UniSENAI

CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ARTIGO SOBRE FUNÇÕES E ARRAYS EM PHP

Pedro Henrique de Souza

Danillo

Mauricio

Gustavo

Funções e Arrays php

JOINVILLE
2025

O que são arrays

Arrays são estruturas capazes de armazenar vários valores em uma única variável, organizando-os em fileira por ordem de inserção.

Como funcionam as arrays (e Como criar, acessar, percorrer e modificar arrays)

Em PHP, existem três tipos principais de arrays, arrays indexadoras, arrays associativas e as arrays multidimensionais, também conhecidas como matrizes. Para criar arrays, pode ser utilizado a função **array( ),** logo a frente do nome da variável (**variavel = array("valor1", "valor2", "valor3")**), ou apenas utilizando colchetes **[ ].**

Array indexadoras

As arrays indexadoras são estruturadas de forma que cada elemento ocupa uma posição na array, iniciando do zero, o que a torna útil para armazenar dados em que a ordem de disposição é importante.

Array associativo

As arrays associativos são semelhantes, porém, cada valor inserido na array é associado a uma **chave** nomeada, como se cada posição da array tivesse um nome específico, o que é útil para armazenar valores como notas de alunos, dados de cadastros, entre outros.

Array associativo utilizando a função array()

```php
<?php
// Criando um array associativo de notas
$notas = array(
    "João" => 8.5,
    "Maria" => 9.0,
    "Pedro" => 7.2
);

// Acessando o valor da chave "Maria"
echo "Nota da Maria: " . $notas["Maria"]; // Saída: 9.0

// Exibindo todas as notas
foreach ($notas as $aluno => $nota) {
    echo "<br>$aluno tirou nota $nota";
}
?>
```

Utilizando colchetes [ ]

```php
<?php
// Criando um array associativo de notas usando sintaxe moderna
$notas = [
    "João" => 8.5,
    "Maria" => 9.0,
    "Pedro" => 7.2
];

// Acessando o valor da chave "Maria"
echo "Nota da Maria: " . $notas["Maria"]; // Saída: 9.0

// Exibindo todas as notas
foreach ($notas as $aluno => $nota) {
    echo "<br>$aluno tirou nota $nota";
}
?>
```

Array Multidimensionais

Arrays multidimensionais, também conhecidas como matrizes, são arrays dentro de arrays formando uma espécie de tabela em que a array principal é associada às linhas de uma matriz e as arrays dentro dela são associadas às colunas. Esse tipo de array é útil para criar tabelas de dados.

Array multidimensional utilizando a função array()

teste.php > ...

```php
<?php
// Criando um array multidimensional de produtos
$produtos = array(
    array("nome" => "Mouse", "preco" => 50, "estoque" => 100),
    array("nome" => "Teclado", "preco" => 120, "estoque" => 50),
    array("nome" => "Monitor", "preco" => 800, "estoque" => 20)
);

// Acessando um valor específico
echo "O preço do " . $produtos[1]["nome"] . " é R$" . $produtos[1]["preco"];
// Saída: O preço do Teclado é R$120

// Percorrendo todo o array
foreach ($produtos as $produto) {
    echo "<br>Produto: " . $produto["nome"] . " - R$" . $produto["preco"];
}
?>
```

Array multidimensional utilizando colchetes [ ]

teste.php > ...

```php
<?php
// Criando um array multidimensional de produtos usando sintaxe moderna
$produtos = [
    ["nome" => "Mouse", "preco" => 50, "estoque" => 100],
    ["nome" => "Teclado", "preco" => 120, "estoque" => 50],
    ["nome" => "Monitor", "preco" => 800, "estoque" => 20]
];

// Acessando um valor específico
echo "O preço do " . $produtos[1]["nome"] . " é R$" . $produtos[1]["preco"];
// Saída: O preço do Teclado é R$120

// Percorrendo todo o array
foreach ($produtos as $produto) {
    echo "<br>Produto: " . $produto["nome"] . " - R$" . $produto["preco"];
}
?>
```

Funções para manipular dados em uma array

Existem algumas funções que facilitam a manipulação de valores dentro de uma array, permitindo contagem automática de valores, ordenar os valores automáticamente, adicionar mais valores e até unir duas arrays em uma só.

