* Overview Of HTML
* **introduction**
* The Internet and web programming technologies Allow you to learn in this course are designed to be *portable*, allowing you to design web pages and applications that run across an enormous range of Internet-enabled devices.
* *Client-side programming* technologies are used to build web pages and applications that are run on the *client* (i.e., in the browser on the user’s device).
* *Server-side programming*—the applications that respond to requests from client-side web browsers, such as searching the Internet, checking your bank-account balance, ordering a book from Amazon, bidding on an eBay auction and ordering concert tickets (web design 2)
* **HTML ,CSS, JavaScript?**

HTML is a markup language for describing web documents (web pages).

HTML stands for Hyper Text Markup Language

A markup language is a set of markup tags

HTML documents are described by HTML tags

Each HTML tag describes different document content.

Cascading Style Sheets (CSS) are used to specify the *presentation*, or styling, of elements on a web page (e.g., fonts, spacing, sizes, colors, positioning).

JavaScript helps you build *dynamic* web pages (i.e., pages that can be modified “on the fly” in response to *events*, such as user input, time changes and more) and computer applications.

It enables you to do the client-side programming of web applications.

**A web Browser**

A [web browser](http://en.wikipedia.org/wiki/Web_browser) can read HTML files and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses them to interpret the content of the page. HTML describes the structure of a website

**Advantages of HTML:**

1. First advantage it is widely used.  
2. Every browser supports HTML language.  
3. Easy to learn and use.  
4. It is by default in every windows so you don't need to purchase extra software.

**Disadvantages of HTML:**

1. It can create only static and plain pages so if we need dynamic pages then HTML  
is not useful.  
2. Need to write lot of code for making simple webpage.  
3. Security features are not good in HTML.  
4. If we need to write long code for making a webpage then it produces some complexity

**PHP**

PHP is the most widely used scripting language for web programming. PHP extends HTML pages by adding server-executed code segments to HTML pages. The output of the execution of the PHP code is merged into the HTML page.

<?php

echo "Hello World";

?>

**MySQL**

MySQL is one of the most popular free and open source database engines in the market place. MySQL powers Facebook, Yahoo!, WordPress, Drupal, Joomla, and millions of other dynamic web sites

**INSERT INTO users VALUES('Smith', 'John', 'jsmith@mysite.com');**

**SELECT surname,firstname FROM users WHERE email='jsmith@mysite.com';**

The History of PHP

* Rasmus Lerdorf
* Not a trained computer scientist
* Consultant building dynamic web sites - got tired of doing the same thing over and over in C
* Reusable bits + HTML Templates

**Apache Web Server**

Originally Developed at National Center for Supercomputing Applications in 1994

Open Source

First project / product of the Apache Foundation

Brian Behlendorf founded Apache

**Small HTML document**

<!DOCTYPE html>  
<html>  
 <body>  
  
 <h1>My First Heading</h1>  
  
 <p>My first paragraph.</p>  
  
 </body>  
</html>

**Example Explained**

The **DOCTYPE** declaration defines the document type

The text between **<html>** and **</html>** describes the web document

The text between **<body>** and **</body>** describes the visible page content

The text between **<h1>** and **</h1>** describes a heading

The text between **<p>** and **</p>** describes paragraph

Using the description, a web browser can display a document with a heading and a paragraph

* HTML tags normally come **in pairs** like <p> and </p>
* The first tag in a pair is the **start tag,** the second tag is the **end tag**
* The end tag is written like the start tag, but with a **slash** before the tag name .
* Note: The start tag is often called the **opening tag**. The end tag is often called the **closing tag**.

