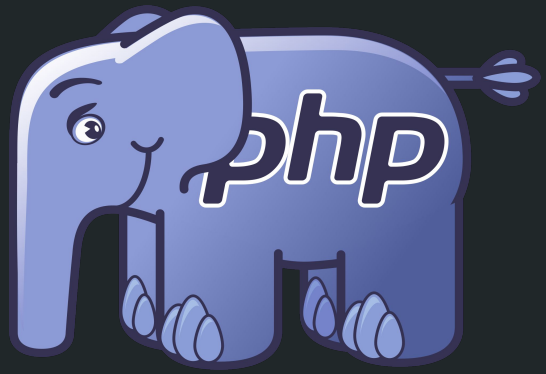
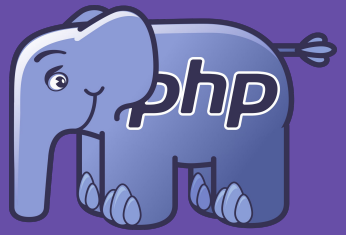
Curso PHP e MySQL



_Estruturas de Controle em PHP

> Expressões, operadores de comparação, operadores lógicos, If, else, while, do-while, switch, sintaxe alternativa para estruturas de controle e capturando entradas pela linha de comando.

Adriel Sales



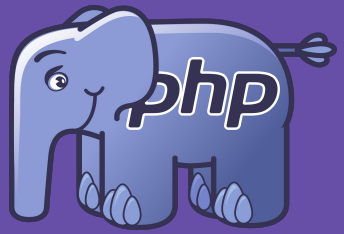
_Estruturas de controle

Um script PHP é construído por **uma série de instruções**. Uma instrução pode ser uma **atribuição**, uma **chamada de função**, um **laço de repetição**, uma **instrução condicional**, ou mesmo uma instrução que não faz nada (um comando vazio). Instruções podem ser agrupadas através do encapsulamento de um grupo de comandos com chaves.

if

else

while



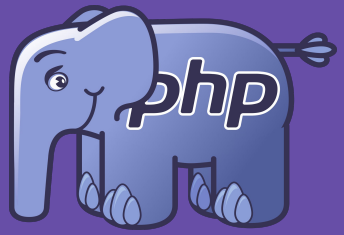
_expressões

Expressões são os blocos de construção mais importantes do PHP. No PHP, quase tudo o que você escreve são expressões. A maneira mais simples e ainda mais precisa de definir uma expressão é "**tudo o que tem um valor**". As formas mais básicas de expressões são **constantes e variáveis**. Exemplo:

$a = 5;

Depois desta atribuição, você pode esperar que o valor de $a seja 5 também, assim se você escrever $b = $a, você pode esperar que ele se comporte da mesma forma que se você escrevesse $b = 5.

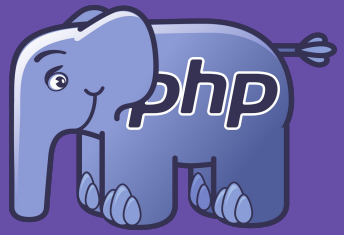
```
$b = $a = 5;        /* atribui o valor cinco às variáveis $a e $b */
```



_Operadores de Comparação

Operadores de comparação, como os seus nomes implicam, **permitem que você compare dois valores**.

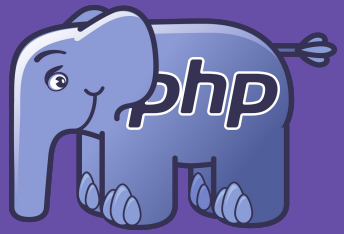
Exemplo	Nome	Resultado
$a == $b	Igual	Verdadeiro (**true**) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (**true**) se $a é igual a $b, e eles são do mesmo tipo.
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (introduzido no PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.
$a <=> $b	Spaceship (nave espacial)	Um integer menor que, igual a ou maior que zero quando *$a* é, respectivamente, menor que, igual a ou maior que *$b*. Disponível a partir do PHP 7.



_Operadores lógicos

Os operadores lógicos são utilizados quando precisamos realizar operações sobre uma expressão e retornar um valor booleano como verdadeiro ou falso.

