ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Web Programming

PhP

References

- PHP 8 Basics, 2020, Springer
  - https://link.springer.com/chapter/10.1007/978-1-4842-8082-9_2
- The Absolute Beginner's Guide to HTML and CSS, 2023, Springer
  - https://link.springer.com/chapter/10.1007/978-1-4842-9250-1_7
- W3C Tutorial
  - https://www.w3schools.com/php
  - https://www.w3schools.com/html
  - https://www.w3schools.com/js
- Additional Topics
  - JQuery: https://www.w3schools.com/jquery
  - Bootstrap 5.0: https://www.w3schools.com/bootstrap5
  - Laravel/Blade Framework 11.0: https://www.w3schools.in/laravel

PhP

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

- HTML
- CSS
- JavaScript

PhP

- PHP supports such a wide range of databases.
- PHP will run on most platforms.
- Compatible with almost all servers used nowadays.
- PHP is free to download and open source.
- PHP is a server scripting language.
- Easy to learn & **large community**.

- powerful tool for making dynamic and interactive Web pages.

PhP

Server side

- PhP

Client side

- HTML
- CSS
- JavaScript

PhP

What is a PHP File?

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

PhP

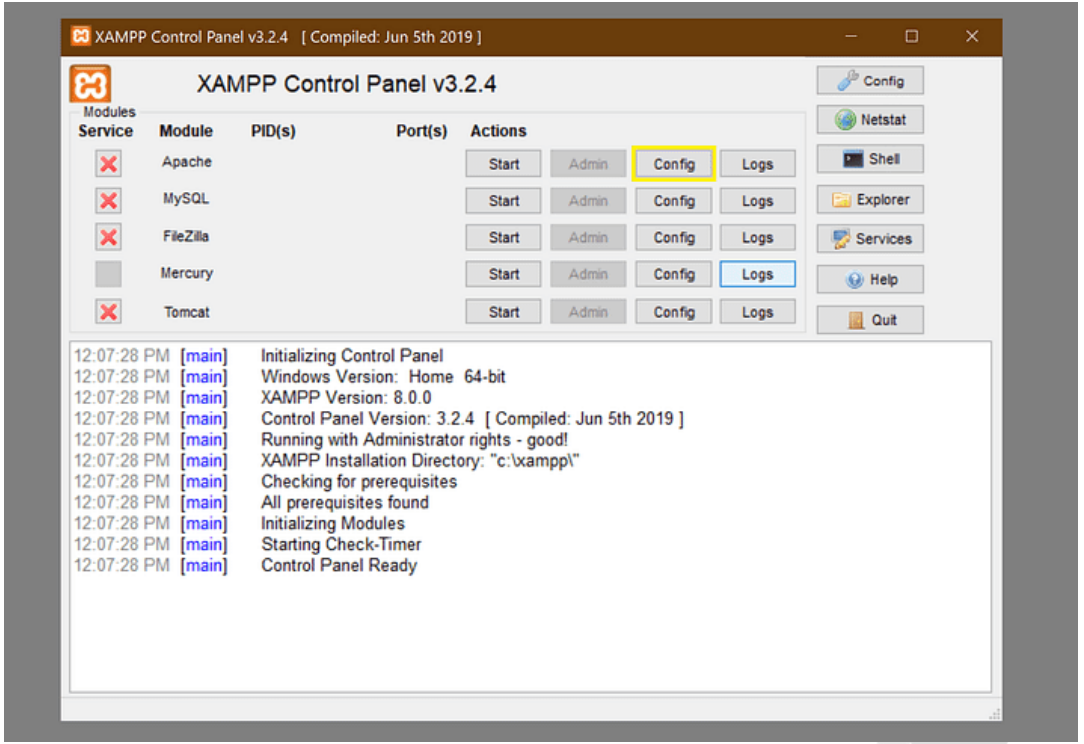
XAMPP Setup

XAMPP

Localhost

- X stands for Cross-Platform,
  A stands for Apache,
- M stands for MYSQL

- Ps stand for PHP and Perl



PhP

PHP Language Basics

PHP Tags

This is the most popular and effective PHP tag style and looks like this:

```
<?php

?>
```

Commenting PHP

```
// This is a single-line comment
 # This is also a single-line comment
/* This is a multi-line comment */
```

Use # to write single-line comments

```
<?php
# this is a comment in PHP, a single line comment
?>
```

Use // to also write single-line comments

```
<?php
// this is also a comment in PHP, a single line comment
?>
```

Use /* ...*/ to write multi-line comments

```
<?php
/* this is a multi line comment
Name: Web Code Geeks
*/
?>
```

PhP

Hello World

```
<?php
print("Hello World");
echo "Hello World";
?>
```

- print returns a value. It always returns 1.
- echo can take a comma delimited list of arguments to output.

