

Etec "Sales Gomes" – 101 – Tatuí

Programação para WEB

PHP Básico

Prof. Bruno Camargo Ribeiro

Array

Assuntos que serão abordados:

1	Array	2
1.1	Criando um array	2
1.2	Testando um array	3
1.3	Adicionando um valor a frente do array	3
1.4	Adicionando um valor no final do array	4
1.5	Removendo um valor da frente do array	4
1.6	Removendo um valor do final do array	4
1.7	Localizando um elemento	5
1.8	Localizando por chaves de associação	5
1.9	Localizando por valores de array associados	5
1.10	Recuperando as chaves de um array	6
1.11	Recuperando valores de um array	6
1.12	Movendo o ponteiro de um array	6
1.13	Recuperando a chave de um ponteiro	7
1.14	Classificando arrays	7
1.15	Invertendo valores e chaves do array	9
2	Operações com array	10
2.1	Unindo arrays	10
2.2	Combinando arrays	10
2.3	Dividindo um array	11
2.4	Interseção de array	11
2.5	Diferença de array	12
2.6	Misturando elementos de array	12
2.7	Somando elementos de array	12
3	Array com array	13
3.1	Visualizando informação	14

Etec "Sales Gomes" – 101 – Tatuí

1 Array

Tradicionalmente é definido como um grupo de itens que compartilham certas características, como similaridade e tipo de dados. Cada item é distinguido por um identificador conhecido como chave.

1.1 Criando um array

Diferente de outras linguagens o PHP não exige que você defina o tamanho do seu array no momento da criação, nem menos que você declare o array antes de usá-lo. Exemplos: `$estado[0] = "São Paulo";` `$estado[1] = "Rio de Janeiro"`

...

```
echo $estado[0];
```

Valores adicionais podem ser inseridos indicando novos índices.

```
$estado[10] = "Bahia";
```

Caso deseje o índice com um valor numérico e crescente poderá utilizar o seguinte recurso:

```
$estado[] = "São Paulo";
```

```
$estado[] = "Rio de Janeiro";
```

```
$estado[] = "Bahia";
```



```
vetor01.php
1  <?php
2      $estado[] = "Sao Paulo";
3      $estado[] = "Rio de Janeiro";
4      $estado[] = "Bahia";
5
6      echo $estado[0];    // Sao Paulo
7  ?>
```

Alternativa é criar um array associado, mas deve-se levar em consideração que a chave é sempre exigida.

```
$estado["SP"] = "São Paulo";
```

```
$estado["RJ"] = "Rio de Janeiro";
```

```
$estado["BA"] = "Bahia";
```

Etec "Sales Gomes" – 101 – Tatuí

```
vetor02.php
1 <?php
2     $estado["SP"] = "Sao Paulo";
3     $estado["RJ"] = "Rio de Janeiro";
4     $estado["BA"] = "Bahia";
5
6     echo $estado["SP"];    // Sao Paulo
7 ?>
```

Criando um array com limite pré-definido.

```
vetor03.php
1 <?php
2     $estados = array("Sao Paulo", "Rio de Janeiro", "Bahia");
3     echo $estados[0];
4 ?>
```

Criando um array com limite pré-definido, usando associação.

```
vetor04.php
1 <?php
2     $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3     echo $estados["SP"];    // Sao Paulo
4 ?>
```

1.2 Testando um array

Para testar se uma variável é array, podemos utilizar a função **is_array()**.

```
is_array.php
1 <?php
2     $variavel = 10;
3     $vetor = array();
4     if (is_array($variavel)){ echo "variavel = array";}
5     if (is_array($vetor)){ echo "vetor = array";}
6 ?>
```

1.3 Adicionando um valor a frente do array

A função **array_unshift()** adiciona elementos na frente do array.