- **foreach()**

Se trata de uma ferramenta do PHP que tem como função percorrer todo o array, item por item, sem você precisar se preocupar com índices e chaves. exemplo:

```php
<?php
// Criando um array com algumas frutas
$frutas = array("maçã", "banana", "laranja", "uva");

// Usando foreach para mostrar cada fruta na tela
foreach ($frutas as $fruta) {
    echo "Fruta: " . $fruta . "<br>";
}
?>
```

A linha **$frutas = array("maçã", "banana", "laranja", "uva");** Serve para criar e inserir os valores dentro da array, utilizando a função array().

Na linha **foreach ($frutas as $fruta) { ... },** a função foreach serve para criar uma variável temporária chamada **$fruta** para cada item dentro do array $frutas, em seguida, ele exibe esse valor através da função **echo.**

Dentro do bloco, é utilizado **echo** para imprimir o texto **"Fruta: "** seguido do nome da fruta que está dentro da array, e depois uma quebra de linha **(
)** para mostrar cada fruta em uma linha nova. A função foreach fará isso até que não tenha mais nenhum valor dentro do vetor $frutas.

Resultado exposto:

Fruta: maçã
Fruta: banana
Fruta: laranja
Fruta: uva

- **count()**

Esta função tem como finalidade, contar quantos elementos foram inseridos dentro do array, ou seja, saber o tamanho do array, podendo ser usado quando você quer controlar loops, validar dados, ou só para mostrar informações.

Exemplo:

```
php

<?php
$frutas = array("maçã", "banana", "laranja", "uva");
$quantidade = count($frutas);
echo "Temos " . $quantidade . " frutas na lista.";
?>
```

A linha **$frutas = array("maçã", "banana", "laranja", "uva"),** Criando um array chamado de **$frutas** e inserimos os valores: maçã, banana, laranja e uva.

A linha **$quantidade = count($frutas);** usa-se a função **count ( );** para contar quantos elementos temos dentro da variável **$frutas** e armazenar na variável $quantidade.

linha : echo "Temos " . $quantidade . " frutas na lista.";

echo é comumente utilizado para apresentar valores em php, neste caso apresenta a variável $quantidade:

Resultado exposto:

Temos 4 frutas na lista.

- **array_push()**

Esta função tem por finalidade adicionar mais elementos ao final do array.

Exemplo :

```
php

<?php
$frutas = array("maçã", "banana");
array_push($frutas, "laranja", "uva");

foreach ($frutas as $item) {
    echo $item . "<br>";
}
?>
```

linha: $frutas = array("maçã", "banana");

Declara a variável $frutas sendo um array contendo dois elementos, maçã e banana.

linha: array_push($frutas, "laranja", "uva");

função array_push irá adicionar a variável $frutas os itens (laranja e uva).

linha: foreach ($frutas as $item) {

Com a função foreach **(loop),** será percorrido o array **$frutas**, este loop usa a Variável $item para representar cada valor do array enquanto o **foreach** passa por eles (no seu exemplo, **$item**). Ao executar o loop, a ordem seria: Na primeira volta**,** **$item** é **"maçã".** Na segunda volta**,** **$item** é **"banana".** Na terceira**,** **$item** é **"laranja".** e na quarta**,** **$item** é **"uva".**

**linha: echo $item . "
";**

Aqui será apresentado o valor direto:

Resultado exposto:

maçã

banana

laranja

uva

- **array_merge()**

A função array_marge() tem por finalidade unidirecionar dois ou mais arrays em um só, unindo todos os elementos em uma única lista. Comumente utilizada para combinar dados de diferentes fontes ou listas em PHP.

com isso temos utilidades como:

- Unir listas de produtos, nomes, números, etc;
- Juntar resultados de diferentes consultas;
- Criar um único array com dados vindos de múltiplas partes do código.

Exemplo:

```php
<?php
$frutas1 = array("maçã", "banana");
$frutas2 = array("laranja", "uva");
$frutasCombinadas = array_merge($frutas1, $frutas2);

echo "Lista completa de frutas:<br>";

foreach ($frutasCombinadas as $fruta) {
    echo $fruta . "<br>";
}
?>
```

linha : $frutas1 = array("maçã", "banana");

Cria o array $frutas1 como os valores maçã e banana.

linha : $frutas2 = array("laranja", "uva");

cria o array com os valores de laranja e uva.

linha : $frutasCombinadas = array_merge($frutas1, $frutas2);

A Função **array_merge($frutas1, $frutas2)** junta os dois arrays em um só, armazenando dentro da variável $frutasCombinadas.

