# Primeiros Passos em PHP

Thiago H. P. Silva

# Primeiros passos

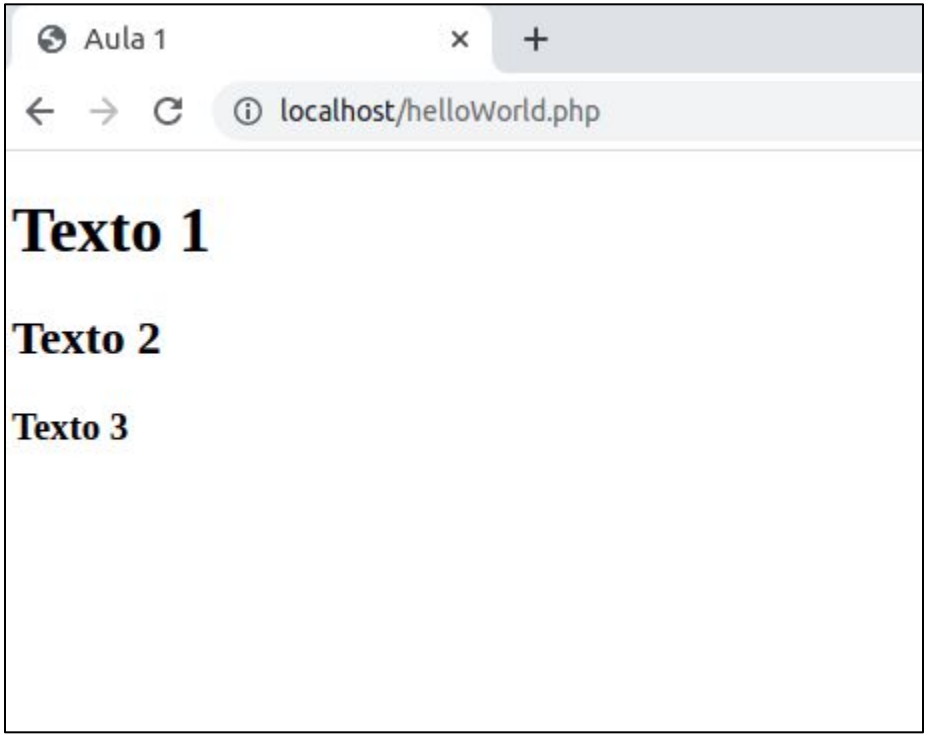
- LAMP: Linux + Apache + Mysql + PHP
- WAMP: Windows + Apache + Mysql + PHP
  - https://www.apachefriends.org/pt_br/download.html

- Rodar os programas online
  - https://paiza.io/

```
<?php
    echo "oi";
?>
```

# Exercícios PHP

Crie o seguinte documento HTML:



# Exercícios PHP

Crie o seguinte documento HTML:

```html
<!DOCTYPE html>
<html>
<head><title>Aula 1</title></head>
  <body>
  <h1>Texto 1</h1>
  <h2>Texto 2</h2>
  <h3>Texto 3</h3>
</body>
</html>
```

# Exercícios PHP

Crie 4 variáveis para guardar:

1. Título
2. Conteúdo em h1
3. Conteúdo em h2
4. Conteúdo em h3

```html
<!DOCTYPE html>
<html>
<head><title>Aula 1</title></head>
  <body>
  <h1>Texto 1</h1>
  <h2>Texto 2</h2>
  <h3>Texto 3</h3>
</body>
</html>
```

Com PHP, escreva a mesma página trocando os textos pelas variáveis.

# Exercícios PHP

```
1 <?php
2 $varT = 'ALterando o título';
3 $var1 = "Texto 1";
4 $var2 = "Texto 2";
5 $var3 = "Texto 3";
6 ?>
7 <!DOCTYPE html>
8 <html>
9 <head><title><?php echo $varT; ?></title></head>
10   <body>
11   <h1><?php echo $var1; ?></h1>
12   <h2><?php echo $var2; ?></h2>
13   <h3><?php echo $var3; ?></h3>
14 </body>
15 </html>
```

