



ARULMIGU PALANIANDAVR ARTS COLLEGE FOR WOMEN

(Autonomous)

(Re-Accredited with 'B⁺⁺' Grade by NAAC 3rd Cycle)

Run by Arulmigu Dhandayuthapani Swamy Thirukoil, H.R & C.E Dept. Government of Tamil Nadu

A Government Aided College - Affiliated to Mother Teresa Women's University, Kodaikanal

CHINNAKALAYAMPUTHUR(PO), PALANI - 624615



DEPARTMENT OF COMPUTER SCIENCE

AND

COMPUTER APPLICATION

LEARNING RESOURCE



Introduction to PHP & Features

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

Example

```
<html>
<body>

<?php
echo "My first
PHPscript!"; ?>
</body>
</html>
```

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- ❖ PHP can generate dynamic page content
- ❖ PHP can create, open, read, write, delete, and close files on the server
- ❖ PHP can collect form data
- ❖ PHP can send and receive cookies
- ❖ PHP can add, delete, modify data in your database
- ❖ PHP can be used to control user-access
- ❖ PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host with PHP Support

- If your server has activated support for PHP you do not need to do anything.
- Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.
- Because PHP is free, most web hosts offer PHP support.
- Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

PHP Scripts

Basic PHP syntax

- ✚ A PHP script can be placed anywhere in the document.
- ✚ A PHP script starts with **<?php** and ends with **?>**:

<?php

// PHP code goes here ?>

Example

```
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

- PHP statements end with a semicolon (;)

Comments in PHP

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.

Comments can be used to:

- Let others understand what you are doing
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

Example

```
<html>

<body>

<?php

// This is a single-line comment

# This is also a single-
linecomment /*

This is a multiple-lines
commentblock that spans over
multiple lines

*/

// You can also use comments to leave out parts of a code line

$x = 5 /* + 15 */ +
5;echo
$x; ?>

</body>

</html>
```

Example

```
<html>

<body>

<?php

ECHO "Hello
World!<br>";echo
"Hello World!<br>";
EcHo "Hello
World!<br>";
```

?></body>

In the example below, only the first statement will display the value of the \$color variable (this is because \$color, \$COLOR, and \$coLOR are treated as three different variables):

Example

```
<html>
<body>

<?php

$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR .
"<br>"; echo "My boat is " .
$COLOR . "<br>"; ?>

</body>

</html>
```

Data Types

- Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = "Hello world!";
```

```
$y = 'Hello
```

```
world!';echo $x;
```

```
echo
```

```
"<br>";
```

```
echo $y;
```

```
?>
```

```
</body>
```

```
</html>
```

OUTPUT:

```
Hello  
world!  
Hello  
world!
```


String Functions

Get The Length of a String

- The PHP `strlen()` function returns the length of a string.
- The example below returns the length of the string "Hello world!":

Example

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

OUTPUT:

12

Count The Number of Words in a String

The PHP `str_word_count()` function counts the number of words in a string:

Example

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

Reverse a String

- The PHP `strrev()` function reverses a string:

Example

<html>

<body>

<?php

```
<?php
echo str_word_count("Hello world!");
?>
</body>
</html>
```

OUTPUT:

2

Search For a Specific Text Within a String

- The PHP `strpos()` function searches for a specific text within a string.
- If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
- The example below searches for the text "world" in the string "Hello world!":

Example

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

OUTPUT:

6

Replace Text Within a String

- The PHP `str_replace()` function replaces some characters with some other characters in a string.
- The example below replaces the text "world" with "Dolly":

Example

```
<html>  
  
<body>  
  
<?php  
echo str_replace("world", "Dolly", "Hello world!");  
?>  
  
</body>  
</html>
```

OUPUT:

Hello Dolly!

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- In the following example \$x is an integer. The PHP var_dump() function returns the data type and value:

Example

```
<html>
<body>
<?php
$x = 5985;
var_dump($x);
?>
</body>
</html>
```

OUTPUT:

```
int(5985)
```

PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example \$x is a float. The PHP var_dump() function

returns the data type and value:

Example

```
<html>
<body>
<?php
$x = 10.365;
var_dump($x);
?>
</body>
</html>
```

OUTPUT:

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
```

```
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

- An array stores multiple values in one single variable:
- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the

cars in single variables could look like this:

```
$cars1 = "Volvo";
```

```
$cars2 = "BMW";
```

```
$cars3 = "Toyota";
```

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The solution is to create an array!
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
$cars = array("Volvo", "BMW", "Toyota");
```

```
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . " .";
```

```
?>
```

```
</body>
```

```
</html>
```

OUTPUT:

I like Volvo, BMW and Toyota.

Create an Array in PHP

- In PHP, the `array()` function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

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

Variables

- Variables are "containers" for storing information.
- Creating (Declaring) PHP Variables
- In PHP, a variable starts with the \$ sign, followed by the name of the variable:

Example

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

```
$txt = "Hello world!";
```

```
$x = 5;  
$y = 10.5;
```

```
echo $txt;  
echo "<br>";  
echo $x;  
echo "<br>";  
echo $y;  
?>  
</body>  
</html>
```

Output:

Hello world! 5 10.5

Rules for PHP variables:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)

Output Variables

- The PHP `echo` statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

```
<html>
<body>
<?php
$txt =
"W3Schools.com";
echo "I love $txt!";
?>
</body>
</html>
```

Output:

I love W3Schools.com!

Example

```
<html>
<body>
<?php
$txt =
"W3Schools.com";
echo "I love " . $txt .
"!";
?>
```

Output:

I love W3Schools.com!

