UPT UNIVERSIDADE PORTUCALENSE

CONTENT

1. Advantages and disadvantages
2. Two different ways to use
3. Variables
    1. Types
    2. Scope
4. External variables (forms with GET and POST)
5. Operators
    1. Arithmetic
    2. Assignment
    3. Comparison
    4. Increment / decrement
    5. Logical
    6. For strings
    7. For arrays
    8. Conditional
6. Conditional instructions
7. Iterative instructions
8. Functions
9. Session variables
10. MySQL connection

Advantages and disadvantages

Disadvantages

- It's an interpreted language (not compiled)
    - Slower than C or C++
- It's not natively object-oriented

Advantages

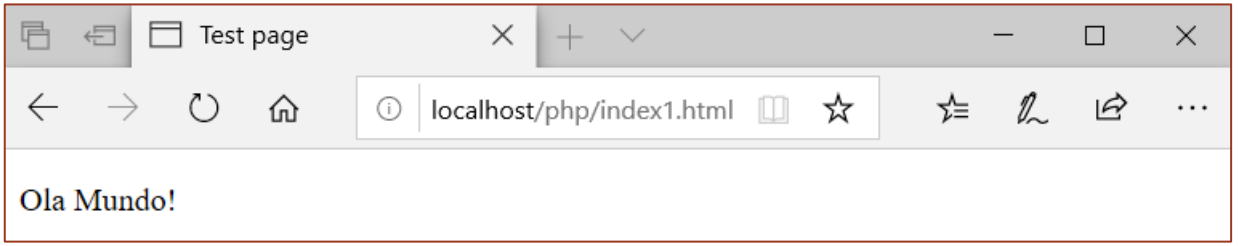
- Allows the fast development of applications
    - Does not need compilation and connection
- Resulting applications are stable and independent of the browser used
    - Also allows choosing the server platform
- Can connect to the most used database systems
- Partially supports object-oriented programming

  https://www.w3schools.com/php/php_oop_classes_objects.asp
- Runs in several platforms and is more powerful than ASP or JSP
- Has many users and solutions already implemented

Two different ways to use

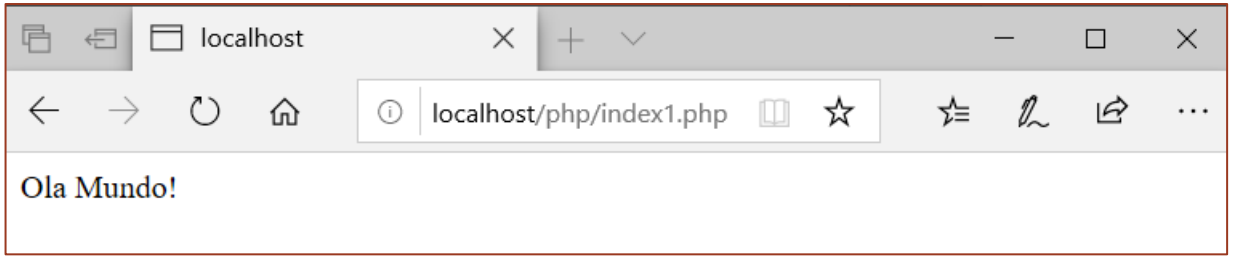
index1.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title>Test page</title>
6      </head>
7      <body>
8          <p>
9          <?php echo "Ola Mundo!"; ?>
10         </p>
11     </body>
12 </html>
```



index1.php