Aula 1

localhost/helloWorld.php

# Texto 1

## Texto 2

### Texto 3

Open | helloWorld.php /var/www/html | Save

```php
<?php
$varT = 'ALterando o título';
$var1 = "Texto 1";
$var2 = "Texto 2";
$var3 = "Texto 3";
?>
<!DOCTYPE html>
<html>
<head><title><?php echo $varT; ?></title></head>
  <body>
  <h1><?php echo $var1; ?></h1>
  <h2><?php echo $var2; ?></h2>
  <h3><?php echo $var3; ?></h3>
</body>
</html>
```

view-source:localhost/hel

view-source:localhost/helloWorld.php

Line wrap

```html
<!DOCTYPE html>
<html>
<head><title>ALterando o título</title></head>
  <body>
  <h1>Texto 1</h1>
  <h2>Texto 2</h2>
  <h3>Texto 3</h3>
</body>
</html>
```

| Arithmetic Operators | | |
|---|---|---|
| **Example** | **Name** | **Result** |
| +$a | Identity | Conversion of *$a* to int or float as appropriate. |
| -$a | Negation | Opposite of *$a*. |
| $a + $b | Addition | Sum of *$a* and *$b*. |
| $a - $b | Subtraction | Difference of *$a* and *$b*. |
| $a * $b | Multiplication | Product of *$a* and *$b*. |
| $a / $b | Division | Quotient of *$a* and *$b*. |
| $a % $b | Modulo | Remainder of *$a* divided by *$b*. |
| $a ** $b | Exponentiation | Result of raising *$a* to the *$b*'th power. |

1000000000000000

```php
<?php
$a = "10";
$b = "15";
echo $a**$b;
?>
```

| Logical Operators | | |
|---|---|---|
| **Example** | **Name** | **Result** |
| $a and $b | And | **true** if both *$a* and *$b* are **true**. |
| $a or $b | Or | **true** if either *$a* or *$b* is **true**. |
| $a xor $b | Xor | **true** if either *$a* or *$b* is **true**, but not both. |
| ! $a | Not | **true** if *$a* is not **true**. |
| $a && $b | And | **true** if both *$a* and *$b* are **true**. |
| $a \|\| $b | Or | **true** if either *$a* or *$b* is **true**. |

**Bitwise Operators**

| Example | Name | Result |
|---|---|---|
| **$a & $b** | And | Bits that are set in both *$a* and *$b* are set. |
| **$a \| $b** | Or (inclusive or) | Bits that are set in either *$a* or *$b* are set. |
| **$a ^ $b** | Xor (exclusive or) | Bits that are set in *$a* or *$b* but not both are set. |
| **~ $a** | Not | Bits that are set in *$a* are not set, and vice versa. |
| **$a << $b** | Shift left | Shift the bits of *$a* *$b* steps to the left (each step means "multiply by two") |
| **$a >> $b** | Shift right | Shift the bits of *$a* *$b* steps to the right (each step means "divide by two") |

**Bitwise Operators**

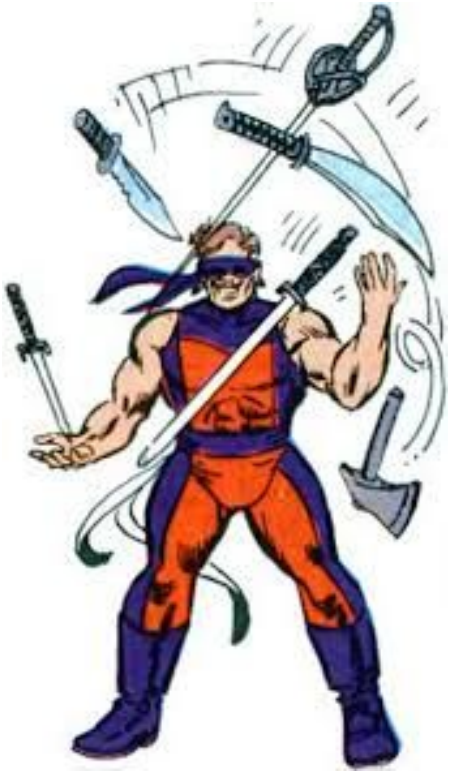
| Example | Name | Result |
|---|---|---|
| $a & $b | And | Bits that are set in both *$a* and *$b* are set. |
| $a \| $b | Or (inclusive or) | Bits that are set in either *$a* or *$b* are set. |
| $a ^ $b | Xor (exclusive or) | Bits that are set in *$a* or *$b* but not both are set. |
| ~ $a | Not | Bits that are set in *$a* are not set, and vice versa. |
| $a << $b | Shift left | Shift the bits of *$a* *$b* steps to the left (each step means "multiply by two") |
| $a >> $b | Shift right | Shift the bits of *$a* *$b* steps to the right (each step means "divide by two") |