```
</body>
```

```
</html>
```

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = 5;
```

Output:

9

```
$y = 4echo $x + $y;
```

```
?>
```

```
</body>
```

```
</html>
```

PHP Variables Scope

- 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
- global
- static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<html>
<body>
<?php
$x = 5; // global
scopefunction
myTest() {
    // using x inside this function will generate an
    errecho "<p>Variable x inside function is:
    $x</p>";
}
myTest();
echo "<p>Variable x outside function is:
$x</p>"; ?>
</body>
</html>
```

OUTPUT:

Variable x inside function is:

Variable x outside function is: 5

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

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

```
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate
anerror echo "<p>Variable x outside
function is:
$x</p>"; ?>
</body>
</html>
```

OUTPUT:

Variable x inside function
is: 5
Variable x outside
function is:

The global Keyword

- The `global` keyword is used to access a global variable from within a function.
- To do this, use the `global` keyword before the variables (inside the function):

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = 5;
```

```
$y = 10;
```

```
function
```

```
myTest() {
```

```
global $x, $y; $y
```

```
    = $x + $y;
```

```
}
```

```
myTest(); // run function
```

```
echo $y; // output the new value for variable
```

```
$y ?>
```

```
</body>
```

```
</html>
```

Output:

15

The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the `static` keyword when you first declare the variable:

Example

```
<html>
<body>
<?php
function
    myTest() {
        static $x = 0;
        echo $x;
        $x++;
    }
myTest();
echo
"<br>";
myTest();
echo
"<br>";
myTest();
?>
</body>
</html>
```

Output:

```
0
1
2
```

echo Statement

The `echo` statement can be used with or without parentheses: `echo` or `echo()`.

Display Text

The following example shows how to output text with the `echo` command (notice that the text can contain HTML markup):

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
multipleparameters."; ?>
</body>
</html>
```

OUTPUT:

PHP is Fun!

Hello world!

I'm about to learn PHP!

This string was made with multiple parameters.

Display Variables

The following example shows how to output text and variables with the `echo` statement:

Example

```
<html>

<body>

<?php

$txt1 = "Learn PHP"; $txt2

= "W3Schools.com";

$x = 5;

$y = 4;

echo "<h2>" . $txt1 . "</h2>";

echo "Study PHP at " . $txt2 . "<br>";

echo $x + $y;

?>

</body>

</html>
```

OUTPUT:

Learn PHP

Study PHP at
W3Schools.com
9

The PHP print Statement

- The `print` statement can be used with or without parentheses: `print` or `print()`.

Display Text

The following example shows how to output text with the `print` command (notice that the text can contain HTML markup):

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
print "<h2>PHP is  
Fun!</h2>";print "Hello  
world!<br>";
```

```
print "I'm about to  
learnPHP!"; ?>
```

```
</body>
```

```
</html>
```

OUTPUT:

PHP is Fun!

Hello world!

I'm about to learn PHP

PHP Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<html>
```

```
<body>

<?php
class Car
{
    function Car() { $this-
        >model = "VW";
    }
}
// create an object
$herbie = new Car();
// show object
properties
echo
$herbie->model; ?>

</body>

</html>
```

OUTPUT:

VW

PHP NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

Example

```
<html>

<body>

<?php

$x = "Hello world!";
```

```
$x = null;  
var_dump($x  
);
```

```
?>
```

```
</body>
```

```
</html>
```

OUTPUT:

Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- **PHP Constants**
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no \$ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

- To create a constant, use the `define()` function.

Syntax

`define(name, value, case-insensitive)`

Parameters:

- ***name***: Specifies the name of the constant
- ***value***: Specifies the value of the constant
- ***case-insensitive***: Specifies whether the constant name should be case-insensitive. Default is false

The example below creates a constant with a **case-sensitive** name:

Example

```
<html>
<body>
<?php
// case-sensitive constant name
define("GREETING", "Welcome
to W3Schools.com!"); echo
GREETING;
?>
</body>
</html>
```

OUTPUT:

Welcome to W3Schools.com!

The example below creates a constant with a **case-insensitive** name:

Example

```
<html>
<body>
<?php
// case-insensitive constant name
define("GREETING", "Welcome to W3Schools.com!",
true);echo greeting;
?>
</body>
```

```
</html>
```

OUTPUT:

Welcome to W3Schools.com!

Constants are Global

- Constants are automatically global and can be used across the entire script.
- The example below uses a constant inside a function, even if it is defined outside the function:

Example

```
<html>
```

```
<body>
```

```
<?php
```

```
define("GREETING", "Welcome to  
W3Schools.com!"); function
```

```
myTest() {
```

```
    echo GREETING;
```

```
}
```

```
myTest();
```

```
?>
```

```
</body>
```

```
</html>
```

OUTPUT:

Welcome to W3Schools.com!

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
-

PHP Arithmetic Operators

- The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$\$x + \y	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \y	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \y	Product of $\$x$ and $\$y$
/	Division	$\$x / \y	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \y	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \y	Result of raising $\$x$ to the $\$y$ 'th power

(Introduced in PHP 5.6)

Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word **function**:

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

- Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example

```
<html>  
  
<body>  
  
<?php  
  
function  
writeMsg()echo  
"Hello world!";  
}  
writeMsg();  
?>
```


</body>

</html>

OUTPUT:

Hello world!

PHP Form Handling

The PHP super globals `$_GET` and `$_POST` are used to collect form-data.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. It is an associative array of variables passed to the current script via the URL parameters (aka. query string). Note that the array is not only populated for GET requests, but rather for all requests with a query string. You can send limited amount of data through get request. The GET variables are passed through `urldecode()`. In general, a URL with GET data will look like this:

`http://www.example.com/action.php?name=xyz&age=32`

`$_GET` examples:

```
<?php
```

```
echo 'Hello ' . htmlspecialchars($_GET["name"]) . '!';
```

```
?>
```

Assuming the user entered

`http://example.com/?name=Hannes` The above example will

output something similar to:

Hello Hannes!

File: form1.html

1. `<form action="welcome.php" method="get">`
2. Name: `<input type="text" name="name"/>`
3. `<input type="submit" value="visit"/>`
4. `</form>`

File: welcome.php

1. `<?php`
2. `$name=$_GET["name"];` //receiving name field value in \$name variable
3. `echo "Welcome,`

`$name";4. ?>`

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 form data is sent with the HTTP GET method.

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc. The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request. It is an associative array of variables passed to the current script via the HTTP POST method when using application/x-www-form-urlencoded or multipart/form-data as the HTTP Content-Type in the request.

`$_POST` examples:

```
<?php
echo 'Hello ' . htmlspecialchars($_POST["name"]) . '!';
?>
```

Assuming the user POSTed name=Hannes

The above example will output something similar to:

Hello Hannes!

File: form1.html

1. `<form action="login.php" method="post">`
2. `<table>`
3. `<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>`
4. `<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>`
5. `<tr><td colspan="2"><input type="submit" value="login"/> </td></tr>`
6. `</table>`
7. `</form>`

File: login.php

1. `<?php`
2. `$name=$_POST["name"];//receiving name field value in $name variable`
3. `$password=$_POST["password"];//receiving password field value in $password variable`
4. `echo "Welcome: $name, your password is:`

`$password";5. ?>`

GET vs. POST

Both GET and POST create an array (e.g. `array(key1 => value1, key2 => value2, key3 =>value3, ...)`). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

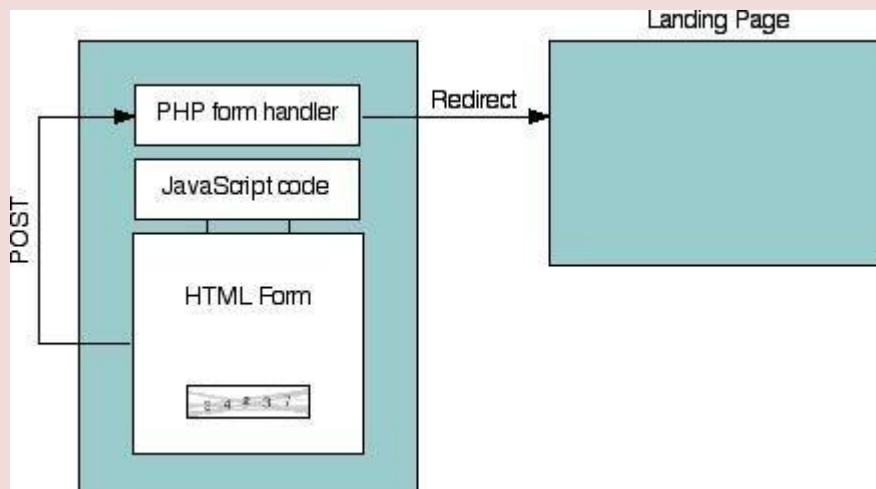
Both GET and POST are treated as `$_GET` and `$_POST`. These are super globals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

`$_GET` is an array of variables passed to the current script via the URL parameters.

`$_POST` is an array of variables passed to the current script via the HTTP POST method.

PHP FORM HANDLER

PHP is used to insert the form `action` as the current page. That's because we are using the "redirect-after-POST" technique as illustrated here:



It prevents the form from being resubmitted if the landing page is reloaded, and allows us to display validation error messages inline using PHP. Finally, the code includes PHP commands to re-insert any submitted values back in to the form so they don't have to be retyped in case of an error.

ARRAY

Array is complex variable that enables us to store multiple values in a single variable. We use this to store related information. This value can have same name and similar action performing and them.

An array is defined as following.

```
<?php
// define array
$ fruits={' apple',' mango',' orange'}
$ fruits[0]=' grapes'
?>
```

Here \$ fruits is an array variable. It contains 3 values" grapes, orange, mango". The elements of the array are accessed using an index. The index number of first element is zero, the index number of second element is one and soon.

So \$ fruits[0]=' grapes'

Suppose if we want to add new element to this array. We can write statement
\$fruits[3]=' banana';

Array using the notation key value pair

In PHP we have a different type of Array which relates key and value. This type of array is also known as hash or associative array.

Example of this array is given below. Position as key value.

```
<?
$fruits={ yellow=' mango'
        Purple=' grapes'
        Orange=' orange'
}
?>
```

Here \$fruits is an array variable of type key values. We access element by using

the key instead of position number. \$fruits[yellow]='mango'. We are using the key yellow instead of position 0. Creating an array:

Method 1:

```
$ Fruits=array(' strawberry'; grape' vanilla'; caramel')
```

Method 2:

```
$ fruits[0]=' strawberry';
```

```
$ fruits[1]=' grapes';
```

Method 3:

```
$ fruits[' red']=' apple';
```

```
$ fruits[' yellow']=' mango';
```

Modifying an array element:

To add an element to the array we can write the statement \$fruit[5] ='lemon'. Suppose if we don't know the last position to add a new element in array we can simply write

```
$ fruit[]=' strawberry';
```

Deleting an array element:

To remove an element we can use array_ push or array_ pop(). These two functions are built-in array functions in PHP.

Processing array with loop:

In PHP we can process arrays using loops like for, while, etc..

We also have a special loop in PHP known as for each loop for array processing. Example for array processing with loop

```
<html>
```

```
<head></head>
```

```
<body>
```

Today shopping list;

```
<ul>
<?php
    $shoppinglist=array('green gram','bengal gram','rice');
    for($x=0;$x< sizeof($shoppinglist);x++)
    {
        echo<li>$shoppinglist[$x];
    }
?>
</ul>
</body>
</html>
```

In this example for loop is used to interact with the array. This loops extract elements from the array and prints in the screen one after another as an order list. The sizeof() function returns the number of elements in the array.

foreach loop()

This loop runs for each element of the array moving through the element of array on each iteration. In **for loop**, we have condition statement and iteration(increment/decrement) statement. Condition statement and iteration statement are not needed in foreach() loop.

The syntax of foreach() loop is.