PhP

Variables in PHP

Any type of variable in PHP starts with a leading dollar **sign ($)** and is assigned a variable type using the = (equals) sign.

```
$txt = "PHP";
```

The value of a variable is the value of its most recent assignment.

In PHP, variables do not need to be declared before assignment

.

The main data types used to construct variables are:

- Integers - whole numbers like 23, 1254, 964 etc
- Doubles - floating-point numbers like 46.2, 733.21 etc
- Booleans - only two possible values, true or false
- Strings - set of characters, like Web Code Geeks
- Arrays - named and indexed collections of other values
- Objects - instances of predefined classes

PhP

The following snippet shows all of these data types declared as variables:

```
<?php
$intNum = 472;
$doubleNum = 29.3;
$boolean = true;
$string = ’Web Code Geeks’;
$array = array("Pineapple", "Grapefruit", "Banana");
// creating a class before declaring an object variable
class person {
function agePrint() {
$age = 5;
echo "This person is $age years old!";
}
}
// creating a new object of type person
$object = new person;
?>
```

PhP

```php
<?php
$txt = "PHP";
echo "I love $txt!";
?>
```

I love PHP!

A simple `.php` file with both HTML code and PHP code:

```php
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php

echo "Hello World!";

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

My first PHP page

Hello World!

PHP Case Sensitivity

In PHP, keywords (e.g. `if`, `else`, `while`, `echo`, etc.), classes, functions, and user-defined functions are **not case-sensitive**.

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

`ECHO` is the same as `echo`:

```
<!DOCTYPE html>
<html>
<body>
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body> </html>
```

Hello World!
Hello World!
Hello World!

PhP

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

`$COLOR` is *not* same as `$color`:

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

PhP

PHP Variables

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
$x = 5; $y = "John";
```

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)

```
$x = 5; $y = 4;
echo $x + $y;
```

PhP

Get the Type

To get the data type of a variable, use the `var_dump()` function.

The `var_dump()` function returns the data type and the value:

```php
$x = 5;
var_dump($x);
```

```
int(5)
```

```php
var_dump(5);
var_dump("John");
var_dump(3.14);
var_dump(true);
var_dump([2, 3, 56]);
var_dump(NULL);
```

```
int(5)
string(4) "John"
float(3.14)
bool(true)
array(3) { [0]=> int(2) [1]=> int(3) [2]=>
int(56) }
NULL
```

PhP

PHP Variables Scope

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:

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

PhP

```php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
```

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

```php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
```

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

PhP

PHP 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):

```php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
```



```php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
```

PHP also stores all global variables in an array called `$GLOBALS[index]`.

PhP

PHP 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:

```
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
```

0
1
2

PhP

Display Text

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

```
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
```

PhP

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

<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo "<h2>$txt1</h2>";
echo "<p>Study PHP at $txt2</p>";
?>

</body>
</html>
```

Learn PHP

Study PHP at W3Schools.com

PhP

When using double quotes, variables can be inserted to the string as in the example above.
When using single quotes, variables have to be inserted using the . operator, like this:

```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 multiple parameters.";
```

```php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo '<h2>' . $txt1 . '</h2>';
echo '<p>Study PHP at ' . $txt2 . '</p>';
```

PhP

The PHP print Statement

```
print "Hello"; //same as: print("Hello");
```

(notice that the text can contain HTML markup)

```
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
```

PhP

```php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

print "<h2>$txt1</h2>";
print "<p>Study PHP at $txt2</p>";
```

```php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

print '<h2>' . $txt1 . '</h2>';
print '<p>Study PHP at ' . $txt2 . '</p>';
```

PHP 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

Casting allows you to change data type on variables:

```
$x = 5;
$x = (string) $x;
var_dump($x);
```

PHP Strings

Return the length of the string "Hello world!":

```
echo strlen("Hello world!");
```

12

Word Count

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

```
echo str_word_count("Hello world!")
```

2

PhP

Search For 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.

Search for the text "world" in the string "Hello world!":

```
echo strpos("Hello world!", "world");
```

6

Tip: The first character position in a string is 0 (not 1).

PhP

```php
$x = "Hello World!";
echo strtoupper($x);
```

```php
$x = "Hello World!";
echo strtolower($x);
```

```php
$x = "Hello World!";
echo str_replace("World", "Dolly", $x);
```

The `trim()` removes any whitespace from the beginning or the end:

```php
$x = " Hello World! "; echo trim($x);
```

Reverse the string "Hello World!":

```php
$x = "Hello World!"; echo strrev($x);
```

!dlroW olleH

PhP

Convert String into Array

The PHP `explode()` function splits a string into an array.

Split the string into an array. Use the space character as separator:

```php
$x = "Hello World!";
$y = explode(" ", $x);

//Use the print_r() function to display the result:
print_r($y);

/*
Result:
Array ( [0] => Hello [1] => World! )
*/
```

```
Array
(
    [0] => Hello
    [1] => World!
)
```

PhP

```php
<?PHP
$x = "Hello World! * hhh";
$y = explode(" * ", $x);

//Use the print_r() function to display the
result:
print_r($y);
?>
```

```
Array
(
    [0] => Hello World!
    [1] => hhh
)
```

PhP

String Concatenation

To concatenate, or combine, two strings you can use the . operator:

```php
$x = "Hello";
$y = "World";
$z = $x . $y;
echo $z;
```

HelloWorld

```php
$x = "Hello";
$y = "World";
$z = $x . " " . $y;
echo $z;
```

Hello World

PhP

An easier and better way is by using the power of double quotes.

```
$x = "Hello";
$y = "World";
$z = "$x $y";
echo $z;
```

PHP - Slicing Strings

Start the slice at index 6 and end the slice 5 positions later:

```
$x = "Hello World!"; echo substr($x, 6, 5);
```

World

PhP

Start the slice at index 6 and go all the way to the end:

```
$x = "Hello World!"; echo substr($x, 6);
```

World!

Get the 3 characters, starting from the "o" in world (index -5):

```
$x = "Hello World!"; echo substr($x, -5, 3);
```

orl

From the string "Hi, how are you?", get the characters starting from index 5, and continue until you reach the 3. character from the end (index -3). Should end up with "ow are y":

```
$x = "Hi, how are you?"; echo substr($x, 5, -3);
```

ow are y

PhP

Escape Character

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

<?php
$x = "We are the so-called \"Vikings\" from the north.";
echo $x;
?>

</body>
</html>
```

We are the so-called "Vikings" from the north.

PhP

Code	Result
\'	Single Quote
\"	Double Quote
\$	PHP variables
\n	New Line
\r	Carriage Return
\t	Tab
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

PhP

PHP Numbers

There are three main numeric types in PHP:

- Integer
- Float
- Number Strings

In addition, PHP has two more data types used for numbers:

- Infinity
- NaN

Check if the type of a variable is integer:

```
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
```

```
$x = 10.365;
 var_dump(is_float($x));
```

PhP

- is_finite()
- is_infinite()

```
$x = 1.9e411;
var_dump($x);
```

float(INF)

PHP Numerical Strings

The PHP `is_numeric()` function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

```
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));
$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
```

bool(true)
bool(true)
bool(true)
bool(false)

Change Data Type

Casting in PHP is done with these statements:

- `(string)` - Converts to data type String
- `(int)` - Converts to data type Integer
- `(float)` - Converts to data type Float
- `(bool)` - Converts to data type Boolean
- `(array)` - Converts to data type Array
- `(object)` - Converts to data type Object
- `(unset)` - Converts to data type NULL

```php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;
echo "<br>";
// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
```

23465
23465

PhP

PHP Math

```php
echo(min(0, 150, 30, 20, -8, -200));
echo(max(0, 150, 30, 20, -8, -200));


echo(abs(-6.7));


echo(sqrt(64));


echo(rand(10, 100));


echo(rand());
```

PhP

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.

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

```
define(name, value);
```

PhP

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

 echo GREETING;
```

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

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

</body>
</html>
```

PhP

PHP const Keyword

```php
const MYCAR = "Volvo";
echo MYCAR;
```

```php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
```

Constants are Global

Constants are automatically global and can be used across the entire script.