4 & 5= 4

0100 & 0101= 0100

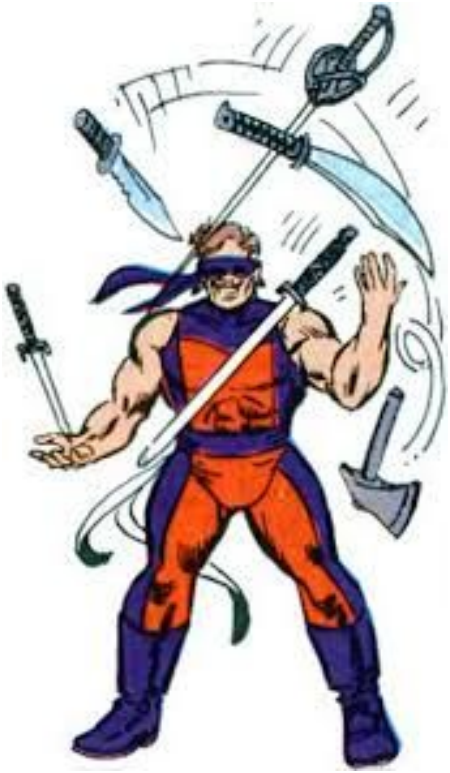
```php
<?php
$a = 4;
$b = 5;
$result = 4 & 5;
echo $a . " & " . $b . "= " . $result . "<br>";
echo sprintf('%04b', $a) . " & " . sprintf('%04b', $b) . "= " . sprintf('%04b', $result);
?>
```

# PHP: Type juggling



# PHP: Type juggling

```php
<?php
$foo = "1";  // $foo is string (ASCII 49)
$foo *= 2;   // $foo is now an integer (2)
$foo = $foo * 1.3;  // $foo is now a float (2.6)
$foo = 5 * "10 Little Piggies"; // $foo is integer (50)
$foo = 5 * "10 Small Pigs";     // $foo is integer (50)
?>
```



# PHP: Type Casting

- (int), (integer) - cast to int
- (bool), (boolean) - cast to bool
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object
- (unset) - cast to NULL

```php
<?php
$a    = 'car'; // $a is a string
$a[0] = 'b';   // $a is still a string
echo $a;       // bar
?>
```

```php
<?php
$foo = 10;            // $foo is an integer
$str = "$foo";        // $str is a string
$fst = (string) $foo; // $fst is also a string
?>
```

# Execution Operators

localhost/helloWorld.php

localhost/helloWorld.php

```
total 28K
drwxrwxrwx 2 root root 4,0K mar  9 16:14 .
drwxrwxrwx 3 root root 4,0K mar  4 12:15 ..
-rw-rw-r-- 1 th   th     57 mar  9 16:14 helloWorld.php
-rw-r--r-- 1 root root  11K mar  4 12:07 index.html
-rw-rw-r-- 1 th   th     31 mar  4 12:13 info.php
```

Open | helloWorld.php /var/www/html | Save

```php
<?php
$output = `ls -alh`;
echo "<pre>$output</pre>";
?>
```

**Comparison Operators**

| Example | Name | Result |
|---|---|---|
| $a == $b | Equal | **true** if *$a* is equal to *$b* after type juggling. |
| $a === $b | Identical | **true** if *$a* is equal to *$b*, and they are of the same type. |
| $a != $b | Not equal | **true** if *$a* is not equal to *$b* after type juggling. |
| $a <> $b | Not equal | **true** if *$a* is not equal to *$b* after type juggling. |
| $a !== $b | Not identical | **true** if *$a* is not equal to *$b*, or they are not of the same type. |
| $a < $b | Less than | **true** if *$a* is strictly less than *$b*. |
| $a > $b | Greater than | **true** if *$a* is strictly greater than *$b*. |
| $a <= $b | Less than or equal to | **true** if *$a* is less than or equal to *$b*. |
| $a >= $b | Greater than or equal to | **true** if *$a* is greater than or equal to *$b*. |
| $a <=> $b | Spaceship | An int less than, equal to, or greater than zero when *$a* is less than, equal to, or greater than *$b*, respectively. |