```
foreach (array variable as loop variable)
{
// loop statement
}
```

example using foreach() loop

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

Today's shopping list;

```
<ul>
```

```
<?php
```

```
$shoppinglist=array ('green gram', 'bengal gram', 'rice');
```

```
    foreach($shoppinglist as $ item)
```

```
{
```

```
}
```

```
?>
```

```
</ul>
```

Using array functions:

There are many built-in array functions in PHP that we can use along with array.

1. is_array()

This function check whether the variable in PHP is a array variable or not. It returns Booleanvalue as output

2. array_key()

This function returns the list of key in associative array for example this function will return

'dog', 'cat', 'parrot' from the array \$animals.

3. array_value()

This function will return only the array element in an associative array. For example, thisfunction returns 'Tripsy', 'Tabitha', 'polly' from the animals array,

4. list()

This list function assigns array elements to array variable. ex:

```
$flavours=array(' strawberry', 'grape',
```



```
'vanilla');list($f1, $f2, $f3) = $flavours;
```

\$f1 will have strawberry

\$f2 will have

grape and so on

5.extract()

The extract() function iterates through (associative array) converting the key value pairs into corresponding variable value pairs.

```
$fruits= array('red'= 'apple', 'yellow'= 'banana', 'purple'= 'grapes');Extract ($fruits):
```

\$red will have 'apple'

\$yellow will have 'banana'

\$purple will have 'grapes'

6.Array_push() function:

It adds an element from the end of

the array.Array_push(\$student,

'John');

The element John is added to the \$students array.

7. Array_pop()

This function removes an element from the end of

the array.ex:-

```
array-pop($students)
```

8.array-shift ()

This function is used to pop element at the beginning of

the array.array-shift (\$student)

9.array-unshift:

This function adds element at the beginning of the array.array-unshift (\$students, 'Ronald');

10.Explode ()

The explode function splits a string into smaller components on the basis of a user defined character and then returns those element in an array.

ex:-

```
$string=' this is a book';
```

```
$words = explode ('$string', ' ');
```

This function returns and array variable \$words will contain ('this', 'is', 'book')

11.implode ()

The implode function create a single string from all the element of an array joining them together with user defined separator.

EX:-

```
$words=array('This', ' is', 'a', 'book', 'of', 'Hindi');
```

```
$string=implode(' ', $words);
```

```
$string='This is a book of Hindi';
```

Function

A function is a set of program statements that perform a specific task. Functions can be called or executed from anywhere in the program. All the programming languages have built-in functions and also allow us to create user defined function. For example we can use the function with name pow from c library math.h or we can define our own function.

Usage of function:-

1. Reducing repetition:

User defined function enables developer to extract commonly used pieces of code as separate package. So it reduces unnecessary code repetition and redundancies also makes the code easier to understand and debug.

2. Easy maintenance:-

Because functions are defined once and used many times, they are used to maintain the code. During code maintenance if we want to change from values in calculation, we need to change only in the function. We need not Traverse the whole program for making change in one value.

3. Improves abstraction:

Function forces programmer to think in abstract terms. We need not worry about implementation of the function. It is enough that we know the function name, number of arguments and return type of the function.

Creating user defined function:

Consider the example below which define a function for displaying Shakespeare quote in a webpage.

```
<?php
//define a function
function displayShakespearQuote()
{
echo'some are born great, some achieve greatness and some have greatness through upon
them'
;.
}
```

invoking function

```
<?php
....
....
displayShakespearQuote();
?>
```

- PHP functions are defined with the function keyword following by name of the function.
- The name of function should follow the rule for naming variables.
- After function name list of arguments enclosed in parenthesis() should be present.
- It is optional and can be omitted if no argument is present.
- After the first line of the function the body of the function should be present inbetween the curly brackets {}.
- The function code can contain any valid PHP statements, which includes loops conditional statements and call to other function.
- Invoking a function is done by calling a function with its name.
- If the function had arguments we have to specify the arguments during invoking

Example of PHP script for function with argument.

```
<?php
//define a function
function triangle_area ($base,$height)
{
$area=$base*$height
*0.5return $area;
}
//invoke a function
$ta=triangle_area(10,50);
echo'the area of a triangle is
$ta';
?>
```

Using arrays as function arguments and return value

PHP fully supports passing arrays to function.Ex:-

```
<?php
//define a function with array as
```

```
argumentfunction
```

```
addDomainToUsername($u,$d)
```

```
{  
    //great empty result array  
    $result_array=array();  
    foreach($u as $element)  
    {  
        $result_array[]=$element,'@',$d;  
    }  
    return $result_array;  
}  
$users=array('John', 'jim', 'harry');  
$newusers=addDomainToUsername($users, 'guese.me.domain');  
?>
```

Defining global and local variable:-

The variables defined inside functions are local variables. They cannot be used from outside the function. If we want to use a variable throughout the script in all functions we have to declare that variable with Global key word. Global variable can be used in PHP script for counting the number of visitors in a website. This is done by the following statement.

Global \$count

Super Global variables:-

There are some variables provided by the PHP interpreter that we can use in our PHP script. These variables can be accessed from anywhere in our program. There are variables which are used commonly for creation of websites with many web pages. All these Global variables starts with \$_

They are

\$_SERVER

\$_POST

Used to collect data after submitting HTML form with method 10 =" post"

\$_GET

Collect form data after submitting HTML form with method = "GET"

`$_REQUEST`

Used to collect data after submitting an HTML form.

`$_GLOBALS`

It contains all the super Global variables in installed PHP version.

`$_FILES`

`$_SESSION`

`$_COOKIE`

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.

What is a PHP Session?

When you work 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 a problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

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

If you need a permanent storage, you may want to store the data in a database.

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 file
```

```
session
```

```
session_start();
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?p // Set session variables
```

```
hp $_SESSION["favcolor"] = "green";
```

```
$_SESSION["favanimal"] =
```

```
"cat"; echo "Session variables
```

```
are set.";
```

```
?>
```

```
</body>
```

```
</html>
```

Get PHP Session Variable Values

Next, 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").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start())

Also notice that all session variable values are stored in the global \$_SESSION variable:

Example

```
<?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>
```

Another way to show all the session variable values for a user session is to run the following code:

```
<?php
session_start
();
?>
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSI
ON);
?>
</body>
</html>
```


How does it work? How users are identified?

Most sessions set a user-key on the user's computer that looks something like this.

765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session

Variable

To change a session variable, just overwrite it by using the following statement

```
$_SESSION["favcolor"] =  
    "yellow";
```

Destroy a PHP Session

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

Cookie

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 a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

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

Only the name parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable \$_COOKIE)

We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Jegan Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>

<html>
<body>
<?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];
}
?>

</body>
</html>
```

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

Note: The value of the cookie is automatically URLencoded when sending the

cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Atul Kailash";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>

<html>
<body>

<?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];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the `setcookie()` function, then count the `$_COOKIE` array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
```

Also notice that all session variable values are stored in the global `$_SESSION` variable

```
<?php
if(count($_COOKIE) >
0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

PHP fscanf() Function

Definition and Usage

The `fscanf()` function parses the input from an open file according to the specified format.

Note: Any whitespace in the format string matches any whitespace in the input stream. This means that a tab (`\t`) in the format string can match a single space character in the input stream.

Syntax

```
fscanf(file, format, mixed)
```

Parameter Description

`file` Required. Specifies the file to check
`format` Required. Specifies the format.

Possible format values:

%% - Returns a percent sign

%b - Binary number

%c - The character according to the ASCII value

%d - Signed decimal number

%e - Scientific notation (e.g. 1.2e+2)

%u - Unsigned decimal number

%f - Floating-point number (local settings aware)

%F - Floating-point number (not local settings aware)

%o - Octal number

%s - String

%x - Hexadecimal number (lowercase letters)

%X - Hexadecimal number (uppercase letters)

Additional format values. These are placed between the % and the letter (example %.2f):

+ (Forces both + and - in front of numbers. By default, only negative numbers are marked)

' (Specifies what to use as padding. Default is space. Must be used together with the widthspecifier. Example: %'x20s (this uses "x" as padding)

- (Left-justifies the variable value)

[0-9] (Specifies the minimum width held of to the variable value)

.[0-9] (Specifies the number of decimal digits or maximum string length)

Note: If multiple additional format values are used, they must be in the same order as above.

parse_ini_file() function Definition and Usage

The parse_ini_file() function parses a configuration (ini) file and returns the settings.

Tip: This function can be used to read in one's own configuration files, and has nothing to do with the php.ini file.

Note: The following reserved words must not be used as keys for ini files: null, yes, no, true, false, on, off, none. Furthermore, the following reserved characters must not be used in the key: { } | & ~ ! () ^ " .

Syntax

`parse_ini_file(file, process_sections, scanner_mode)`

Parameter Description

`file` Required. Specifies the ini file to parse

`process_sections` Optional. If set to TRUE, it returns is a multidimensional array with section names and settings included. Default is FALSE

`scanner_mode`

Optional. Can be one of the following values:

INI_SCANNER_NORMAL (default)

INI_SCANNER_RAW (means option values will not be parsed)

INI_SCANNER_TYPED (means that boolean, null and integer types are preserved when possible. "true", "on", "yes" are converted to TRUE. "false", "off", "no", "none" are converted to FALSE. "null" is converted to NULL. Numeric strings are converted to integer type if possible)

Getting file information with stat

Definition and Usage

The stat() function returns information about a file as an array.

Note: The results from this function will differ from server to server. The array may contain the number index, the name index, or both.

Note: The result of this function is cached. Use clearstatcache() to clear the cache.

Syntax

```
stat(filename)
```

Parameter Description

filename Required. Specifies the path to the file

Return Value:

An array with the following elements:

- [0] or [dev] - Device number
- [1] or [ino] - Inode number
- [2] or [mode] - Inode protection mode
- [3] or [nlink] - Number of links
- [4] or [uid] - User ID of owner
- [5] or [gid] - Group ID of owner
- [6] or [rdev] - Inode device type
- [7] or [size] - Size in bytes
- [8] or [atime] - Last access (as Unix timestamp)
- [9] or [mtime] - Last modified (as Unix timestamp)
- [10] or [ctime] - Last inode change (as Unix timestamp)
- [11] or [blksize] - Blocksize of filesystem IO (if supported)
- [12] or [blocks] - Number of blocks

failure

Example

Get information about a file using stat():

```
<?php
$stat = stat("test.txt");
echo "Access time: " . $stat["atime"];
echo "<br>Modification time: "
.$stat["mtime"];echo "<br>Device number: "
.$stat["dev"];
?>
```

Output

```
Access time: 1566689194
Modification time: 1550577107
Device number: 2049
```

Fseek

Definition and Usage

The `fseek()` function seeks in an open file.

This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.

Tip: You can find the current position by using `ftell()`!

Syntax

