how you can create a table named "Person", with three columns. The column names will be "FirstName", "LastName" and "Age":

```
<?php
```

```
$con = mysql_connect("localhost","peter","abc123"); if (!$con)
  {
  die('Could not connect: ' .
mysql_error());
}
// Create database
if (mysql_query("CREATE DATABASE
my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " .
mysql_error();
  }
// Create table in my_db database
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE Person
(
FirstName varchar(15),
LastName varchar(15),
Age int
)";
mysql_query($sql,$con);
mysql_close($con);
?>
```

Important: A database must be selected before a table can be created. The database is selected with the mysql_select_db() function.

Note: When you create a database field of type varchar, you must specify the maximum length of the field, e.g. varchar(15).

MySQL Data Types

Below is the different MySQL data types that can be used:

Numeric Data Types	**Description**

int(size) smallint(size) tinyint(size) mediumint(size) bigint(size)	Hold integers only. The maximum number of digits can be specified in the size parameter
decimal(size,d) double(size,d) float(size,d)	Hold numbers with fractions. The maximum number of digits can be specified in the size parameter. The maximum number of digits to the right of the decimal is specified in the d

Textual Data Types	**Description**
char(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis
varchar(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis
tinytext	Holds a variable string with a maximum length of 255 characters
text blob	Holds a variable string with a maximum length of 65535 characters
mediumtext mediumblob	Holds a variable string with a maximum length of 16777215 characters
longtext longblob	Holds a variable string with a maximum length of 4294967295 characters

Date Data Types	**Description**

date(yyyy-mm-dd) datetime(yyyy-mm-dd hh:mm:ss) timestamp(yyyymmddhhmmss) time(hh:mm:ss)	Holds date and/or time

Misc. Data Types	**Description**
Enum(value1,value2,ect)	ENUM is short for ENUMERATED list. Can store one of up to 65535 values listed within the ( ) brackets. If a value is inserted that is not in the list, a blank value will be inserted
Set	SET is similar to ENUM. However, SET can have up to 64 list items and can store more than one choice

Primary Keys and Auto Increment Fields

Each table should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The primary key field is always indexed. There is no exception to this rule! You must index the primary key field so the database engine can quickly locate rows based on the key's value.

The following example sets the personID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting.

AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be

null, we must add the NOT NULL setting to the field.

Example

```
$sql = "CREATE TABLE Person (
personID int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(personID),
FirstName varchar(15),
LastName varchar(15), Age int
)";
mysql_query($sql,$con);
```

PHP MySQL Insert Into

Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

Syntax

```
INSERT INTO table_name VALUES (value1, value2,....)
```

You can also specify the columns where you want to insert the data:

```
INSERT INTO table_name (column1, column2,...)

VALUES (value1, value2,....)
```

Note: SQL statements are not case sensitive. INSERT INTO is the same as insert into.

PHP

To get PHP to execute the statements above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example

In the previous chapter we created a table named "Person", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example. The following example adds two new records to the "Person" table:

```
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("INSERT INTO person (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin', '35')");

mysql_query("INSERT
INTO person
(FirstName,
LastName, Age)
VALUES ('Glenn', 'Quagmire',
'33')");
mysql_close($con);
?>
```

PHP Database ODBC

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database).

PHP

Create an ODBC Connection

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is how to create an ODBC connection to a MS Access Database:

1. Open the **Administrative Tools** icon in your Control Panel.
2. Double-click on the **Data Sources (ODBC)** icon inside.
3. Choose the **System DSN** tab.
4. Click on **Add** in the System DSN tab.
5. Select the **Microsoft Access Driver**. Click **Finish.**
6. In the next screen, click **Select** to locate the database.
7. Give the database a **Data Source Name (DSN)**.
8. Click **OK**.

Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to to set up a DSN for you to use.

Connecting to an ODBC

The odbc_connect() function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

The odbc_exec() function is used to execute an SQL statement.

Example

The following example creates a connection to a DSN called northwind, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('northwind','','');
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
```

Retrieving Records

PHP

The odbc_fetch_rows() function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false.

The function takes two parameters: the ODBC result identifier and an optional row number:

odbc_fetch_row($rs)

Retrieving Fields from a Record

The odbc_result() function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.

The code line below returns the value of the first field from the record:

$compname=odbc_result($rs,1);

The code line below returns the value of a field called "CompanyName":

$compname=odbc_result($rs,"CompanyName");

Closing an ODBC Connection

The odbc_close() function is used to close an ODBC connection.

odbc_close($conn);

An ODBC Example

The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
if (!$rs)
{exit("Connection Failed: " .$conn);}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while(odbc_fetch_row($rs))
{
echo
```

PHP

```
$compname=odbc_result($rs,"CompanyNam
e");
$conname=odbc_result($rs,"ContactNam
  e");  echo
  "<tr><td>$compname</td>";
  echo "<td>$conname</td></tr>";
}
odbc_close($co
nn);  echo
"</table>";
?>
</body>
</html>
```