# Spaceship

```php
// Integers
echo 1 <=> 1; // 0
echo 1 <=> 2; // -1
echo 2 <=> 1; // 1

// Floats
echo 1.5 <=> 1.5; // 0
echo 1.5 <=> 2.5; // -1
echo 2.5 <=> 1.5; // 1

// Strings
echo "a" <=> "a"; // 0
echo "a" <=> "b"; // -1
echo "b" <=> "a"; // 1

echo "a" <=> "aa"; // -1
echo "zz" <=> "aa"; // 1

// Arrays
echo [] <=> []; // 0
echo [1, 2, 3] <=> [1, 2, 3]; // 0
echo [1, 2, 3] <=> []; // 1
echo [1, 2, 3] <=> [1, 2, 1]; // 1
echo [1, 2, 3] <=> [1, 2, 4]; // -1
```

**Comparison with Various Types**

| Type of Operand 1 | Type of Operand 2 | Result |
|---|---|---|
| null or string | string | Convert `null` to "", numerical or lexical comparison |
| bool or null | anything | Convert both sides to bool, `false` < `true` |
| object | object | Built-in classes can define its own comparison, different classes are uncomparable, same class see Object Comparison |
| string, resource, int or float | string, resource, int or float | Translate strings and resources to numbers, usual math |
| array | array | Array with fewer members is smaller, if key from operand 1 is not found in operand 2 then arrays are uncomparable, otherwise - compare value by value (see following example) |
| object | anything | object is always greater |
| array | anything | array is always greater |

# Comparison Operators

bool(true) bool(false)

```php
<?php
    var_dump("1" == 1);
    var_dump("1" === 1);
?>
```

# Comparison Operators

```php
<?php
var_dump(0 == "a");
var_dump("1" == "01");
var_dump("10" == "1e1");
var_dump(100 == "1e2");

switch ("a") {
case 0:
    echo "0";
    break;
case "a":
    echo "a";
    break;
}
?>
```

# Comparison Operators

```php
<?php
var_dump(0 == "a");
var_dump("1" == "01");
var_dump("10" == "1e1");
var_dump(100 == "1e2");

switch ("a") {
case 0:
    echo "0";
    break;
case "a":
    echo "a";
    break;
}
?>
```

Output of the above example in PHP 7:

```
bool(true)
bool(true)
bool(true)
bool(true)
0
```

Output of the above example in PHP 8:

```
bool(false)
bool(true)
bool(true)
bool(true)
a
```

**Warning** Prior to PHP 8.0.0, if a string is compared to a number or a numeric string then the string was converted to a number before performing the comparison.

**Warning**

## Comparison of floating point numbers

Because of the way floats are represented internally, you should not test two floats for equality.

# Incrementing/Decrementing Operators

| Increment/decrement Operators | | |
|---|---|---|
| **Example** | **Name** | **Effect** |
| ++$a | Pre-increment | Increments $a by one, then returns $a. |
| $a++ | Post-increment | Returns $a, then increments $a by one. |
| --$a | Pre-decrement | Decrements $a by one, then returns $a. |
| $a-- | Post-decrement | Returns $a, then decrements $a by one. |

```php
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++ . "<br />\n";
echo "Should be 6: " . $a . "<br />\n";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br />\n";
echo "Should be 6: " . $a . "<br />\n";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a-- . "<br />\n";
echo "Should be 4: " . $a . "<br />\n";

echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . --$a . "<br />\n";
echo "Should be 4: " . $a . "<br />\n";
?>
```

# String Operators

```php
<?php
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
?>
```

# String Operators

```php
<?php
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
?>
```

```php
$a = '12345';

// This works:
echo "qwe{$a}rty"; // qwe12345rty, using braces
echo "qwe" . $a . "rty"; // qwe12345rty, concatenation used
```

# Control Structures

```
if (expr)
        statement
```

# Control Structures

if (expr)
    statement

```
<?php

$a = 30;
$b = 20;

if ($a > $b)
  echo "a is bigger than b";

?>
```

# Control Structures

if (expr)
    statement

```
<?php

$a = 20;
$b = 20;

if ($a > $b)
  echo "a is bigger than b";

?>
```