```
1  <?php
2      echo "Ola Mundo!";
3  ?>
```



Variable types

Group	Type	Example	Conversion
Scalar	boolean	`$b1 = True;` `$b2 = False;`	`$b3 = (bool)$i1; // True`

	integer	`$i1 = 123;`
		`$i2 = -123;`
		`$i3 = 0123; // octal (83)`
		`$i4 = 0x1A; // hexadecimal (26)`
			`$i5 = (int)$b1; // 1`
			`$i6 = (int)$b2; // 0`
			`$i7 = (int)$f1; // 1`
			`$i8 = (int)$f2; // 1`
			`$i9 = 1+ "123"  // 124`

	float	`$f1 = 1.234;`
		`$f2 = 1.734;`
		`$f3 = 1.2e3; // 1200`
		`$f4 = 7E-10; // 0.0000000007`
			`$f5 = 1 + "1.23" // 2.23`

	string	`$s1 = 'ab\nc'; // ab\nc`
		`$s2 = "ab\nc"; // ab`
		`// c`
		`$s3 = <<<FIM // ab`
		`ab // c`
		`c`
		`FIM;`
			`$s4 = "Price: $i1"`
			`// Price: 123`
			`$s5 = $s1{0} // a`
			`$s6 = $s1{1} // b`

FALSE values:
- Boolean FALSE
- Integer 0
- Real 0.0
- Empty String ""
- String "0"
- Array with no elements
- Special type NULL

Special characters:
- \n: new line
- \r: return
- \t: tab
- \\: backslash
- \$: dollar sign
- \": quote
- \[0-7]{1,3}: an octal
- \x[0-9A-Fa-f]{1,2}: an hexadecimal

Variable types

Group	Type	Example

Composed	array	`$a = array {1 => 'um', 2=> 'dois', 3 => 'tres'};`
		`$a[3] = 30; // array {1 => 'um', 2=> 'dois', 3 => 30}`
		`$a[ ] = 'quinze'; // array {1 => 'um', 2=> 'dois', 3 => 'tres', 4 =>'quinze'}`
		`unset($a[2]); // array {1 => 'um', 3 => 'tres' , 4 => 'quinze'}`
		
		`$b = array {'cor' => 'azul', 'forma' => 'quadrada'}`
		`$b['cor'] = 'azul';`
		`$b['forma'] = 'quadrada';`
		
		`$c = array{`
		`'cores' => array {'a'=>'azul','b'=>'amarelo','c'=>'branco'},`
		`'numeros' => array (1,2,3,4)`
		`}`

	object	`$o = new teste();` `$o->testar();`
Special	resource	`$ligacao = mysql_connect($server,$user,$password);`
	NULL	`$n = NULL // variable with no value`

Variables

- Name
  - Always starts with $.
  - Second character must be a letter or _
  - The remainder can be letters or numbers
- Reference (&)

```
$a = 'Ana';
$b = &$a; // $b points to $a
$b = 'Joana'; // also alters $a
```

- Scope
  - Global (next slide)
  - Static: keep their value even out of scope
- Variables of variables

```
$a = 'Ola';
$$a = 'Mundo';
echo "$a ${$a}";
echo "$a $Ola";
```

- Predefined vectorial variables

Vector	Description
$GLOBALS	Reference to each global variable
$_SERVER	Server variables (IP, port)
$_GET	Variables obtained with GET
$_POST	Variables obtained with POST
$_COOKIE	Variables obtained with *cookies*
$_FILES	Variables referring to file loading
$_ENV	Environment variables
$_REQUEST	Variables requested to the user
$_SESSION	Session variables

Variable scope

Global variables

- The scope is extended to the included files

```
$a = 2;
include "file.php"; // $a is available in file.php, which can generate warnings
// "require" has a similar behaviour → file.php can generate fatal warnings
// we can also use "include_once" and "require_once" (to avoid repetitions)
```

- The scope is not extended through functions…

```
$b = 3;
function func(){
    echo $b; // reference to a local variable (undefined)
}
```

- … unless it is explicitly specified