Etec "Sales Gomes" – 101 – Tatuí

```
array_unshift().php
1 <?php
2     $estados = array("GO", "MG", "RJ", "SP");
3     array_unshift($estados, "AM");
4
5     foreach($estados as $itemvetor){
6         echo "$itemvetor <br>";
7     }
8     ?>
```

1.4 Adicionando um valor no final do array

A função **array_push()** adiciona elementos no final do array.

```
array_push().php
1 <?php
2     $estados = array("GO", "MG", "RJ", "SP");
3     array_push($estados, "TO");
4
5     foreach($estados as $itemvetor){
6         echo "$itemvetor <br>";
7     }
8     ?>
```

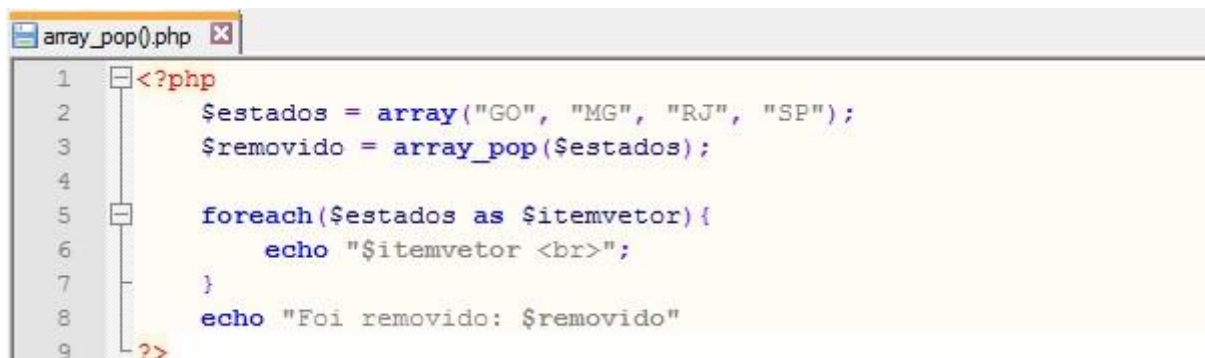
1.5 Removendo um valor da frente do array

A função **array_shift()** remove e retorna o item encontrado na primeira posição do vetor.

```
array_shift().php
1 <?php
2     $estados = array("GO", "MG", "RJ", "SP");
3     $removido = array_shift($estados);
4
5     foreach($estados as $itemvetor){
6         echo "$itemvetor <br>";
7     }
8     echo "Foi removido: $removido"
9     ?>
```

1.6 Removendo um valor do final do array

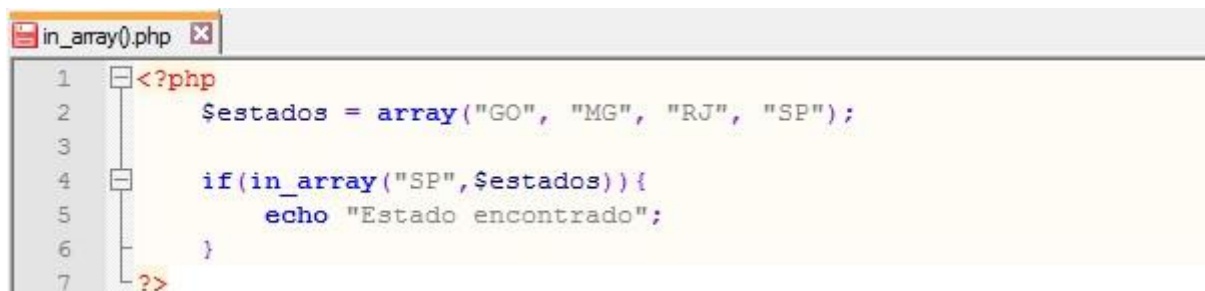
A função **array_pop()** remove e retorna o item encontrado no final do vetor.

Etec "Sales Gomes" – 101 – Tatuí

```
1 <?php
2 $estados = array("GO", "MG", "RJ", "SP");
3 $removido = array_pop($estados);
4
5 foreach($estados as $itemvetor){
6     echo "$itemvetor <br>";
7 }
8 echo "Foi removido: $removido"
9 ?>
```

