foreach ¶

(PHP 4, PHP 5, PHP 7, PHP 8)

O construtor foreach fornece uma maneira fácil de iterar sobre arrays. O foreach funciona somente em arrays e objetos, e emitirá um erro ao tentar usá-lo em uma variável com um tipo de dado diferente ou em uma variável não inicializada. Possui duas sintaxes:

foreach (iterable\_expression as $valor)

statement

foreach (iterable\_expression as $chave => $valor)

statement

A primeira forma percorre o iterável dado por iterable\_expression. A cada iteração, o valor do elemento atual é atribuído a $valor.

A segunda forma var, adicionalmente, atribuir a chave do elemento corrente a variável $chave a cada iteração.

Note que o foreach não modifica o ponteiro interno do array, que é usado por funções como current() e key().

É possível customizar a iteração em objetos.

Para modificar diretamente elementos de um array dentro de um laço, preceda $valor com &. Neste caso, o valor será atribuído por referência.

<?php

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

foreach ($arr as &$valor) {

$valor = $valor \* 2;

}

// $arr is now array(2, 4, 6, 8)

unset($valor); // quebra a referência com o último elemento

?>

Aviso

A referência ao $valor e o último elemento do array permanecerá inalterado mesmo após a iteração do foreach. É recomendado destruí-lo utilizando a função unset(). Caso contrário você experienciará o seguinte comportamento:

<?php

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

foreach ($arr as &$valor) {

$valor = $valor \* 2;

}

// $arr agora é array(2, 4, 6, 8)

// sem um unset($valor), $valor continuará como referência ao último item: $arr[3]

foreach ($arr as $chave => $valor) {

// $arr[3] será atualizado com cada valor de $arr...

echo "{$chave} => {$valor} ";

print\_r($arr);

}

// ...até que, o segundo e último valor é copiado para o último valor

// saída:

// 0 => 2 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 2 )

// 1 => 4 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 4 )

// 2 => 6 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 6 )

// 3 => 6 Array ( [0] => 2, [1] => 4, [2] => 6, [3] => 6 )

?>

É possível iterar o valor de um array constante por referência

<?php

foreach (array(1, 2, 3, 4) as &$valor) {

$valor = $valor \* 2;

}

?>

Nota:

O foreach não possui suporte a habilidade de suprimir mensagens de erro utilizando @.

Mais alguns exemplos para demonstrar o uso:

<?php

/\* Exemplo foreach1: somente valores \*/

$a = array(1, 2, 3, 17);

foreach ($a as $v) {

echo "Current value of \$a: $v.\n";

}

/\* Exemplo foreach 2: valor com acesso manual (apenas ilustrativo) \*/

$a = array(1, 2, 3, 17);

$i = 0; /\* apenas para ilustrar \*/

foreach ($a as $v) {

echo "\$a[$i] => $v.\n";

$i++;

}

/\* Exemplo foreach 3: chave e valor \*/

$a = array(

"one" => 1,

"two" => 2,

"three" => 3,

"seventeen" => 17

);

foreach ($a as $k => $v) {

echo "\$a[$k] => $v.\n";

}

/\* Exemplo foreach 4: arrays multi dimencionais \*/

$a = array();

$a[0][0] = "a";

$a[0][1] = "b";

$a[1][0] = "y";

$a[1][1] = "z";

foreach ($a as $v1) {

foreach ($v1 as $v2) {

echo "$v2\n";

}

}

/\* Exemplo foreach 5: arrays dinâmicos \*/

foreach (array(1, 2, 3, 4, 5) as $v) {

echo "$v\n";

}

?>

Desempacotando arrays aninhados com o construtor list() ¶

(PHP 5 >= 5.5.0, PHP 7, PHP 8)

É possível iterar sobre um array de arrays e desempacotar arrays aninhados em variáveis do laço fornecendo o construtor list() como valor.

Por exemplo:

<?php

$array = [

[1, 2],

[3, 4],

];

foreach ($array as list($a, $b)) {

// $a contém o primeiro elemento do array aninhado,

// e $b contém o segundo elemento.

echo "A: $a; B: $b\n";

}

?>

O exemplo acima produzirá:

A: 1; B: 2

A: 3; B: 4

Pode-se se fornecer menos elementos ao construtor list() aos que existem no array aninhado, neste caso, os valores do array que sobrarem serão ignorados.

<?php

$array = [

[1, 2],

[3, 4],

];

foreach ($array as list($a)) {

// Veja que não existe $b aqui.

echo "$a\n";

}

?>

O exemplo acima produzirá:

1

3

Um aviso será gerado se não houver elementos suficientes no array para o preenchimento do construtor list():

<?php

$array = [

[1, 2],

[3, 4],

];

foreach ($array as list($a, $b, $c)) {

echo "A: $a; B: $b; C: $c\n";

}

?>

O exemplo acima produzirá:

Notice: Undefined offset: 2 in example.php on line 7

A: 1; B: 2; C:

Notice: Undefined offset: 2 in example.php on line 7

A: 3; B: 4; C:

＋add a note

User Contributed Notes 2 notes

up

down

26Okafor Chiagozie ¶1 year ago

An easier way to unpack nested array elements

$array = [

[1, 2],

[3, 4],

];

foreach ($array as [$a, $b]) {

echo "A: $a; B: $b\n";

}

up

down

-2Sanusi Hassan ¶1 year ago

destructure array elements

you can unpac nested array elements using the following

<?php

$array = [

[1, 2],

[3, 4],

];

foreach ($array as $v) {

[$a, $b] = $v;

echo "A: $a; B: $b\n";

}

?>