
Js
CSS
HTML
php
php
</>
<code>
</...>
[#]

What is PHP ?

- **P**HP **H**ypertext **P**reprocessor
- Server-side scripting language.
- .php extension can embedded into an HTML source (contain text, HTML, CSS, JavaScript, and PHP code)
- Object oriented
- Integrated with popular database like MySql or MongoDB
- Runs on various platforms (Windows, Linux, Unix, Mac OS X) Web Server (Apache, IIS, nginx)
- Free to download and use
- PHP 7.0 was released in Dec 2015.

Why do we use PHP ?

- Blogging system on the web (WordPress)

- Largest social network (Facebook)

- Easy enough to be a beginner's first server side language

Web Server

- Apache HTTP Server 2.4

  https://httpd.apache.org/download.cgi

- IIS
- Nginx

Database Server

- MySql

  https://www.mysql.com/downloads/

- MongoDB

  https://www.mongodb.com/download-center

Tools

- XAMPP

  https://www.apachefriends.org/download.html

- MAMP

  https://www.mamp.info/en/downloads/

- WAMP

  http://www.wampserver.com/en/

PHP Syntax

1. XML Style

   <?php ….. ?>

2. Short-open Style (SGML-style)

   <? …. ?> (Set the short_open_tag = On)

3. HTML script Style

   <script language=“PHP”>….</script>

PHP Syntax

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<input type=“text” value=“<?php echo "Hello World!";?>”>

</body>
</html>
```

PHP Syntax

1. One line comment

```
<?php echo “Hello World”;
//C++ style single line comment
# this is also a single line comment      ?>
```

2. Multi line comment

```
<?php
    /* echo “Hello World”;
       echo “I want to be friendly”; */
?>
```

PHP Syntax

- Keyword classes, functions is NOT case sensitive

e.g.

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

- Variable is Case sensitive

e.g.

```
<?php
        $color = "red";
        echo "My car is " . $color ;         //My car is red
        echo "My house is " . $COLOR ; //My house is
        echo "My boat is " . $coLOR ;     //My boat is
 ?>
```

PHP Syntax

- Statements are expressions terminated by semicolons ;
- Braces make blocks { }
- Whitespace insensitive

Echo vs Print

- Echo does’t return a value
- Print always returs 1

  e.g.

```
print <<<END
    This uses the "here document" syntax to output
    multiple lines with $variable interpolation. Note
    that the here document terminator must appear on a
    line with just a semicolon no extra whitespace!
END;
```

- Printf

  e.g.

```
printf("There are %u million bicycles",$number);
```

PHP Variables

- Starts with the dollar sign ($).
- No command for declaring a variable (Dynamic Typing)

  e.g. $month='May';

  $age=18;
- Cannot start with a number
- Contain alpha-numeric characters and underscores one or more characters (A-z, 0-9, and _) e.g. productId, product_name
- Case sensitive
- assigned with the = operator

  e.g. $MyName = 'Banana';

PHP Variable variable

```
e.g.    $a = 'hello’;
        $$a = 'world’;
        echo "$a $hello";          // hello world

e.g.    ${'a' . 'b'} = 'hello there’;
        echo $ab;                  // hello there

e.g.    $b='b’;
        ${$a . $b} = 'hello there’;
        echo $hellob;              // hello there
```

PHP Data Types

Integers : whole numbers, without a decimal point, like 4195.

Doubles : floating-point numbers, like 3.14159 or 49.1.

Booleans : have only two possible values either true or false.

e.g.

```
$false_int=0;
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_emptystr = "";
$false_str = “0";
```

NULL : is a special type that only has one value: NULL.

PHP Data Types

Strings : sequences of characters, like 'PHP’

```
$x = "Hello world!";   // Hello world!
$y = 'Hello world!’;    // Hello world!
```

```
$variable = "name";
$word = 'My $variable will not print’;   // My $variable will not print
$word = "My $variable will print";       // My name will print
```

backslash (\) are replaced with special characters
\n , \r, \t ,\$,\“,\\ e.g. “My name \’ s John.”

PHP Data Types

Arrays : named and indexed collections of other values.

e.g. $arrNumber[0] = “0”;

Objects : instances of programmer-defined classes

Resources : special variables that hold references to resources external to PHP (such as database connections).

PHP Constants

- Like variables but cannot be changed
- No $ sign before the constant name
- Automatically global across the entire script
- Boolean, integer, float and string can be contained in constants
- Syntax

  define(name, value, case-insensitive=false)

PHP Magic Constants

- **__LINE__** //20

- **__FILE__**
  e.g. dirname(__FILE__); // C:\Apache24\htdocs

- **__FUNCTION__**

- **__CLASS__**

- **__METHOD__**

PHP Operators

- Arithmetic operators

- Comparison operators

- Logical operators

- Assignment operators

- Conditional operators

- Increment/Decrement operators

- String operators

Arithmetic Operators

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Comparison Operators

- Operator ===

  e.g. $x = 100; $y = "100";
  $x === $y // bool(false)

  Returns true if $x is equal to $y, and they are of the same type

- Operator !==

  e.g. $x = 100; $y = "100";
  $x !== $y // bool(true)

  Returns true if $x is not equal to $y, or they are not of the same type

Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

$variable = (condition) ? (true return value) : (false return value);

e.g.

```
$score = 5;
$result = ($score > 2 ? true : false);       // bool(true)
echo ‘You are a ’.($score >= 5 ? ‘genius’ : ‘nobody’);
// You are a genius
```

Increment/Decrement Operators

- Pre-increment

  e.g. $x = 10;
  echo ++$x; //11

- Post-increment

  e.g. $x = 10;
  echo $x++; //10

String Operators

- Concatenation .

```
e.g.        $txt1 = "Hello";
            $txt2 = " world!";
            echo $txt1 . $txt2;     //Hello world!
```

- Concatenation assignment .=

```
e.g.        $txt1 = "Hello";
            $txt2 = " world!";
            $txt1 .= $txt2;
            echo $txt1;             //Hello world!
```

Conditional Statements

- if statement

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

- if...else statement

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

- if...elseif

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

Switch Statement

- 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;
  }

While Loops

- while

  while (*condition is true*) {

  code to be executed;

  }

- do…while

  do {

  code to be executed;

  } while (*condition is true*);

- Break statement

  used to terminate the execution of a loop prematurely.

- Continue statement

  used to halt the current iteration of a loop but it does not terminate the loop.

For Loops

- for

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

- foreach

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

e.g.

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

foreach ($colors as $value) {
    echo "$value <br>";
}
//red
//green
//blue
//yellow
```

Arrays

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

Index Array

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

Or

$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";

echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
 // I like Volvo, BMW and Toyota.

echo count($cars);
 // 3
```

Loop an Index Array

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

// for($x = 0; $x < count($cars); $x++)

for($x = 0; $x < $arrlength; $x++) {
      echo $cars[$x];
}

// Volvo
// BMW
// Toyota
```

Associative Arrays

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

Or

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";

echo "Peter is " . $age['Peter'] . " years old.";
// Peter is 35 years old.
```

Loop an Associative Array

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
}
```

```
// Key=Peter, Value=35
// Key=Ben, Value=37
// Key=Joe, Value=43
```

Multidimemsional Arrays

```
$cars = array
            (
                array("Volvo",22,18),
                array("BMW",15,13),
                array("Saab",5,2),
                array("Land Rover",17,15)
            );

 echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2];
// Volvo: In stock: 22, sold: 18

echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2];
// BMW: In stock: 15, sold: 13
```

Multidimemsional Arrays

```
$marks = array(
        “john" => array (
          “course" => “physics”,
          “point" => 32
        ),
        "zara" => array (
          “course " => “physics”,
          "point" => 39
        )
      );
echo “John get $marks[‘john’][‘point’] point in $marks[‘john'][‘course’] ;
// John get 32 point in physics
```

Functions

- Internal Functions

  http://php.net/manual/en/funcref.php

- User-defined Function

Functions

- Not execute immediately when a page loads.
- Executed by a call to the function
- Function name can start with a letter or underscore

Functions

- Syntax

```
function functionName($args) {
    code to be executed;
    return $data;
}
```

- Syntax

```
function functionName($args=defaultValue) {
    code to be executed;
}
```

- Syntax

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

Variable Functions

```
function sayHello() {
    echo "Hello";
}

$function_holder = "sayHello";
$function_holder();                    // Hello
```

Variable Scope

- Local variables
- Global variables
- Static variables

Local Variable

```
$x = 4;

function assignx () {
      $x = 0;
     print "\$x inside function is $x. <br />";
}

assignx();
print "\$x outside of function is $x. <br />";

// $x inside function is 0.
// $x outside of function is 4.
```

Static Variable

```
function keep_track() {
    STATIC $count = 0;
    $count++;
    print $count;
 }

$count=9;

keep_track();    // 1
keep_track();    // 2
keep_track();    // 3
```

Global Variable

```
$somevar = 15;

   function addit() {
      GLOBAL $somevar;
      $somevar++;
      print "Somevar is $somevar";
   }

   addit();

// Somevar is 16
```

Superglobals

- Always accessible
- Access from any where

e.g. $GLOBALS e.g. $GLOBALS['x']

$_SERVER e.g. $_SERVER['PHP_SELF']

$_POST e.g. $POST['x']

$_GET e.g. $GET['x']

$_SESSION e.g. $_SESSION[‘username']

$_COOKIE e.g. $_ COOKIE[‘allow']

http://php.net/manual/en/reserved.variables.php

File Inclusion

- include()

  e.g. include("menu.php");

- require()

  e.g. require("menu.php");

- include_once()

  e.g. include_once(“important.php");

- require_once()

  e.g. require_once(" important.php");

Absolute vs Relative Paths

- Absolute

  e.g.

  ```
  include(__DIR__ . '/menu.php');
  // C:\Apache24\htdocs\menu.php
  ```

- Relative

  e.g.

  ```
  <a href="../../template/contact.php">
  ```

Error Handling

- **Notice**: Undefined offset: 2
  in **C:\Apache24\htdocs\db.php** on line **5**

- **Parse error**: syntax error, unexpected end of file
  in **C:\Apache24\htdocs\db.php** on line **107**

- **Fatal error**: Call to undefined function a() in
  /var/www/html/mypage.php on line **34**

Error Handling

- For display in the browser
  display_errors = on

- Send errors to the web server
  log_errors = on

- **Missing Semicolons**
- **Missing Dollar Signs**
- **Array Index**
- **Missing Parentheses and curly brackets**
- **Troubling Quotes e.g. echo “test’;**

Error Report Levels

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Custom Error Handler

- die()

```
e.g. if(!file_exists("welcome.txt"))
        die("File not found");
     else
        $file=fopen("welcome.txt","r");


// File not found
```

Custom Error Handler

Syntax

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

e.g.

```
function myErrorHandler($errno, $errstr, $errfile, $errline) {
    echo "<b>Custom error:</b> [$errno] $errstr<br>";
    echo " Error on line $errline in $errfile<br>";
}

set_error_handler("myErrorHandler");    // Set user-defined error

$test=2;

// Trigger error
if ($test>1) {
   trigger_error("A custom error has been triggered");
}
```

Custom error: [1024] A custom error has been triggered
Error on line 14 in C:\Apache24\htdocs\test.php

Exceptions Handling

- Try
- Throw
- Catch

e.g.
```
try {
        $test=2;
        if ($test>1)
                throw new Exception(“Value must be 1 or below”);

        // Code following an exception is not executed.
        echo 'Never executed';
 }catch (Exception $e) {
                echo 'Caught exception: ',  $e->getMessage(), “<br>";
}

 // Continue execution
echo 'Hello World';
```

Caught exception: Value must be 1 or below
Hello World

End