1.7 Localizando um elemento

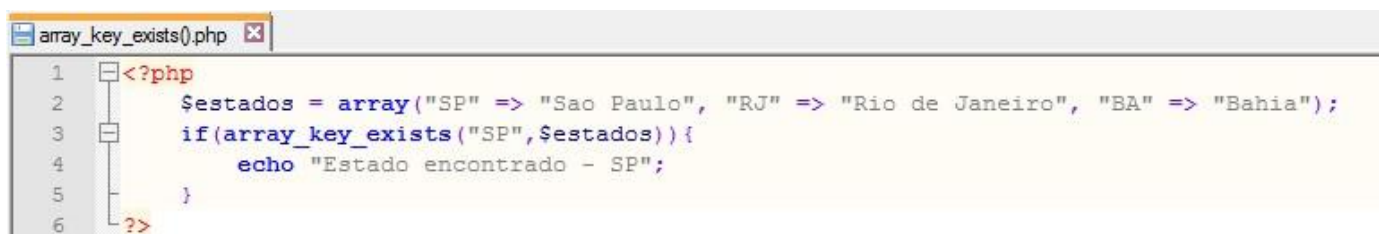
A função **in_array()** busca um elemento em um vetor, retornando *TRUE* ou *FALSE*.



```
1 <?php
2 $estados = array("GO", "MG", "RJ", "SP");
3
4 if(in_array("SP", $estados)){
5     echo "Estado encontrado";
6 }
7 ?>
```

1.8 Localizando por chaves de associação

A função **array_key_exists()** retorna *TRUE* se a chave for localizada.



```
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3 if(array_key_exists("SP", $estados)){
4     echo "Estado encontrado - SP";
5 }
6 ?>
```

1.9 Localizando por valores de array associados

A função **array_search()** busca um valor no array e retorna a chave se for localizado.

Etec "Sales Gomes" – 101 – Tatuí

```
array_search().php
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3 $encontrou = array_search("Bahia", $estados);
4
5 echo "O resultado foi $encontrou ";
6
7 ?>
```

1.10 Recuperando as chaves de um array

A função **array_keys()** retorna um array formado por todas as chaves localizadas de outro array.

```
array_key().php
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4 $chaves = array_keys($estados);
5
6 print_r($chaves);
7
8 ?>
```

1.11 Recuperando valores de um array

A função **array_values()** retorna um array com todos os valores localizados de outro array.

```
array_values().php
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4 $valores = array_values($estados);
5
6 print_r($valores);
7
8 ?>
```

1.12 Movendo o ponteiro de um array

A função **next()** retorna o valor do próximo elemento do array na posição seguinte ao ponteiro.

Etec "Sales Gomes" – 101 – Tatuí

```
next().php
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4 echo next($estados); // Rio de Janeiro
5 echo "<br>";
6 echo next($estados); // Bahia
7
8 ?>
```

Outras funções relacionadas à movimentação do ponteiro do array:

Função **prev()**, retorna o valor anterior da atual posição do ponteiro.

Função **reset()**, volta ao começo do array.

Função **end()**, vai para o final do array.

1.13 Recuperando a chave de um ponteiro

A função **key()** retorna a chave da posição atual do ponteiro.

```
key().php
1 <?php
2 $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4 while($chave = key($estados)){
5     echo "$chave <br>";
6     next($estados);
7 }
8
9 ?>
```

1.14 Classificando arrays

A função **array_reverse()** reverte a ordem os elementos do vetor.

Etec "Sales Gomes" – 101 – Tatuí

```
array_reverse()01.php
1 <?php
2     $estados = array("Sao Paulo", "Rio de Janeiro", "Bahia");
3
4     $outro_estados = array_reverse($estados);
5     print_r($outro_estados);
6
7 ?>
```

Array ([0] => Bahia [1] => Rio de Janeiro [2] => Sao Paulo)

Funciona para chaves associadas.

```
array_reverse()02.php
1 <?php
2     $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4     $outro_estados = array_reverse($estados);
5     print_r($outro_estados);
6
7 ?>
```

Array ([BA] => Bahia [RJ] => Rio de Janeiro [SP] => Sao Paulo)

A função **sort()** classifica um array ordenando do menor para o maior.

```
sort().php
1 <?php
2     $estados = array("SP", "DF", "AM", "PE", "CE", "TO", "PI", "AL");
3
4     sort($estados);
5     print_r($estados);
6
7 ?>
```

Array ([0] => AL [1] => AM [2] => CE [3] => DF [4] => PE [5] => PI [6] => SP [7] => TO)

A função **rsort()** classifica um array ordenando do maior para o menor.

```
rsort().php
1 <?php
2     $estados = array("SP", "DF", "AM", "PE", "CE", "TO", "PI", "AL");
3
4     rsort($estados);
5     print_r($estados);
6
7 ?>
```

Array ([0] => TO [1] => SP [2] => PI [3] => PE [4] => DF [5] => CE [6] => AM [7] => AL)

A função **natsort()** organiza o array de forma natural, exemplo usando **sort()**.