**HTML Documents**

* All HTML documents must start with a type declaration: **<!DOCTYPE html>**.
* The HTML document itself begins with **<html>** and ends with **< /html>**.
* The visible part of the HTML document is between **<body>** and **</body**
* HTML headings are defined with the **<h1>** to **<h6>** tags:
* **Example**

**HTML headings**

<h1>This is a heading</h1>  
<h2>This is a heading</h2>  
<h3>This is a heading</h3>

**HTML Links**

HTML links are defined with the **<a>** tag:

**Example**

<a href="http://www.google.com">This is a link</a>

**HTML IMGES**

<!DOCTYPE html>

<!-Including images in HTML5 files. -->-

<html>

<head>

<meta charset = "utf-8">

<title>Images</title>

</head>

<body>

<p>

<img src = "download.jpg" width = "92" height = "120"

alt = "Astronomy Picture">

<img src = "download2.jpg" width = "92" height = "120"

alt = "Astronomy Picture">

</p>

</body>

</html

**HTML LISTS**

**<!DOCTYPE html>**

**<!-- Unordered list containing hyperlinks. -->**

**<html>**

**<head>**

**<meta charset = "utf-8">**

**<title>Links</title>**

**</head>**

**<body>**

**<h1>Here are my favorite sites</h1>**

**<p><strong>Click on a name to go to that page</strong></p>**

**<!-- create an unordered list -->**

**<ul>**

**<!-- the list contains four list items -->**

**<li><a href = "http://www.youtube.com">YouTube</a></li>**

**<li><a href = "http://www.wikipedia.org">Wikipedia</a></li>**

**<li><a href = "http://www.amazon.com">Amazon</a></li>**

**<li><a href = "http://www.linkedin.com">LinkedIn</a></li>**

**</ul>**

**</body>**

**</html>**

**HTML Tables**

<!DOCTYPE html>

<html>

 <head>

<style>

table, th, td { border: 1px solid black; }

</style>

</head>

 <body>

 <table style="width:100%">

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

<tr>

<td>John</td>

<td>Doe</td>

<td>80</td>

</tr>

</table> </body> </html>

**Ch. 2 Introduction to PHP**

* **PHP is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages quickly**
* **PHP is a widely-used, and free.**
* **PHP runs over different operating systems such as Windows, Linux, Mac Os and Unix.**
* **PHP scripts are executed on the server, and the plain HTML result is sent back to the browser.**

**About the PHP Language**

**Syntax is inspired by C**

**Curly braces, semicolons, no signficant whitespace**

**Syntax inspired by perl**

**Dollar signs to start variable names, associative arrays**

**Extends HTML to add segments of PHP within an HTML file.**

**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 restrict users to access some pages on your website.**
* **PHP can encrypt data.**

**Introduction**

* **Documents end with the extension *.php***
* **To trigger the PHP commands, you need <?php tag.**

**and they finish only when the closing part ?> is encountered**

***Example:***

**<?php**

**echo "Hello world";**

**?>**

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

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?

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.

PHP Syntax

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

A PHP script can be placed anywhere in the document.

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

<?php  
// PHP code goes here  
?>

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

**Note:** PHP statements end with a semicolon (;)

## PHP Case Sensitivity

In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.

In the example below, all three echo statements below are equal and legal:

**<?php**

**ECHO "Hello World!<br>";**

**echo "Hello World!<br>";**

**EcHo "Hello World!<br>";**

**?>**

**Note:** However; all variable names are case-sensitive!

Look at 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:

<?php

$color = "red";

echo "My car is " . $color . "<br>";

echo "My house is " . $COLOR . "<br>";

echo "My boat is " . $coLOR . "<br>";

?>

PHP Comments

## Comments in PHP

A comment in PHP code is a line that is not executed as a 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 your code
* 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

PHP supports several ways of commenting:

<?php

// This is a single-line comment

# This is also a single-line comment

/\* This is for multiple-line comments

multiple-line comments

multiple-line comments \*/

?>

PHP 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:

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

echo $txt;

echo "<br>";

echo $x;

echo "<br>";

echo $y;

?>

After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

**Note:** When you assign a text value to a variable, put quotes around the value.

PHP Variables

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

Rules for PHP variables:

* 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)

Remember that PHP variable names are case-sensitive!

## 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:

<?php

$txt = "Isra University";

echo "I love $txt!";

?>

<?php

$txt = " Isra University ";

echo "I love " . $txt . "!";

?>

<?php

$x = 5;

$y = 4;

echo $x + $y;

?>

Output = 9

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch

PHP Data Types

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 In PHP!".

A string can be any text inside quotes. You can use single or double quotes:

<?php

$x = "Hello PHP!";

$y = 'I like PHP!';

echo $x;

echo "<br>";

echo $y;

## 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: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

**var\_dump() function returns the data type and value**

In the following example $x is an integer.

<?php

$x = 100;

var\_dump($x);

?>

Out put : int(100)

## 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.

<?php

$x = 9.251;

var\_dump($x);

?>

## PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

Thursday, Friday, and Saturday.

## PHP Array

An array stores multiple values in one single variable.

<?php

$week = array("Sunday","Monday","Tesday" , "Wednesday" , "Thursday", "Friday" , "Saturday");

var\_dump($week);

?>

## 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.

**Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.

Va <?php

$x = "Hello php!";

$x = null;

var\_dump($x);

?>

riables can also be emptied by setting the value to NULL:

# **PHP echo and print Statements**

With PHP, there are two basic ways to get output: echo and print.

In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

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

## The PHP 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):

<?php

echo "<h2>PHP is Fun!</h2>";

echo "Hello Isra unv.!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

?>

<?php

$txt1 = "Learn PHP";

$txt2 = "Isra un.";

$x = 5;

$y = 4;

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

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

echo $x + $y;

?>

## 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):

<?php

print "<h2>PHP is Fun!</h2>";

print "Hello Isra unv.!<br>";

print "I'm about to learn PHP!";

?>

**isplay Variables**

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

<?php

$txt1 = "Learn PHP";

$txt2 = "Isra un.";

$x = 5;

$y = 4;

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

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

print $x + $y;

?>

A screenshot of a cell phone

Description automatically generated

PHP String Functions

strlen() - Return the Length of a String

<?php

echo strlen("Hello php");

?>

OUTPUT 9

## str\_word\_count() - Count Words in a String

The PHP str\_word\_count() function counts the number of words in a string.

<?php

echo str\_word\_count("Hello php!");

?>

## strrev() - Reverse a String

The PHP strrev() function reverses a string.

<?php

echo strrev("Isra University !");

?>

# **PHP Numbers**

PHP Numbers

One thing to notice about PHP is that it provides automatic data type conversion.

So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.

This automatic conversion can sometimes break your code.

PHP Integers

An integer is a number without any decimal part.

2, 256, -256, 10358, -179567 are all integers. While 7.56, 10.0, 150.67 are floats.

So, an integer data type is a non-decimal number between -2147483648 and 2147483647. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.

Another important thing to know is that even if 4 \* 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

Here are some 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)

PHP has the following functions to check if the type of a variable is integer:

* is\_int()
* is\_integer() - alias of is\_int()
* is\_long() - alias of is\_int()

<?php

// Check if the type of a variable is integer

$x = 5985;

var\_dump(is\_int($x)); // The var\_dump() function dumps information about one or more variables. The information holds type and value of the variable(s).

echo "<br>";

// Check again...

$x = 59.85;

var\_dump(is\_int($x));

?>

PHP Floats

A float is a number with a decimal point or a number in exponential form.

2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.

The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following functions to check if the type of a variable is float:

* is\_float()
* is\_double() - alias of is\_float()

<?php

// Check if the type of a variable is float

$x = 10.365;

var\_dump(is\_float($x));

?>

## 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

### **Example**

Create a constant with a **case-sensitive** name:

<?php

// case-sensitive constant name

define("GREETING", "Welcome to Isra un.!");

echo GREETING;

?>

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
* Conditional assignment 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 |

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

|  |  |  |
| --- | --- | --- |
| **Assignment** | **Same as...** | **Description** |
| x = y | x = y | The left operand gets set to the value of the expression on the right |
| x += y | x = x + y | Addition |
| x -= y | x = x - y | Subtraction |
| x \*= y | x = x \* y | Multiplication |
| x /= y | x = x / y | Division |
| x %= y | x = x % y | Modulus |

PHP Arithmetic Operators

<?php

$x = 10;

$y = 4;

echo($x + $y);

echo "<br>";

echo($x - $y);

echo "<br>";

echo($x \* $y);

echo "<br>";

echo($x / $y);

echo "<br>";

echo($x % $y);