```
fseek(file, offset, whence)
```

Parameter Description

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

whence Optional. Possible values:

SEEK_SET - Set position equal to offset. Default

SEEK_CUR - Set position to current location plus
offset

SEEK_END - Set position to EOF plus offset (to move to a position before EOF, the
offset must be a negative value)

Return Value: 0 on success, otherwise -1

Copying files with copy

Definition and Usage

The copy() function copies a file.

Note: If the to_file file already exists, it will be overwritten.

Syntax

```
copy(from_file, to_file, context)
```

Parameter Description

from_file Required. Specifies the path to the file to copy

fromto_file Required. Specifies the path to the file to

copy to

context Optional. Specifies a context resource created with stream_context_create()

Return Value: TRUE on success, FALSE on failure

Example

```
<?php  
echo copy("source.txt","target.txt");  
?>
```

Deleting files

Deletion of files is done using unlink()

function. Definition and Usage

The unlink() function deletes a file.

Syntax

```
unlink(filename, context)
```

Parameter Description

filename Required. Specifies the path to the file to delete

context Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

Return Value: TRUE on success, FALSE on failure

Example

Delete a file using unlink()

```
<?php
```

```
$file = "test.txt";
```

```
if (!unlink($file)) {
```

```
    echo ("Error deleting $file");
```

```
} else {
```

```
    echo ("Deleted $file");
```

```
}
```

```
?>
```

Reading and writing binary files

A better method to open files is with the fopen () function.

The file may be opened in one of the following modes

Modes

Description

R	Open a file for read only. File pointer starts at the beginning of the file
W	Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
A	Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
X	Creates a new file for write only. Returns FALSE and an error if file already exists

r+	Open a file for read/write. File pointer starts at the beginning of the file
w+	Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	Creates a new file for read/write. Returns FALSE and an error if file already exists
B	Opens the file in binary mode for reading/writing.

Working with Files

- File handling is an important part of any web application. You often need to

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

PHP `readfile()` Function

- The `readfile()` function reads a file and writes it to the output buffer.
- Assume we have a text file called "**webdictionary.txt**", stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and

XMLCSS = Cascading Style Sheets

HTML = Hyper Text Markup

LanguagePHP = PHP Hypertext

Preprocessor SQL = Structured

Query Language SVG = Scalable

Vector Graphics

XML = EXtensible Markup Language

- The PHP code to read the file and write it to the output buffer is as follows (the `readfile()` function returns the number of bytes read on success):

Example

```
<html>  
<body>  
<?php  
echo readfile("webdictionary.txt");
```

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language236

File Open/Read/Close

PHP Open File -

fopen()

- A better method to open files is with the `fopen()` function. This function gives you more options than the `readfile()` function.

We will use the text file, "**webdictionary.txt**", during the lessons: AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup

Language PHP = PHP Hypertext

Preprocessor SQL = Structured

Query Language SVG = Scalable

Vector Graphics

XML = EXtensible Markup Language

The first parameter of `fopen()` contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the `fopen()` function is unable to open the specified file:

Example

```
<?php
```

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
```

```
file!"); echo fread($myfile, filesize("webdictionary.txt"));
```

```
fclose($myfile);
```

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

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

Modes	Description
r	Open a file for read only. File pointer starts at the beginning of the file
w	Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	Creates a new file for write only. Returns FALSE and an error if file already exists

r+	Open a file for read/write. File pointer starts at the beginning of the file
w+	Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	Creates a new file for read/write. Returns FALSE and an error if file already Exists

PHP Read File - fread()

- The `fread()` function reads from an open file.
- The first parameter of `fread()` contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose()

- The `fclose()` function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
- The `fclose()` requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
```

```
$myfile =
```

```
fopen("webdictionary.txt","r"); //
```

```
some code to be executed....
```

```
fclose($myfile);
```

```
?>
```

PHP Read Single Line - fgets()

- The `fgets()` function is used to read a single line from a file.
- The example below outputs the first line of the "webdictionary.txt" file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");echo fgets($myfile);

fclose($myfile);

?>
```

OUTPUT:

AJAX = Asynchronous JavaScript and XML

After a call to the `fgets()` function, the file pointer has moved to the next line.

PHP Check End-Of-File - feof()

- 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.
- The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-
filewhile(!feof($myfile)) {
    echo fgets($myfile) . "<br>";
```

```
}  
fclose($myfile);  
?>
```

OUTPUT:

AJAX = Asynchronous JavaScript and

XMLCSS = Cascading Style Sheets

HTML = Hyper Text Markup

LanguagePHP = PHP Hypertext

Preprocessor SQL = Structured

Query Language SVG = Scalable

Vector Graphics

XML = EXtensible Markup Language

PHP Read Single Character - fgetc()

- The `fgetc()` function is used to read a single character from a file.
- The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

```
<?php
```

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
```

```
// Output one character until end-of-file
```

```
while(!feof($myfil  
e)) { echo  
fgetc($myfile);  
  
}  
fclose($myfile);  
?>
```

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets

HTML = HyperText Markup Language PHP = PHP Hypertext Preprocessor SQL

= Structured Query Language SVG = Scalable Vector Graphics XML =

EXtensible Markup Language

- After a call to the `fgetc()` function, the file pointer moves to the next character.

File Create/Write

PHP Create File - `fopen()`

- The `fopen()` function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use `fopen()` on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
- The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite()

- The `fwrite()` function is used to write to a file.
- The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.
- The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
```

```
$myfile = fopen("newfile.txt", "w") or die("Unable  
to openfile!"); $txt = "John Doe\n";  
fwrite($myfile, $txt);
```

```
$txt = "Jane  
Doe\n";  
fwrite($myfile,  
$txt);  
fclose($myfile);
```

```
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string \$txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the `fclose()` function.