Exemplo	Nome	Resultado
$a and $b	E	Verdadeiro (**true**) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.



_if

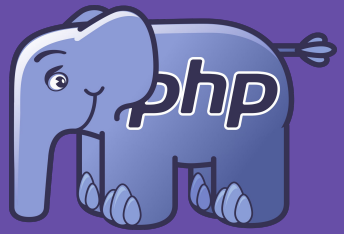
A estrutura condicional **if** é um dos recursos mais importantes nas linguagens de programação, pois **permite a execução condicional de fragmentos de código**.

```
if (expressao)
        fragmento_codigo
```

As expressões são avaliadas por seus valores Booleanos. Se uma expressão for avaliada como **true** (verdadeiro), o PHP vai executar a declaração, e se avaliá-la como **false** (falso) - irá ignorá-la.

Neste exemplo a seguir, será exibido ‘a is bigger than b’ se $a for maior que $b

```php
<?php
if ($a > $b)
  echo "a is bigger than b";
?>
```



_if

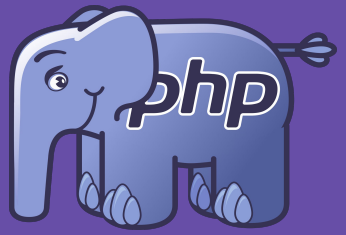
Para utilizar mais de uma declaração após a condicional ser executada, pode-se agrupar várias declarações utilizando parênteses “{ }” sinalizando a abertura e o fechamento do bloco condicional.

Por exemplo, o código abaixo exibirá ‘**a is bigger than b**’ se **$a** for maior que **$b**, e na linha seguinte **atribuirá o valor de $a em $b**:

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

```php
<?php
if( $a == 1 || $a == 2 ) {
    if( $b == 3 || $b == 4 ) {
        if( $c == 5 || $ d == 6 ) {
            //Do something here.
        }
    }
}
?>
```

Observação:
A declaração If pode ser aninhada indefinidamente dentro de outras declarações if, fornecendo completa flexibilidade para execução condicional de várias partes do programa.

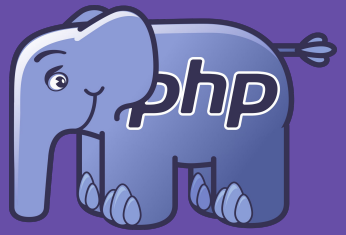


_else

O else estende a instrução if para executar outra caso a expressão no if retornar false. Por exemplo, o código a seguir exibirá a is greater than b se $a for maior que $b, e a is NOT greater than b caso contrário:

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

A instrução **else** só é executada **se a expressão de avaliação do if for avaliada como false**.



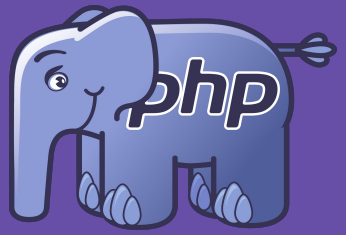
_while

Laços while são os mais simples tipos de laços do PHP. O formato básico de uma declaração while é:

```
while (expressao)
    fragmento_codigo
```

O propósito da declaração while é simples. Ele dirá ao PHP para executar as declarações aninhadas repetidamente, enquanto a expressão do while for avaliada como true.

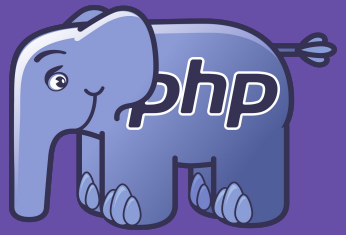
```php
$i = 1;
while ($i <= 10) {
    echo $i++;  /* the printed value would be
                   $i before the increment
                   (post-increment) */
}
```



_do while

O laço do-while é similar ao laço while, com exceção que a expressão de avaliação é verificada ao final de cada iteração em vez de no começo. **A maior diferença para o laço while é que a primeira iteração do laço do-while sempre é executada**.

```php
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```



_for

O laço **for** é outro poderoso recurso do PHP.