?>

## PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| == | Equal | $x == $y | Returns true if $x is equal to $y |
| === | Identical | $x === $y | Returns true if $x is equal to $y, and they are of the same type |
| != | Not equal | $x != $y | Returns true if $x is not equal to $y |
| <> | Not equal | $x <> $y | Returns true if $x is not equal to $y |
| !== | Not identical | $x !== $y | Returns true if $x is not equal to $y, or they are not of the same type |
| > | Greater than | $x > $y | Returns true if $x is greater than $y |
| < | Less than | $x < $y | Returns true if $x is less than $y |
| >= | Greater than or equal to | $x >= $y | Returns true if $x is greater than or equal to $y |
| <= | Less than or equal to | $x <= $y | Returns true if $x is less than or equal to $y |
| <=> | Spaceship | $x <=> $y | Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7. |

## PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Description** |
| ++$x | Pre-increment | Increments $x by one, then returns $x |
| $x++ | Post-increment | Returns $x, then increments $x by one |
| --$x | Pre-decrement | Decrements $x by one, then returns $x |
| $x-- | Post-decrement | Returns $x, then decrements $x by one |  |

## PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| and | And | $x and $y | True if both $x and $y are true |
| or | Or | $x or $y | True if either $x or $y is true |
| xor | Xor | $x xor $y | True if either $x or $y is true, but not both |
| && | And | $x && $y | True if both $x and $y are true |
| || | Or | $x || $y | True if either $x or $y is true |
| ! | Not | !$x | True if $x is not true |

## PHP String Operators

PHP has two operators that are specially designed for strings.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| . | Concatenation | $txt1 . $txt2 | Concatenation of $txt1 and $txt2 |
| .= | Concatenation assignment | $txt1 .= $txt2 | Appends $txt2 to $txt1 |

## PHP Array Operators

The PHP array operators are used to compare arrays.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| + | Union | $x + $y | Union of $x and $y |
| == | Equality | $x == $y | Returns true if $x and $y have the same key/value pairs |
| === | Identity | $x === $y | Returns true if $x and $y have the same key/value pairs in the same order and of the same types |
| != | Inequality | $x != $y | Returns true if $x is not equal to $y |
| <> | Inequality | $x <> $y | Returns true if $x is not equal to $y |
| !== | Non-identity | $x !== $y | Returns true if $x is not identical to $y |

# **PHP if...else...elseif Statements**

Conditional statements are used to perform different actions based on different conditions.

## PHP Conditional Statements

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

In PHP we have the following conditional statements:

* if statement - executes some code if one condition is true
* if...else statement - executes some code if a condition is true and another code if that condition is false
* if...elseif...else statement - executes different codes for more than two conditions
* switch statement - selects one of many blocks of code to be executed

## PHP - The if Statement

The if statement executes some code if one condition is true.

### **Syntax**

if (*condition*) { *code to be executed if condition is true*;  
}

### **Example**

Output "Have a good day!" if the current time (HOUR) is less than 20:

<?php

$t = date("H");

if ($t < "20") {

echo "Have a good day!";

}

?>

## PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

### **Syntax**

if (*condition*) {  
    *code to be executed if condition is true;*  
} else {  
  *code to be executed if condition is false;*}

### **Example**

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

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

==

## PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

### **Syntax**

if (*condition*) {  
    *code to be executed if this condition is true;*} elseif (*condition*) {  
  *code to be executed if first condition is false and this condition is true;*} else {  
    *code to be executed if all conditions are false;*}

### **Example**

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

<?php  
$t = date("H");  
  
if ($t < "10") {  
    echo "Have a good morning!";  
} elseif ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>

## The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

### **Syntax**

switch (*n*) {  
    case *label1:*  
  *code to be executed if n=label1;*  
        break;  
    case *label2:*  
  *code to be executed if n=label2;*  
        break;  
    case *label3:*  
  *code to be executed if n=label3;*  
        break;  
    ...  
    default:  
  *code to be executed if n is different from all labels;*  
}

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

### **Example**

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

# **PHP Loops**

## PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

* while - loops through a block of code as long as the specified condition is true
* do...while - loops through a block of code once, and then repeats the loop as long as the specified 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 following chapters will explain and give examples of each loop type.