If we open the "newfile.txt" file it would look like this:

John

DoeJane

Doe

PHP Overwriting

- Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
- In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
```

```
$myfile = fopen("newfile.txt", "w") or die("Unable
```

```
to openfile!"); $txt = "Mickey Mouse\n";
```

```
fwrite($myfile, $txt);
```

```
$txt = "Minnie
```

```
Mouse\n";
```

```
fwrite($myfile, $txt);
```

```
fclose($myfile);
```

?>

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

OUTPUT:

Mickey

Mouse

Minnie

Mouse

PHP ftp_fput()

Function Example

- Open local file, and upload it to a file on the FTP server:

```
<?php
// connect and login to FTP server

$ftp_server = "ftp.example.com";

$ftp_conn = ftp_connect($ftp_server) or die("Could not connect to
$ftp_server"); $login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);

// open file for reading
$file = "test.txt";
$fp = fopen($file,"r");
// upload file
if (ftp_fput($ftp_conn, "somefile.txt", $fp, FTP_ASCII))
{
    echo "Successfully uploaded $file.";
}
else
{
    echo "Error uploading $file.";
}
// close this connection and file
handlerftp_close($ftp_conn);
fclose($fp);
```


?>

Working with Databases

PHP MySQL Database

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- 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

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers

- Orders

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

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

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like". Both MySQLi and PDO have their advantages:

- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

- Both are object-oriented, but MySQLi also offers a procedural API.
- Both support Prepared Statements. Prepared Statements protect from SQLinjection, and are very important for web application security.

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

MySQL Installation

- For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.
- For installation details, go to: <http://php.net/manual/en/mysqli.installation.php>
- PDO Installation
- For installation details, go to: <http://php.net/manual/en/pdo.installation.php>

Open a Connection to MySQL

- Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

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

```
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo
"Connected
successfully";
?>
```

Example (MySQLi Procedural)

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

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check
connectionif
(!$conn) {

    die("Connection failed: " . mysqli_connect_error());
}
```

```
echo
```

```
"Connected  
successfully";
```

```
?>
```

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
try {
```

```
    $conn = new PDO("mysql:host=$servername;dbname=myDB",  
$username, $password);
```

```
    // set the PDO error mode to exception $conn-
```

```
    >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); echo
```

```
    "Connected successfully";
```

```
}
```

```
catch(PDOException $e)
```

```
{
```

```
    echo "Connection failed: " . $e->getMessage();
```

```
}
```

```
?>
```

Notice that in the PDO example above we have also specified a database (myDB). PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

Example (MySQLi Object-Oriented)

```
$conn->close();
```

Example (MySQLi Procedural)

```
mysqli_close($conn);
```

Example (PDO)

```
$conn = null;
```

PHP Create a MySQL Database

- A database consists of one or more tables.
- You will need special CREATE privileges to create or to delete a MySQL database.
- **Create a MySQL Database Using MySQLi**
- The CREATE DATABASE statement is used to create a database in MySQL.
- The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```

$password = "password";

// Create connection

$conn = new mysqli($servername, $username,
$password); // Check
connectionif ($conn-
>connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database

$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database
createdsuccessfully"; }
else {
    echo "Error creating database: " . $conn->error;
}

$conn-
>close(); ?>

```

PHP Create MySQL Table

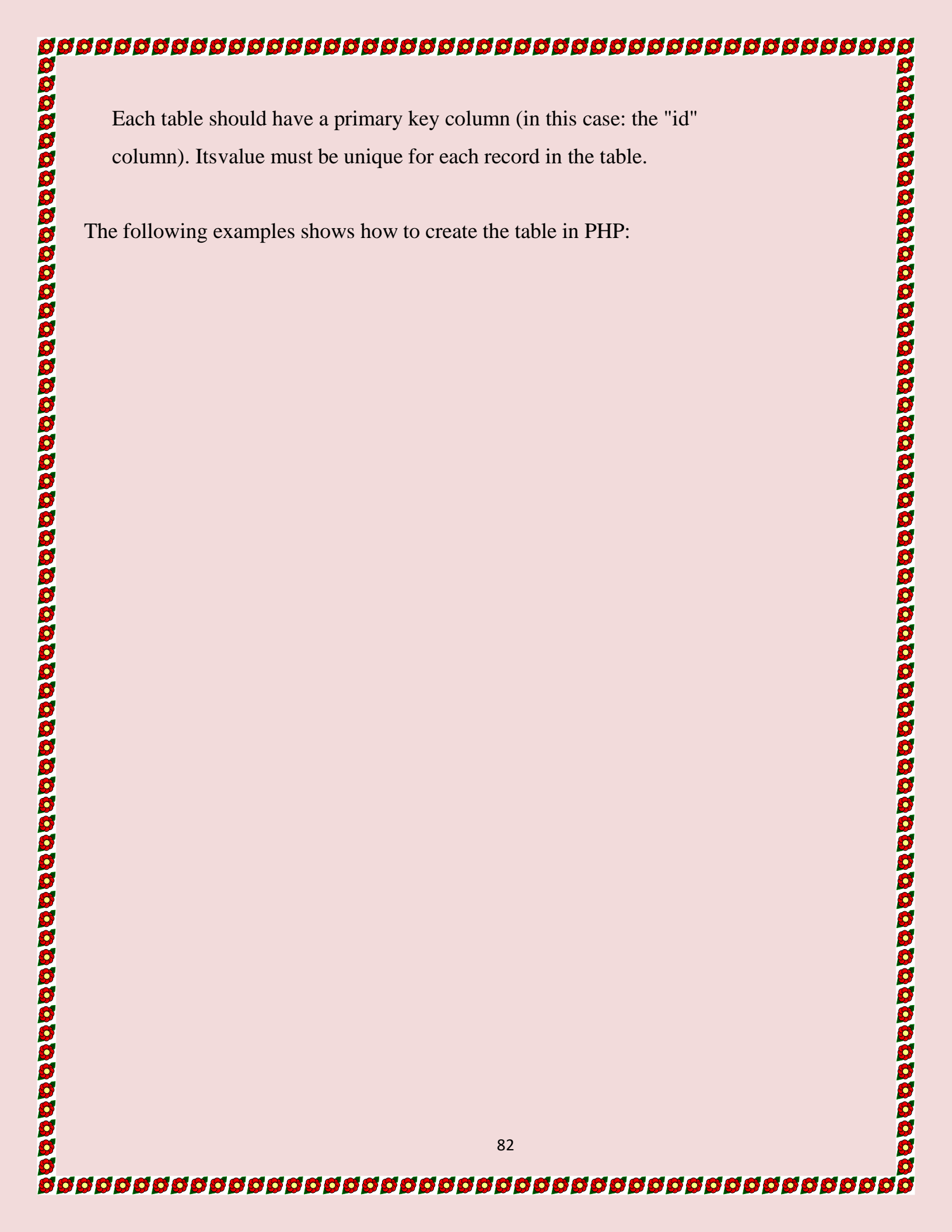
- A database table has its own unique name and consists of columns and rows.
- Create a MySQL Table Using MySQLi
- The CREATE TABLE statement is used to create a table in MySQL.