```
for (expressao_1; expressao_2; expressao_3) {
      Fragmento_codigo
}
```

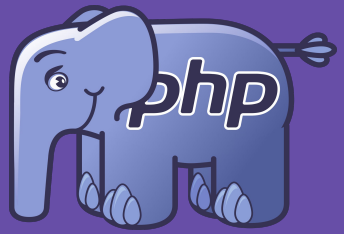
- Expressao_1 é avaliada (executada), uma vez, incondicionalmente, no início do laço.
- No começo de cada iteração a expressao_2 é avaliada. Se avaliada como true, o laço continuará e as instruções aninhadas serão executadas. Se avaliada como false, a execução do laço terminará.
- No final de cada iteração, a expressao_3 é avaliada (executada).

```php
/* exemplo 1 */

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

```php
/* exemplo 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
```



_foreach

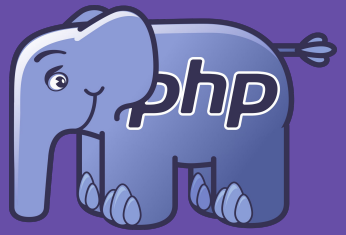
O foreach fornece uma maneira fácil de iterar sobre arrays e objetos e funciona apenas com estrutura desses tipos. Possui duas sintaxes:

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

```
foreach (iterable_expression as $chave => $valor)
    Statement
```

```php
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$valor) {
    $valor = $valor * 2;
}
// $arr is now array(2, 4, 6, 8)
```

```php
foreach ($arr as $chave => $valor) {
    // $arr[3] será atualizado com cada valor de $arr...
    echo "{$chave} => {$valor} ";
    print_r($arr);
}
```



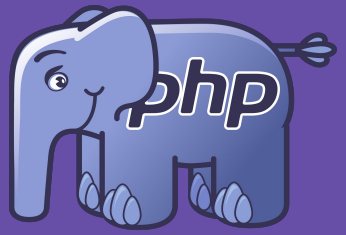
_switch

A declaração switch **é similar a uma série de declarações if** na mesma expressão. Em muitos casos, se deseja comparar as mesmas variáveis (ou expressões), com diferentes valores, e executar pedaços diferentes de código dependendo de qual valor ela é igual. Esta é exatamente a serventia da declaração switch.

```php
<?php
if ($i == 0) {
    echo "i é igual a 0";
} elseif ($i == 1) {
    echo "i é igual a 1";
} elseif ($i == 2) {
    echo "i é igual a 2";
}
```



```php
switch ($i) {
    case 0:
        echo "i é igual a 0";
        break;
    case 1:
        echo "i é igual a 1";
        break;
    case 2:
        echo "i é igual a 2";
        break;
}
?>
```

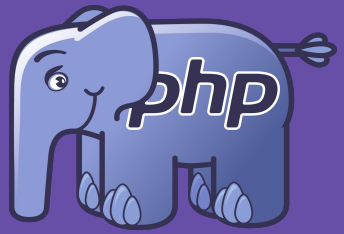


_sintaxe alternativa

O PHP oferece uma sintaxe alternativa para algumas estruturas de controle; a saber, if, while, for, foreach, e switch. Em cada caso, basicamente a sintaxe alternativa é trocar a chave de abertura por dois pontos (:) e a chave de fechamento por endif;, endwhile;, endfor;, endforeach;, ou endswitch;, respectivamente.

```
<?php if ($a == 5): ?>
A é igual a 5
<?php endif; ?>
```

```
<?php switch ($foo): ?>
    <?php case 1: ?>
    ...
<?php endswitch ?>
```



_capturando entradas pela linha de comando

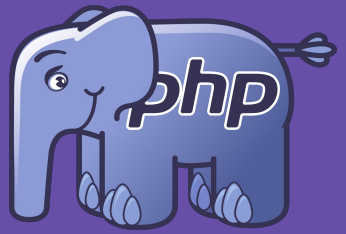
Para capturarmos um valor de entrada digitado pelo usuário na linha de comando, podemos utilizar o seguinte código PHP:

```
$teclado = trim(fgets(STDIN));
```

Onde:

- **$teclado** → variável que receberá o valor;
- **trim** → função PHP que retorna uma string com os espaços retirados no início e no final;
- **fgets** → função PHP usada para retornar o valor da linha de um arquivo aberto ou do prompt de comando.
- **STDIN (standard input stream)** → captura uma entrada na linha de comando.

```php
<?php
$line = trim(fgets(STDIN)); // Lê uma linha do STDIN
```



_atividades de fixação

Dica: utilize a função `fgets(STDIN)` para capturar o valor digitado no terminal.

1. Faça um programa PHP que:
   a. Leia o nome digitado pelo usuário;
   b. Leia o sobrenome digitado pelo usuário;
   c. Concatene (junte) o nome com o sobrenome e apresente o nome completo.

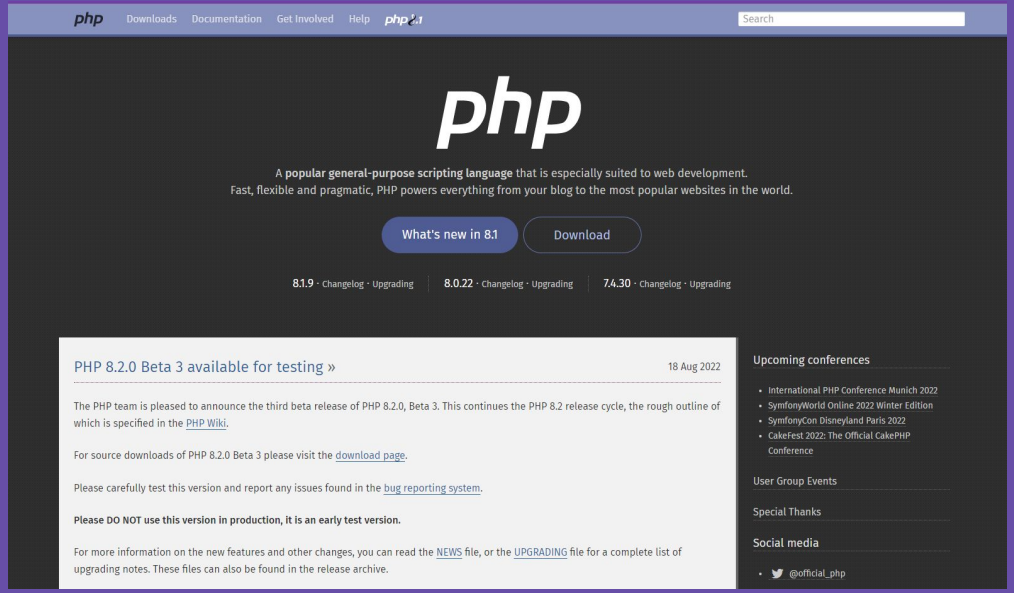
2. Faça um programa PHP que:
   a. Leia os valores de COMPRIMENTO, LARGURA e ALTURA e apresente o valor do volume de uma caixa retangular. Utilize para o cálculo a fórmula VOLUME = COMPRIMENTO * LARGURA * ALTURA.

3. Faça um programa PHP que:
   a. Obtenha o valor para a variável $ht (horas trabalhadas no mês);
   b. Obtenha o valor para a variável $vt (valor hora trabalhada):
   c. Obtenha o valor para a variável $pd (percentual de desconto);
   d. Calcule o salário bruto => $sb = $ht * $vt;
   e. Calcule o total de desconto => $td = ($pd/100)*$sb;
   f. Calcule o salário líquido => $sl = $sb – $td;
   g. Apresente os valores de: Horas trabalhadas, Salário Bruto, Desconto, Salário líquido.

4. Faça um algoritmo que leia os valores A, B e C. Mostre uma mensagem que informe se a soma de A com B é menor, maior ou igual a C.

5. Faça um algoritmo que leia um número na variável $num1 e calcule se esse número é par ou ímpar.

Fonte? Vá na fonte!



https://www.php.net

https://github.com/adrielacademico

Professor:
Adriel Sales