Etec "Sales Gomes" – 101 – Tatuí

```
natsort()-01.php
1 <?php
2     $fotos = array("foto1", "foto2", "foto10", "foto20", "foto3", "foto30", "foto150", "foto120");
3
4     sort($fotos);
5     print_r($fotos);
6
7 ?>
```

Array ([0] => foto1 [1] => foto10 [2] => foto120 [3] => foto150 [4] => foto2 [5] => foto20 [6] => foto3 [7] => foto30)

Exemplo usando **natsort()**.

```
natsort()-02.php
1 <?php
2     $fotos = array("foto1", "foto2", "foto10", "foto20", "foto3", "foto30", "foto150", "foto120");
3
4     natsort($fotos);
5     print_r($fotos);
6
7 ?>
```

Array ([0] => foto1 [1] => foto2 [4] => foto3 [2] => foto10 [3] => foto20 [5] => foto30 [7] => foto120 [6] => foto150)

A função **natcasesort()** oferece outro recurso com relação as letras maiúsculas e minúsculas.

```
natcasesort().php
1 <?php
2     $fotos = array("Foto1", "foto2", "FOTO10", "Foto20", "foto3", "FoTo30", "foto150", "FOTO120");
3
4     natcasesort($fotos);
5     print_r($fotos);
6
7 ?>
```

Array ([0] => Foto1 [1] => foto2 [4] => foto3 [2] => FOTO10 [3] => Foto20 [5] => FoTo30 [7] => FOTO120 [6] => foto150)

1.15 Invertendo valores e chaves do array

A função **array_flip()** inverte a função da chave com seus valores.

Etec "Sales Gomes" – 101 – Tatuí

```
array_flip()-01.php
1 <?php
2     $estados = array("Sao Paulo","Rio de Janeiro","Bahia");
3
4     $outro_estados = array_flip($estados);
5     print_r($outro_estados);
6
7 ?>
```

Array ([Sao Paulo] => 0 [Rio de Janeiro] => 1 [Bahia] => 2)

Funciona para chaves associadas.

```
array_flip()-02.php
1 <?php
2     $estados = array("SP" => "Sao Paulo", "RJ" => "Rio de Janeiro", "BA" => "Bahia");
3
4     $outro_estados = array_flip($estados);
5     print_r($outro_estados);
6
7 ?>
```

Array ([Sao Paulo] => SP [Rio de Janeiro] => RJ [Bahia] => BA)

2 Operações com array

O PHP tem funções que são capazes de realizar algumas operações um pouco mais complexas com arrays, veja algumas delas.

2.1 Unindo arrays

A função **array_merge()** permite unir arrays, retornando um único array.

```
array_merge().php
1 <?php
2     $estados1 = array('AC','AL','AP','AM','BA','CE','DF','ES','GO','MA','MT','MS');
3     $estados2 = array('MG','PA','PB','PR','PE','PI','RJ','RN','RS','RO','RR','SC','SP','SE','TO');
4
5     $br = array_merge($estados1,$estados2 );
6     print_r($br);
7
8 ?>
```

2.2 Combinando arrays

A função **array_combine()** permite criar um novo array realizando uma associação entre chaves e valores.

Etec "Sales Gomes" – 101 – Tatuí

```
array_combine().php
1  <?php
2      $estados1 = array('SP', 'RJ', 'MG');
3      $estados2 = array('Sao Paulo', 'Rio de Janeiro', 'Minas Gerais');
4
5      $br = array_combine($estados1, $estados2);
6      print_r($br);
7
8  ?>
```

Array ([SP] => Sao Paulo [RJ] => Rio de Janeiro [MG] => Minas Gerais)

2.3 Dividindo um array

A função **array_slice()** permite dividir um array a partir de um *offset* inicial e final.

```
array_slice()-01.php
1  <?php
2      $estados = array('AC', 'AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS',
3                      'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO');
4
5      $parte = array_slice($estados, 20);
6      print_r($parte);
7
8  ?>
```

Array ([0] => RS [1] => RO [2] => RR [3] => SC [4] => SP [5] => SE [6] => TO)

Outro exemplo:

```
array_slice()-02.php
1  <?php
2      $estados = array('AC', 'AL', 'AP', 'AM', 'BA', 'CE', 'DF', 'ES', 'GO', 'MA', 'MT', 'MS',
3                      'MG', 'PA', 'PB', 'PR', 'PE', 'PI', 'RJ', 'RN', 'RS', 'RO', 'RR', 'SC', 'SP', 'SE', 'TO');
4
5      $parte = array_slice($estados, 24, 1);
6      print_r($parte);
7
8  ?>
```

Array ([0] => SP)

2.4 Interseção de array

A função **array_intersect()** mantém os valores iguais presente nos arrays, considerando o primeiro array.

Etec "Sales Gomes" – 101 – Tatuí

```
array_intersect().php
1  <?php
2      $estados1 = array('AC', 'AL', 'AP', 'AM');
3      $estados2 = array('AC', 'SP', 'AM', 'MG');
4      $estados3 = array('SP', 'AM', 'AC', 'AL');
5
6      $br = array_intersect($estados1, $estados2, $estados3);
7      print_r($br);
8
9  ?>
```

Array ([0] => AC [3] => AM)

2.5 Diferença de array

A função **array_diff()** retorna os valores localizados no primeiro array que não existe nos demais.

```
array_diff().php
1  <?php
2      $estados1 = array('AC', 'AL', 'AP', 'TO');
3      $estados2 = array('BA', 'SP', 'AM', 'MG');
4      $estados3 = array('SP', 'AM', 'AC', 'AL');
5
6      $br = array_diff($estados1, $estados2, $estados3);
7      print_r($br);
8
9  ?>
```

Array ([2] => AP [3] => TO)

2.6 Misturando elementos de array

A função **shuffle()** reordena um array aleatoriamente.

```
shuffle().php
1  <?php
2      $numeros = array(1,2,3,4,5,6,7,8,9,10);
3
4      shuffle($numeros);
5      print_r($numeros);
6
7  ?>
```

Array ([0] => 10 [1] => 4 [2] => 1 [3] => 8 [4] => 3 [5] => 5 [6] => 6 [7] => 2 [8] => 7 [9] => 9)

2.7 Somando elementos de array

A função **array_sum()** soma todos os valores do array, caso existe algum elemento do tipo string, o mesmo será desconsiderado.

Etec "Sales Gomes" – 101 – Tatuí

```
array_sum() .php
1  <?php
2      $numeros = array(1,2,3,4,5,6,7,8,9,10);
3
4      $total = array_sum($numeros);
5      print_r($total);
6
7  ?>
```

Resposta: 55

3 Array com array

Uma opção é realizar a inserção de arrays dentro de outro array (matriz).

```
array-01.php
1  <?php
2      $vetor1 = array(1, 4, 7);
3      $vetor2 = array(2, 5, 8);
4      $vetor3 = array(3, 7, 9);
5
6      $teclado = array($vetor1, $vetor2, $vetor3);
7
8      print_r($teclado);
9
10 ?>
```

Array ([0] => Array ([0] => 1 [1] => 4 [2] => 7) [1] => Array ([0] => 2 [1] => 5 [2] => 8) [2] => Array ([0] => 3 [1] => 7 [2] => 9))

Alternativa para a construção do mesmo vetor.

```
array-02.php
1  <?php
2      $teclado = array(array(1, 4, 7), array(2, 5, 8), array(3, 7, 9));
3      print_r($teclado);
4  ?>
```

Valores podem ser carregados diretamente no array composto, basta indicar as posições (linha x coluna);

```
$teclado[1][1] = 5;
```


Etec "Sales Gomes" – 101 – Tatuí

```
array-elementos.php
1  <?php
2      $teclado[0][0] = 1;
3      $teclado[0][1] = 4;
4      $teclado[0][2] = 7;
5
6      $teclado[1][0] = 2;
7      $teclado[1][1] = 5;
8      $teclado[1][2] = 8;
9
10     print_r($teclado);
11
12     ?>
```

Array ([0] => Array ([0] => 1 [1] => 4 [2] => 7) [1] => Array ([0] => 2 [1] => 5 [2] => 8))

3.1 Visualizando informação

Para visualizar informações em estruturas compostas informe os índices para cada um dos arrays (linha x coluna).

```
array-03.php
1  <?php
2      $teclado = array(array(1, 4, 7), array(2, 5, 8), array(3, 7, 9));
3
4      echo $teclado[0][1];
5
6      ?>
```