PHP

Server-Side Basics

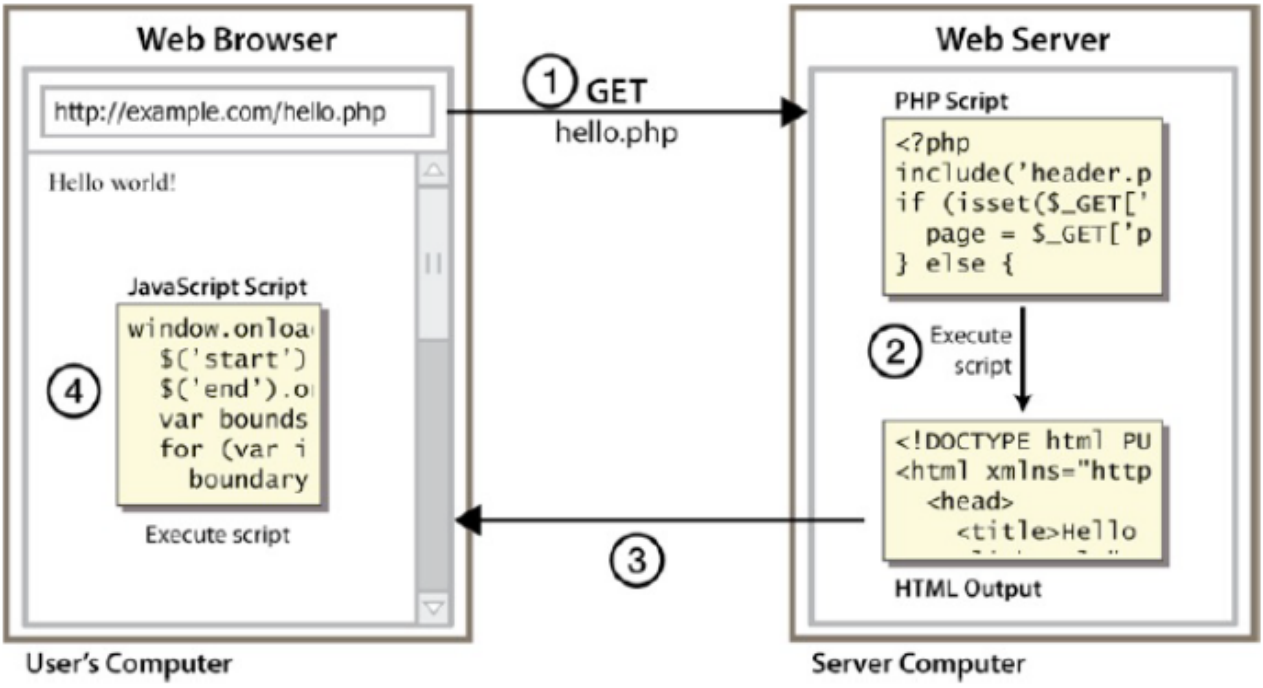
URLs and web servers

What happens when we type an URL like http://server/path/file ?

- **Scenario 1:**
  - computer looks up server's IP address using DNS
  - browser connects to that IP address and requests the file
  - web server grabs the file from its local file system and sends it back to the browser
- **Scenario 2:**
  - same as above but the requested file is a program instead!
  - instead of sending back the requested file it is executed on the web server
  - The output of the program/script is the result sent back to the client

Cofinanciado por:
MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

Server-Side scripting (and client too)

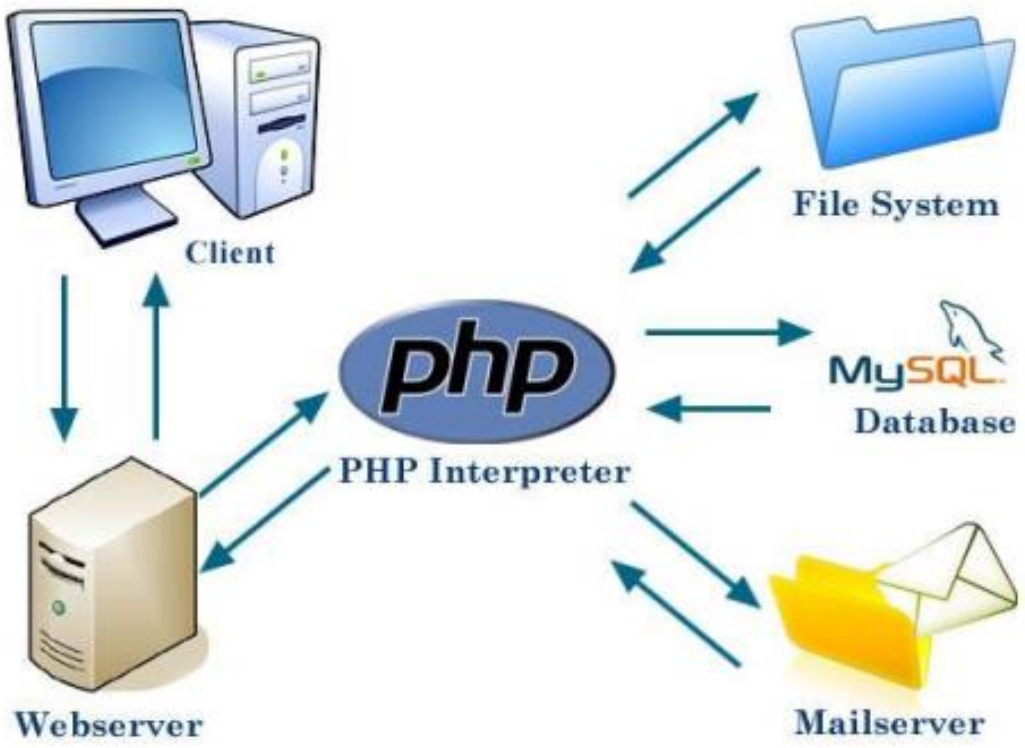


Introduction to PHP

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP scripts are executed on the server / backend
    - The web server's software allows it to run those programs and send their output back to the client