## The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

### **Syntax**

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

### **Examples**

The example below displays the numbers from 1 to 5:

### **Example**

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

### **Example Explained**

* $x = 1; - Initialize the loop counter ($x), and set the start value to 1
* $x <= 5 - Continue the loop as long as $x is less than or equal to 5
* $x++; - Increase the loop counter value by 1 for each iteration

## The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

### **Syntax**

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

### **Examples**

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

### **Example**

<?php   
$x = 1;   
  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>

**Note:** In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

This example sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

### **Example**

<?php   
$x = 6;  
  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>

## The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

### **Syntax**

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

Parameters:

* *init counter*: Initialize the loop counter value
* *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
* *increment counter*: Increases the loop counter value

### **Examples**

The example below displays the numbers from 0 to 10:

### **Example**

<?php   
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}   
?>

### **Example Explained**

* $x = 0; - Initialize the loop counter ($x), and set the start value to 0
* $x <= 10; - Continue the loop as long as $x is less than or equal to 10
* $x++ - Increase the loop counter value by 1 for each iteration

## The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

### **Syntax**

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

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

### **Examples**

The following example will output the values of the given array ($colors):

### **Example**

<?php   
$colors = array("red", "green", "blue", "yellow");   
  
foreach ($colors as $value) {  
  echo "$value <br>";  
}  
?>

The following example will output both the keys and the values of the given array ($age):

### **Example**

<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
  
foreach($age as $x => $val) {  
  echo "$x = $val<br>";  
}  
?>

PHP Functions

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

## PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

* A function is a block of statements that can be used repeatedly in a program.
* A function will not execute automatically 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:

### **Syntax**

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

**Note:** A function name must start with a letter or an underscore. 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 followed by brackets ():

<?php

function writeMsg() {

echo "Hello PHP";

}

writeMsg();

?>

<?php

**function** addFunction($num1, $num2) {

$sum = $num1 + $num2;

echo "Sum of the two numbers is : $sum";

}

addFunction(10, 20);

addFunction(200, 30);

?>

<?php

function areaR($len, $wid) {

$area = $len \* $wid;

echo "area of rectangle : $area ";

}

areaR(10, 20);

echo "<br>";

areaR(200, 30);

?>

<?php

function fact ($n)

{

if($n <= 1)

{

return 1;

}

else

{

return $n \* fact($n - 1);

}

}

echo "Factorial of 6 is " .fact(6);

?>

<?php

function addNumbers(int $x, int $y) {

return $x + $y;

}

echo addNumbers(6, "7 days");

// since strict is NOT enabled "7days" is changed to int(7), and it will return 13

?>

To specify strict we need to set declare(strict\_types=1);. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:

<?php declare(strict\_types=1); // strict requirement

function addNumbers(int $x, int $y) {

return $x + $y;

}

echo addNumbers(6, "7 days");

// since strict is enabled and "5 days" is not an integer, an error will be thrown

?>

<?php declare(strict\_types=1); // strict requirement

function addNumbers(int $x, int $y) {

return $x + $y;

}

echo addNumbers(6,7);

// since strict is enabled and "5 days" is not an integer, an error will be thrown

?>

In PHP the declare(strict\_types = 1); directive enables strict mode. In strict mode, only a variable of exact type of the “type declaration” will be accepted, or a TypeError will be thrown.

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

<?php

function setHeight(int $minheight = 50) {

echo "The height is : $minheight <br>";

}

setHeight(350);

setHeight(); // will use the default value of 50

setHeight(135);

setHeight(80);

?>

## PHP Functions - Returning values

To let a function return a value, use the return statement:

<?php

function sum(int $x, int $y) {

$z = $x + $y;

return $z;

}

echo "5 + 10 = " . sum(5,10) . "<br>";

echo "7 + 13 = " . sum(7,13) . "<br>";

echo "2 + 4 = " . sum(2,4);

?>

## PHP Return Type Declarations

PHP 7 also supports Type Declarations for the return statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon ( : ) and the type right before the opening curly ( {)bracket when declaring the function.

In the following example we specify the return type for the function:

<?php declare(strict\_types=1); // strict requirement

function addNumbers(float $a, float $b) : float {

return $a + $b;

}

echo addNumbers(1.2, 5.2);

?>

<?php declare(strict\_types=1); // strict requirement

function addNumbers(float $a, float $b) : int {

return (int)($a + $b);

}

echo addNumbers(1.2, 5.2);

?>

=============