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

```
CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO_INCREMENT  
PRIMARYKEY, firstname VARCHAR(30) NOT  
NULL,  
lastname VARCHAR(30)  
NOTNULL, email  
VARCHAR(50),  
reg_date TIMESTAMP  
)
```

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT



Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example (MySQLi Object-oriented)

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

// Create connection

$conn = new mysqli($servername, $username, $password,
$dbname); // Check
connectionif ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

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

```
if ($conn->query($sql) === TRUE)

{echo "Table MyGuests created
successfully"; } else {
    echo "Error creating table: " . $conn->error;
}
$conn-
>close(); ?>
```

PHP Insert Data Into MySQL

Insert Data Into MySQL Using MySQLi

- After a database and a table have been created, we can start adding data in them.

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

The INSERT INTO statement is used to add new records to a MySQL table:

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

The following examples add a new record to the "MyGuests" table:

```
<?php
```

```
$servername = "localhost";
```

```
$username = "username";
```

```
$password = "password";
```

```
$dbname = "myDB";
```

```
// Create connection
```

```
$conn = new mysqli($servername, $username, $password,  
$dbname); // Check
```

```
connectionif ($conn-
```

```
>connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
```

```
}
```

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

```
if ($conn->query($sql) ===
```

```
    TRUE) {echo "New record
```

```
    created
```

```
    successfully"; } else {
```

```
    echo "Error: " . $sql . "<br>" . $conn->error;
```

```
}
```

```
$conn-
```

```
>close(); ?>
```

Example:

Student Application using PHP and

Mysqlaa.html:

```
<!DOCTYPE HTML>

<html>

<head>

<title>Student
Table</title>
</head>

<body>

<div id="dept"> <h3 align=center>Student table
entry</h3> <form method="POST"
action="connect.php">
SName <br><input type="text"
name="sname"><br> Reg.No <br><input
type="text" name="regno"><br>
Mark1<br><input type="text"
name="m1"><br> Mark2<br><input
type="text" name="m2"><br> <input
type="submit" value="ok">

</form>

</div>

</body>

</html>
```

Conn.php:

```
<?php

$servername = "localhost";

$username = "root";
$password = "";

$dbname = "sample";

// Create connection

$conn = new mysqli($servername, $username, $password,$dbname);
// Check connection

if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}

echo "Connected
successfully"; //connect
table

$sql="desc student"; if($conn-
>query($sql)==TRUE)
{
echo "<br>";

echo "connected to the table";
```

```
}  
  
else  
  
{  
  
echo "error";  
  
}  
  
//Inserting the  
contents echo  
"<br>";  
  
//insertion from html  
  
$sname=$_POST['sname'];  
  
$regno=$_POST['regno'];  
  
$m1=$_POST['m1'];  
  
$m2=$_POST['m2'];  
  
$sql11="insert into student  
values('$sname',$regno,$m1,$m2)";if($conn-  
>query($sql11)==TRUE)  
  
{  
  
echo "inserted";  
  
}
```



```

else

{echo
"error";}

echo "<br>";

$sql1="select * from student";

$result = $conn-
>query($sql1);if ($result-
>num_rows > 0) {

    // output data of each row

echo "<b>Sname Regno M1
M2</b><br>";while($row = $result-
>fetch_assoc()) {

    echo $row["sname"]." ". $row["regno"]." ".$row["m1"]."
".$row["m2"]."<br>";

    }

} else {

    echo "empty table";

}
$conn-
>close(); ?>

```

php - vimaljantech@gp x Index of /sample
localhost/sample

Index of /sample

Name	Last modified	Size	Description
Parent Directory	-	-	-
connect.php	04-Oct-2017 21:52	1.2K	
stud.html	04-Oct-2017 21:58	448	

Apache/2.2.9 (Win32) DA17/2 mod_ssl/2.2.9 OpenSSL/0.9.8h mod_authn_core/PHP/5.2.6 Server at localhost Port 80

Taskbar with icons for Internet Explorer, Firefox, Chrome, and others. System tray shows 10:48 PM, 10/4/2017.

php - vimaljsmtech@g... X localhost/sample/conne... X
localhost/sample/connect.php

Connected successfully
connected to the table
inserted
Surname Regno M1 M2
aaa 1234 67 90
xxx 123 56 66
zzz 1234567 78 99

10:34 PM
10/A/2017