```
$a = 3;
$b = 2;
function mult(){
global $a, $b;
$b = $a*$b;
}
mult();
echo $b; // 6
```

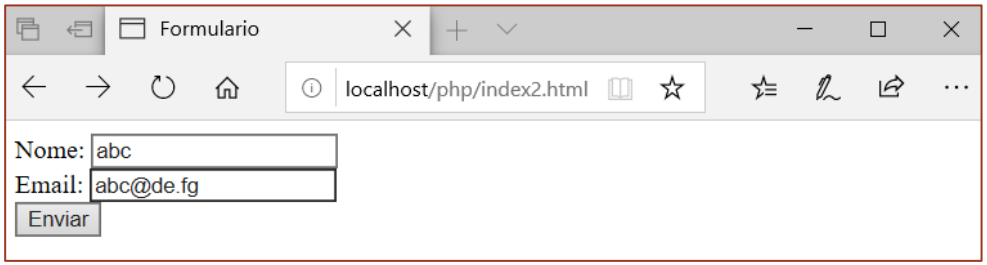
External variables (forms with GET and POST)

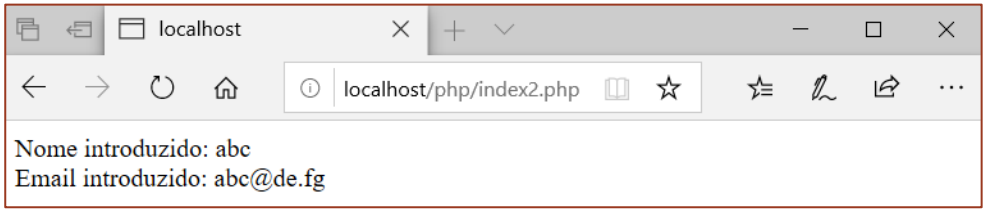
index2.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8">
5          <title>Formulario</title>
6      </head>
7      <body>
8          <form action="index2.php" method="post">
9              Nome: <input type="text" name="nome"></br>
10             Email: <input type="email" name="email"></br>
11             <input type="submit" value="Enviar">
12         </form>
13     </body>
14 </html>
```

index2.php

```
1  <?php
2      echo "Nome introduzido: " . $_POST['nome'] . "</br>";
3      echo "Email introduzido: " .$_POST['email'];
4  ?>
```






```
setcookie("mycookie","test",time()+3600);
// define a cookie named mycookie with value "test" with 1h (3600s) timespan

echo $_COOKIE['mycookie'];
// access the cookie
```

Arithmetic operators

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

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

Assignment operators

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

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

Comparison operators

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.

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

Increment / decrement operators

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

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

Logical operators

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

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

Operators for strings

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

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

Operators for 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

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

Conditional 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

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

Conditional instructions

```
if($a == 0) {
    print "nenhum";
} elseif ($a == 1) {
    print "um";
} else {
    print "muitos";
}
```

```
switch($a) {
    case 0:
        print "nenhum";
        break;
     case 1:
        print "um";
        break;
    default:
        print "muitos";
}
```

Iterative instructions

```
$i = 0;
while($i < 10){
   print $i;
   i++;
}
```

```
$i = 0;
do {
   print $i;
   i++;
} while($i < 10)
```

```
for($i=0; $i<10; $i++){
   print $i;
}
```

```
$a = array (1,2,3,4);
foreach($a as $v){
   print $v;
}
```

```
$i = 0;
while(++$i){
   switch($i){
      case 3:
         print "tres";
         break 1; // breaks the switch only
      case 6:
         print "seis";
         break 2; // breaks the switch and the while
      default:
         break;
}
```

```
$a = array (1,2,3,4);
foreach($a as $v){
   if(($v % 2)==0) {
      continue;
   }
   impares($v);
}
```

Function

```
function cinco(){
   return 5;
}
$x = cinco(); // $x is now 5
```

```
function sum($p, $s = 1){ // arguments by omission
   return $p + $s;
}
$a = soma(2,3); // 5
$b = soma(2); // 3
```

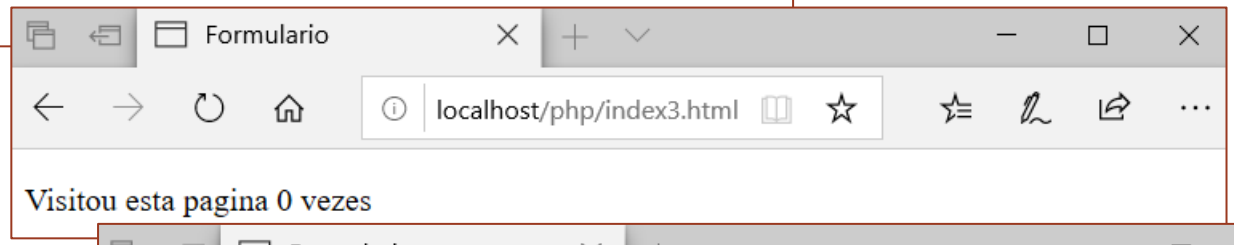
```
function conc(&$var){ // arguments by reference
   $var .= ' extra';
}
$h = 'horas';
conc($h);
echo $h; // 'horas extra'
```