Cofinanciado por:

Architecture



Installation

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

Cofinanciado por:
MADEIRA 14-20
REGIÃO AUTÓNOMA DA MADEIRA
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Social Europeu

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"

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
?>
```

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

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

</body>
</html>
```

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

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Comments

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

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
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>
```

Case Sensitivity

- all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

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

Cofinanciado por:

- However; all variable names are case-sensitive.

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

Console output: print

```php
<?php
print "Hello, World!";
print "Escape \"chars\" are the SAME as in Java!";
print "You can have
line breaks in a string.";
print 'A string can use "single-quotes". It\'s cool!';
?>
```

Output:

Hello world! Escape "chars" are the SAME as in Java! You can have line breaks in a string. A string can use "single-quotes". It's cool!

output

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Variables

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

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)
- Always **implicitly declared** by assignment (**type is not written**)

Output variables:

```php
<?php
$txt = "PHP language";
echo "I love $txt!";
// OR
echo "I love " . $txt . "!";
?>
```

Variables Scope

PHP has three different variable scopes:

- local
- global
- static

global scope

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

Cofinanciado por:

```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>";
?>
```

local scope

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

```php
<?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>";
?>
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

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
<?php
$x = 5;
$y = 10;

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

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

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:

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

myTest();
myTest();
?>
```

Cofinanciado por:
MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Data types

PHP supports the following data types:

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

String Type

- **zero-based indexing** using bracket notation
- there is **no char type**; each letter is itself a String
  - $favorite_food = "Ethiopian";
  - print $favorite_food[2];
    - Output: h
- string **concatenation operator is . (period), not +**
  - 5 + "2 turtle doves" == 7
  - 5 . "2 turtle doves" == "52 turtle doves”

String Functions

```
# index 0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);
$cmp = strcmp($name, "Brian Le"); // compare

$index = strpos($name, "e"); // outputs 2
echo strpos("Hello world!","world"); // outputs 6

$sub = substr($name, 9); // outputs Hatcher
$name = strtoupper($name);
```

Cofinanciado por:

```
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
```

Types auto convert

- PHP converts between types automatically in many cases:
  - string → int auto-conversion on +
    - 5 + "7" is 12
  - int → float auto-conversion on /

- type-cast with (type):
  - $age = (int) "21";

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:**

```php
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

Cofinanciado por:

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.

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

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

if...else...elseif Statements

```php
<?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!";
}
?>
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

switch Statement

```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!";
}
?>
```

while Loops

```php
<?php
$x = 1;

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

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

for Loops

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

foreach

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

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

User Defined Functions

```php
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

Function Arguments

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

familyName("Jani");
familyName("Hege");
familyName("Kai Jim");
?>
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Default Argument Value

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

setHeight(350);
setHeight(); // will use the default value of 50
?>
```

Function Returning values

```php
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

Arrays

```php
<?php

// $cars = array("Volvo", "BMW", "Toyota"); // Funciona com
todas as versões do PHP
$cars = ["Volvo", "BMW", "Toyota"];
echo "I like " . $cars[0] . ", " . $cars[1] . " and " .
$cars[2] . ".";
?>
```

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

Cofinanciado por:

Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

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

or the index can be assigned manually:

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

Get The Length of an Array - The count() Function

The `count()` function is used to return the length (the number of elements) of an array:

```
<?php
$cars = ["Volvo", "BMW", "Toyota"];
echo count($cars);
?>
```

Loop Through an Indexed Array

```
<?php
$cars = ["Volvo", "BMW", "Toyota"];
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
// Funciona com todas as versões do PHP
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

// PHP 5.4 introduziu uma nova sintaxe
$age = ["Peter"=>"35", "Ben"=>"37", "Joe"=>"43"];
```

or:

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

The named keys can then be used in a script:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

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

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

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Sorting arrays

PHP - Sort Functions For Arrays

- `sort()` - sort arrays in ascending order
- `rsort()` - sort arrays in descending order
- `asort()` - sort associative arrays in ascending order, according to the value
- `ksort()` - sort associative arrays in ascending order, according to the key
- `arsort()` - sort associative arrays in descending order, according to the value
- `krsort()` - sort associative arrays in descending order, according to the key

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

Descending Order

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

associative arrays in ascending order, according to the value

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

Mais recursos

- DOCS: https://secure.php.net/manual/en/
- Interactive Tutorial: https://www.learn-php.org/en/Welcome
- https://learnxinyminutes.com/docs/pt-br/php-pt/
- Video: https://www.youtube.com/watch?v=ZdP0KM49IVk
- https://www.w3schools.com/PHP/default.asp

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu