Este curso é de propriedade do aplicativo SoloLearn e foi traduzido por Ana Beatriz Augusto usando os recursos Reverso Context e Google Tradutor.

if...else

Afirmações condicionais executam diferentes ações para diferentes decisões.
A afirmação `if...else` é usada para executar um código específico se uma condição for verdadeira e outro código se a afirmação for falsa.

Sintaxe:

```
if (condição) {
   Código a ser executado
se a condição for
verdadeira;
} else {
   Código a ser executado
se a condição for falsa;
}
```

Você também pode usar apenas `if`, se não quiser fazer nada caso a afirmação for falsa.

elseif

Use a afirmação `if...elseif...else` para especificar novas condições a serem testadas, se a primeira condição for falsa.

Você pode adicionar quantas afirmações `elseif` quiser. Apenas note que primeiro se deve escrever uma afirmação `if`.

Sintaxe:

```
if (condição) {
  Código a ser executado se
a condição for verdadeira;
} elseif (condição) {
  Código a ser executado se
a condição for verdadeira;
} else {
   Código a ser executado
se a condição for falsa;
}
```

Por exemplo:

```php
<?php
    $age = 21;
    if($age <= 13){
        echo "Criança";
    }
    elseif($age > 13 && $age
< 19){
        echo "Adolescente";
    } else {
        echo "Adulto";
    }
    //O resultado será
"Adulto"
?>
```

Usamos o operador lógico && para combinar as duas condições e verificar se `$age` está entre 13 e 19.

As chaves podem ser apagadas se apenas há um comando dentro de `if`/`elseif`/`else`.

Por exemplo:

```php
if($age <= 13)
  echo "Criança";
else
  echo "Adulto";
```

while

Quando você está escrevendo código, talvez queira que um bloco de código seja executado várias vezes. Invés de adicionar várias linhas de código parecidas no script, podemos usar ciclos para fazer uma tarefa como essa.

O ciclo `while` executa um bloco de código enquanto a condição for verdadeira.

Sintaxe:

```
while (a condição for verdadeira){
    Código a ser executado;
}
```

Se a condição nunca se tornar falsa, a afirmação continuará sendo executada.

O exemplo abaixo primeiro dá o valor 1 a variável `$i`. Então, o ciclo `while` é executado enquanto `$i` for menor que 7. Será adicionado 1 a variável `$i` toda vez que o ciclo for executado:

```
$i = 1;
while($i < 7){
    echo "O valor é $i <br />";
    $i++;
}
```

do...while

Sempre executará o bloco de código uma vez, irá ver a condição e repetirá o ciclo enquanto a condição for verdadeira.

Sintaxe:

```
do {
   Código a ser executado;
} while (a condição for verdadeira);
```

O exemplo abaixo mostrará algum texto na tela e então irá adicionar 1 a variável `$i`. Então irá checar a condição e o ciclo continuará a ser executado, enquanto `$i` for menor igual a 7:

```
$i = 5;
do{
    echo "O número é " . $i
. "<br/>";
    $i++;
}while($i <= 7);
```

Independente se a condição for verdadeira ou falsa, o código irá ser executado pelo menos uma vez, o que pode ser necessário em algumas situações.

Note que, em um ciclo `do while`, a condição é testada DEPOIS de executar os códigos dentro do ciclo. Isso significa que o ciclo do `while` irá executar suas afirmações pelo menos uma vez, mesmo se a condição for falsa na primeira vez.

for

É usado quando você sabe quantas vezes o script deve ser executado.

```
for (início; condição; aumento) {
   Código a ser executado;
}
```

Parâmetros:
Início: Inicializa o valor inicial do ciclo.
Condição: É avaliada toda vez que o ciclo for executado, continua se for verdadeira e termina se for falsa.
Aumento: Adiciona 1 ao valor inicial

As expressões dos parâmetros podem ser vazias ou conter múltiplas expressões que são separadas por vírgulas.
No for, os parâmetros são separados por ponto e vírgula.

O exemplo abaixo mostra os números de 0 a 5:

```
for($a = 0; $a < 6; $a++){
    echo "O valor de a : ". $a . "<br />";
}
```

foreach

Apenas funciona em arrays e é usado para percorrer cada par de chave/valor em uma array.

Há duas sintaxes:

```
foreach(array as $valor){
  Código a ser executado;
}
//ou
foreach(array as $chave => $valor){
   Código a ser executado;
}
```

A primeira forma dá a volta na array. Em cada volta, o valor do elemento atual é atribuído a variável `$valor` em seguida, o ciclo passa para o próximo elemento até os elementos acabarem.

A segunda forma irá, ainda, atribuir a chave do elemento atual a variável `$chave` em cada volta.

O seguinte exemplo demonstra um ciclo que mostra os valores da array `$nomes`

```php
$nomes = array("John", "David", "Amy");
foreach($nomes as $nome) {
    echo $nome.'<br />';
}
```

switch

A afirmação switch é uma alternativa à afirmação `if...elseif...else`.

Use a afirmação `switch` para selecionar um entre vários blocos de código a serem executados.

Sintaxe:

```
switch (n) {
  case valor1:
    //código a ser executado
caso n=valor1
    break;
  case valor2:
     //código a ser executado
caso n=valor2
     break;
  default:
    // código a ser executado
caso n for diferente de todos
os valores escritos
}
```

Primeiro, nossa única expressão, `n` (uma variável, na maioria dos casos), é avaliada uma vez. Em seguida, o valor da expressão é comparado com o valor de cada `case`. Se combinam, o bloco de código associado a esse `case` é executado.

Usar afirmações `if else` aninhadas resulta em comportamento similar mas, `switch` oferece uma solução mais elegante e ótima.

Considere o seguinte exemplo que mostra a mensagem apropriada para cada dia:

```php
$hoje = 'Ter';
switch ($hoje) {
  case "Seg":
    echo "Hoje é segunda-feira.";
    break;
  case "Ter":
    echo "Hoje é terça-feira.";
    break;
  case "Qua":
    echo "Hoje é quarta-feira.";
    break;
  case "Qui":
    echo "Hoje é quinta-feira.";
    break;
  case "Sex":
    echo "Hoje é sexta-feira.";
    break;
  case "Sab":
    echo "Hoje é sábado.";
    break;
  case "Dom":
    echo "Hoje é domingo.";
    break;
  default:
    echo "Dia invalido.";
}
// O resultado será "Hoje é terça-feira."
```

O comando `break` no fim de cada `case` impede que o código execute automaticamente o próximo `case`. Se você esquecer o `break`, o PHP irá automaticamente mostrar os comandos dos próximos cases mesmo que não aconteça combinação.

O bloco `default` é executado quando não houver combinação. É um bloco opcional.

```php
$x = 5;
switch ($x) {
  case 1:
    echo "Um";
    break;
  case 2:
    echo "Dois";
    break;
  default:
    echo "Sem combinação";
}
// O resultado será "Sem combinação"
```

Esquecer o `break` faz com que o PHP continue a executar os cases até encontrar um `break`. Você pode usar este comportamento se quiser o mesmo resultado para vários cases.

```php
$dia = 'Qua';
switch ($dia) {
  case 'Seg':
    echo '1o dia da semana';
    break;
  case 'Ter':
  case 'Qua':
  case 'Qui':
    echo 'Dia de trabalho';
    break;
  case 'Sex':
    echo 'Sexta-feira!';
    break;
  default:
    echo 'Quarta-feira!';
}
// O resultado será "Dia de trabalho"
```

O resultado será o mesmo se `$dia` tivesse valor `'Ter'`, `'Qua'` ou `'Qui'`

break

Como falado anteriormente, o `break` é usado para parar o `switch` quando há combinação com algum `case`.

Se não há `break`, o código continua a ser executado. Por exemplo:

```php
$x = 1;
switch ($x) {
  case 1:
    echo "Um";
  case 2:
    echo "Dois";
  case 3:
    echo "Tres";
  default:
    echo "Sem combinacao";
}
// O resultado será "UmDoisTresSem combinacao"
```

`break` também pode ser usado para suspender a execução das estruturas `for`, `foreach`, `while`, `do-while`.

O `break` termina o `for`, `foreach`, `while`, `do-while ou switch e` continua a executar o programa a partir da linha seguinte depois do ciclo. Um `break` na parte exterior de um programa (ex: fora de um ciclo) vai parar o script.

continue

Quando usado dentro de uma estrutura de ciclos, o `continue` permite passar a frente o que permanece da atual volta do ciclo. Ele então continua a execução na validação da condição e segue para o começo da próxima volta.

O seguinte exemplo passa pelos números pares no ciclo `for`:

```
for($i = 0; $i < 10; $i++){
  if($i % 2 == 0){
    continue;
  }
  echo $i . ' ';
}
//O resultado será: 1 3 5 7 9
```

Você pode usar o `continue` com todas as estruturas de ciclo

include e require

Permitem a inserção do conteúdo de um arquivo php dentro de outro arquivo php antes do servidor executá-lo.
Incluir arquivos salva bastante trabalho. Você pode criar um cabeçalho padrão, rodapé, ou arquivo de menu para todas as suas páginas web. Então, quando precisar atualizar o cabeçalho, você pode atualizar apenas o arquivo do cabeçalho.

Suponhamos que temos um arquivo de cabeçalho padrão chamado header.php:

```php
<?php
  echo '<h1>Welcome</h1>';
?>
```

Use o include para incluir o arquivo de cabeçalho em uma página.

```
<html>
  <body>
  <?php include 'header.php'; ?>
  <p> Texto. </p>
  <p> Texto. </p>
  </body>
</html>
```

Este curso é de propriedade do aplicativo SoloLearn e foi traduzido por Ana Beatriz Augusto usando os recursos Reverso Context e Google Tradutor.