# Control Structures

```
if (expr) {
        statement 1;
        statement 2;
        …
}
```

```
if ($a > $b) {
  echo "a is bigger than b";
  $b = $a;
}
```

# Control Structures

```
if (expr) {
        statement 1;
        statement 2;
        …
} else {
        statement 3;
        statement 4;
        …
}
```

# Control Structures

```
if (expr) {
        statement 1;
        statement 2;
        …
} else {
        statement 3;
        statement 4;
        …
}
```

```php
if ($a > $b) {
  echo "a is greater than b";
} else {
  echo "a is NOT greater than b";
}
```

# Control Structures

```
if (expr1) {
        statement 1;
        statement 2;
        …
} elseif (expr2) {
        statement 3;
        statement 4;
        …
} else {
        statement 5;
        statement 6;
        …
}
```

# Control Structures

```
if (expr1) {
        statement 1;
        statement 2;
        …
} elseif (expr2) {
        statement 3;
        statement 4;
        …
} else {
        statement 5;
        statement 6;
        …
}
```

```php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
```

# Control Structures

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

# Control Structures

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

# Control Structures

```php
<?php
if ($a == 5):
    echo "a equals 5";
    echo "...";
elseif ($a == 6):
    echo "a equals 6";
    echo "!!!";
else:
    echo "a is neither 5 nor 6";
endif;
?>
```

# Control Structures

```
while (expr)
        statement
```

```
while (expr):
    statement
    ...
endwhile;
```

# Control Structures

- Imprima na tela os números a sequência de números inteiros de 1 a 10.
- Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

# Control Structures

- Imprima na tela os números a sequência de números inteiros de 1 a 10.
- Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

12345678910
10987654321

```php
<?php
$i = 1;
while ($i <= 10) {
    echo $i++;
}

echo "<br>";

$i = 10;
while ($i >0) {
    echo $i--;
}

?>
```

# Control Structures

```
do {
        statement
} while (expr);
```

# Control Structures

```
do {
        statement
} while (expr);
```

# Control Structures

Usando **do-while**…

- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10.
- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

# Control Structures

Usando **do-while**…

- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10.
- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

12345678910
10987654321

```php
<?php

$i = 0;

do {
    echo ++$i;
} while ($i < 10);

echo "<br>";

do {
    echo $i--;
} while ($i > 0);
?>
```

# Control Structures

```
for (expr1; expr2; expr3)
        statement
```

# Control Structures

```
for (expr1; expr2; expr3)
    statement
```

- expr1 → Avaliada uma vez no início do loop
- expr2 → Avaliada a cada início de uma iteração
  - true → loop continua
  - false → loop finaliza
- expr3 → Avaliada a cada fim de uma iteração

# Control Structures

Usando **for**…

- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10.
- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

# Control Structures

Usando **for**…

- Imprima na tela os números a sequência de números inteiros de 1 a 10.
- Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

12345678910
10987654321

```php
<?php

for($i = 1; $i<=10;$i++)
  echo $i;

echo "<br>";

for($i = 10; $i>0;$i--)
  echo $i;

?>
```

# Control Structures

Usando **for**…

- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10.
- ❑ Imprima na tela os números a sequência de números inteiros de 1 a 10 em ordem decrescente.

12345678910
10987654321

```php
<?php
for ($i = 1; $i <= 10; print $i++);
echo "<br>";
for ($i = 10; $i >0; print $i--);
?>
```

# Control Structures

```
foreach (iterable_expression as $value)
    statement

foreach (iterable_expression as $key => $value)
    statement
```

# Control Structures

```
foreach (iterable_expression as $value)
    statement

foreach (iterable_expression as $key => $value)
    statement
```

Usando **foreach**…

- ❑ Crie um arranjo: $arr = array(1, 2, 3, 4);
- ❑ Imprima cada um dos elementos.
- ❑ Imprima cada par chave-valor, imprima-a no seguinte formato: chave => valor

# Control Structures

```
foreach (iterable_expression as $value)
    statement
```

```
foreach (iterable_expression as $key => $value)
    statement
```

```
0 => 1
1 => 2
2 => 3
3 => 4
Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
```

```php
<?php

$arr = array(1, 2, 3, 4);

foreach ($arr as $key => $value) {
    echo "{$key} => {$value} " . "<br>";
}

print_r($arr);

?>
```