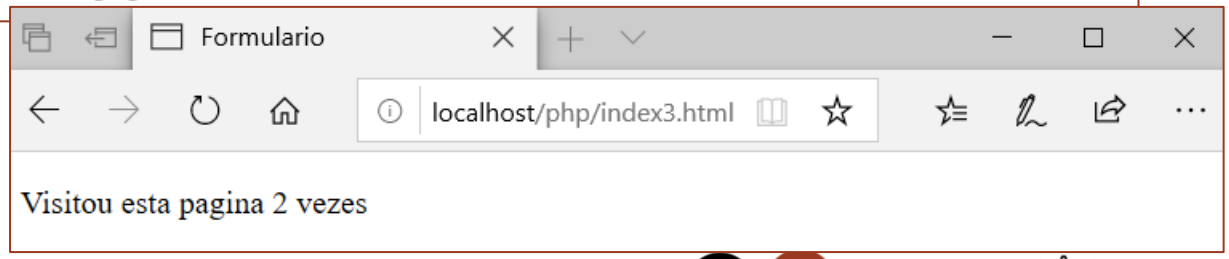
```
function vec(){
   return array (1,2,3);
}
$y = vec(); // $x = array (1,2,3)
```

Session variables

index3.html

```php
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Formulario</title>
    </head>
    <body>
        <?php
            session_start();
            if(!isset($_SESSION['contador'])){
                $_SESSION['contador'] = 0;
            } else {
                $_SESSION['contador']++;
            }
        ?>
        <p>Visitou esta pagina <?php echo $_SESSION['contador'];?> vezes</p>
    </body>
</html>
```



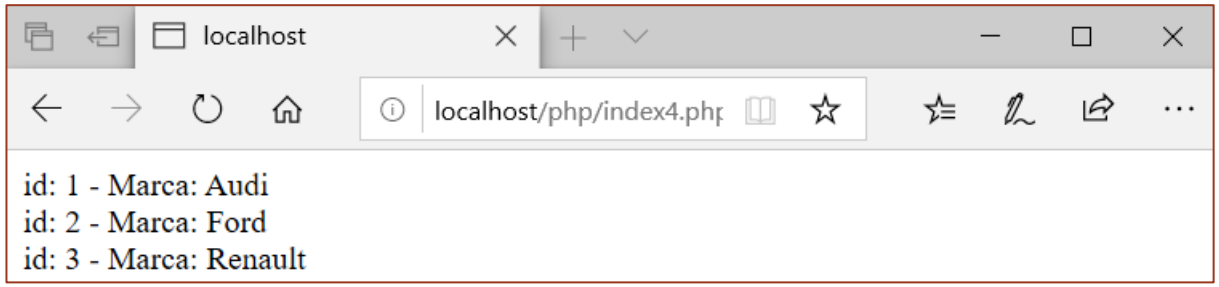


Connection to MySQL

index4.php

```php
<?php
    $servidor = "localhost";
    $user = "root";

    $ligacao = mysqli_connect($servidor,$user) or die("Sem ligação");
    mysqli_select_db($ligacao, "carros") or die("Sem DB");
    $consulta = "SELECT * FROM marca";
    $resultado = mysqli_query($ligacao,$consulta);
    if($resultado->num_rows > 0){
        while($row = $resultado->fetch_assoc()){
            echo "id: " . $row["mc_id"] . " - Marca: " . $row["mc_marca"] . "<br>";
        }
    } else {
        echo "0 results";
    }
?>
```



UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.