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

function myTest() {
  echo GREETING;
}

myTest();
```

PhP

PHP Magic Constants

PHP has nine predefined constants that change value depending on where they are used, and therefor they are called "magic constants".

These magic constants are written with a double underscore at the start and the end, except for the ClassName::class constant

PhP

Constant	Description
__CLASS__	If used inside a class, the class name is returned.
__DIR__	The directory of the file.
__FILE__	The file name including the full path.
__FUNCTION__	If inside a function, the function name is returned.
__LINE__	The current line number.
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.
__TRAIT__	If used inside a trait, the trait name is returned.
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.

PhP

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

<?php
class Fruits {
  public function myValue(){
    return __CLASS__;
  }
}
$kiwi = new Fruits();
echo $kiwi->myValue();
?>

</body>
</html>
```

Fruits

PhP

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

<?php
echo __DIR__;
?>

</body>
</html>
```

C:\awesomesites\w3schools\php

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

https://www.w3schools.com/php/php_operators.asp

PhP

PHP Conditional Assignment Operators

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

PhP

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

<?php
   // if empty($user) = TRUE, set $status = "anonymous"
   echo $status = (empty($user)) ? "anonymous" : "logged in";
   echo("<br>");

   $user = "John Doe";
   // if empty($user) = FALSE, set $status = "logged in"
   echo $status = (empty($user)) ? "anonymous" : "logged in";
?>

</body>
</html>
```

anonymous
logged in

PhP

PHP 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

PHP - The if Statement

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

```
if (5 > 3)
 { echo "Have a good day!"; }
```

```
$a = 200; $b = 33; $c = 500;
if ($a > $b && $a < $c )
{
echo "Both conditions are true";
}
```

```
$a = 5;

if ($a == 2 || $a == 3 || $a == 4 || $a == 5 || $a == 6 || $a == 7) {
  echo "$a is a number between 2 and 7";
}
```

PhP

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;
}
```

PhP

Short Hand If

```php
$a = 5;

if ($a < 10) $b = "Hello";

echo $b
```

Short Hand If...Else

```php
$a = 13;

$b = $a < 10 ? "Hello" : "Good Bye";

echo $b;
```

PhP

The PHP switch Statement

```
switch (expression) {
  case label1:
    //code block
    break;
  case label2:
    //code block;
    break;
  case label3:
    //code block
    break;
  default:
    //code block
}
```

This is how it works:

- The *expression* is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The `break` keyword breaks out of the switch block
- The `default` code block is executed if there is no match

PhP

```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!";
}
```

Warning: If you omit the `break` statement in a case that is not the last, and that case gets a match, the next case will also be executed even if the evaluation does not match the case!

What happens if we remove the `break` statement from case "red"?

PhP

PHP Loops

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

PhP

```php
$i = 1;
while ($i < 6) {
  echo $i;
  $i++;
}
```

```php
$i = 1;
while ($i < 6) {
  if ($i == 3) break;
  echo $i;
  $i++;
}
```

```php
$i = 0;
while ($i < 6) {
  $i++;
  if ($i == 3) continue;
  echo $i;
}
```

```php
$i = 1;
while ($i < 6):
  echo $i;
  $i++;
endwhile;
```

```php
$i = 0;
while ($i < 100) {
  $i+=10;
  echo $i "<br>";
}
```

```php
$i = 1;

do {
  echo $i;
  $i++;
} while ($i < 6);
```

PhP

PHP for Loop

```
for (expression1, expression2, expression3) {
  // code block
}
```

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

PHP foreach Loop

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

foreach ($colors as $x) {
  echo "$x <br>";
}
```

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

foreach ($members as $x => $y) {
  echo "$x : $y <br>";
}
```

PhP

The foreach Loop on Objects

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

foreach ($colors as $x) {
  echo "$x <br>";
}
```

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

foreach ($members as $x => $y) {
  echo "$x : $y <br>";
}
```

PhP

PHP Functions

https://www.w3schools.com/php/php_functions.asp

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP User Defined 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

PhP

```php
function myMessage() {
  echo "Hello world!";
}

myMessage();
```

```php
function setHeight($minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
```

```php
function familyName($fname) {
  echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
```

```php
function sum($x, $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

Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in.

To turn a function argument into a reference, the & operator is used:

```
function add_five(&$value) {
  $value += 5;
}

$num = 2;
add_five($num);
echo $num;
```

PhP

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

<?php
function add(&$value) {
  $value += 20;
}

$num = 20;
add($num);
echo $num;
?>

</body>
</html>
```

PhP

PHP Functions - Returning values

```php
function sum($x, $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

Variable Number of Arguments

By using the `...` operator in front of the function parameter, the function accepts an unknown number of arguments. This is also called a variadic function.

```php
function sumMyNumbers(...$x) {
  $n = 0;
  $len = count($x);
  for($i = 0; $i < $len; $i++) {
    $n += $x[$i];
  }
  return $n;
}

$a = sumMyNumbers(5, 2, 6, 2, 7, 7);
echo $a;
```

PhP

PHP is a Loosely Typed Language

In the examples above, notice that we did not have to tell PHP which data type the variable is.
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 the following example we try to send both a number and a string to the function without using `strict`:

```php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
```

PhP

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

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```