$frutasCombinadas assume o valor = **Array ([0] => maçã [1] => banana [2] => laranja [3] => uva).**

linha : echo "Lista completa de frutas: ";

Exibe o texto : Lista completa de frutas:

```
linha : foreach ($frutasCombinadas as $fruta) {
    echo $fruta . "<br>";
}
```

O **foreach** percorre cada elemento do array e exibe uma fruta por linha.

Resultado exposto:

Lista completa de frutas: maçã, banana, laranja, uva

- **sort():**

É uma função que organiza os elementos do array em ordem crescente, Tem uma boa finalidade quando há a necessidade de exibir ou processar os dados já ordenados, ou seja, do menor para o maior ou em caso de strings, usa ordem alfabética (A-Z).

Exemplo:

```php
<?php
$frutas = array("laranja", "maçã", "banana", "uva");
sort($frutas);

echo "Frutas em ordem alfabética:<br>";

foreach ($frutas as $fruta) {
    echo $fruta . "<br>";
}
?>
```

linha : $frutas = array("laranja", "maçã", "banana", "uva");

Cria o array $frutas com quatro elementos desordenados.

linha : sort($frutas);

A função sort($frutas) organiza os valores do array em ordem crescente.

**linha : echo "Frutas em ordem alfabética:
";**

Exibe o texto Frutas em ordem alfabética : seguindo de uma quebra de linha
.

linha : foreach ($frutas as $fruta) {

**echo $fruta . "
";**

}

O **foreach** percorre cada elemento do array e exibe uma fruta por linha.

Resultado exposto:

Frutas em ordem alfabética:

banana

laranja

maçã

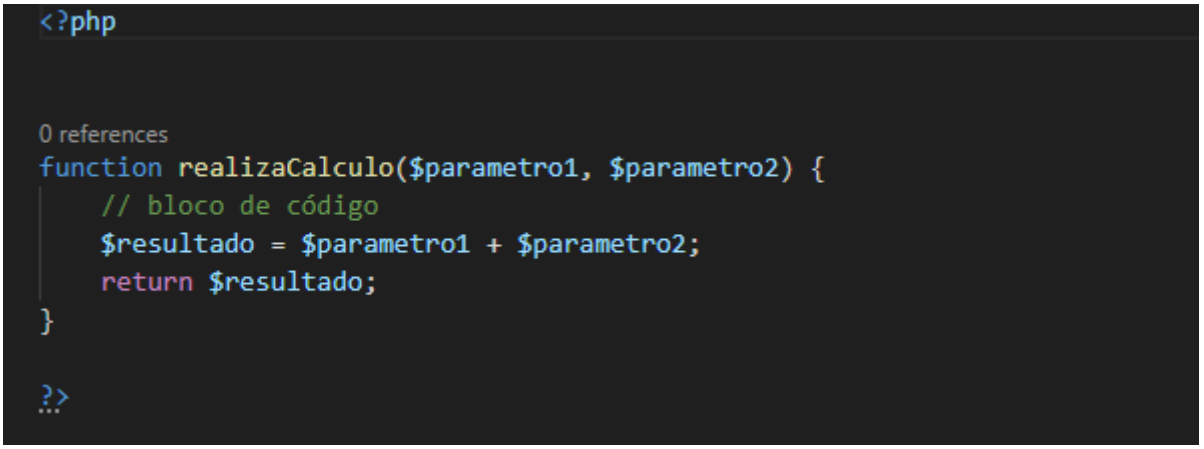
uva

O que são funções

Funções são blocos de código criados para executar uma tarefa específica dentro de um programa. Elas permitem reutilizar instruções, tornando o código mais organizado, legível e fácil de manter. Em vez de repetir o mesmo trecho de código diversas vezes, basta criar uma função e chamá-la sempre que necessário.

Estrutura básica de uma função

Em PHP, uma função é definida utilizando a palavra-chave function, seguida do nome da função, parênteses **()** e um bloco de código entre chaves **{}**. Dentro dos parênteses, podem ser definidos parâmetros, que são valores enviados para a função. A estrutura básica é:



Para executar uma função, basta chamá-la pelo nome, seguida de parênteses, podendo passar valores se necessário, inserindo-os dentro dos parênteses. Esses valores serão associados aos parâmetros inseridos ao criar a função.

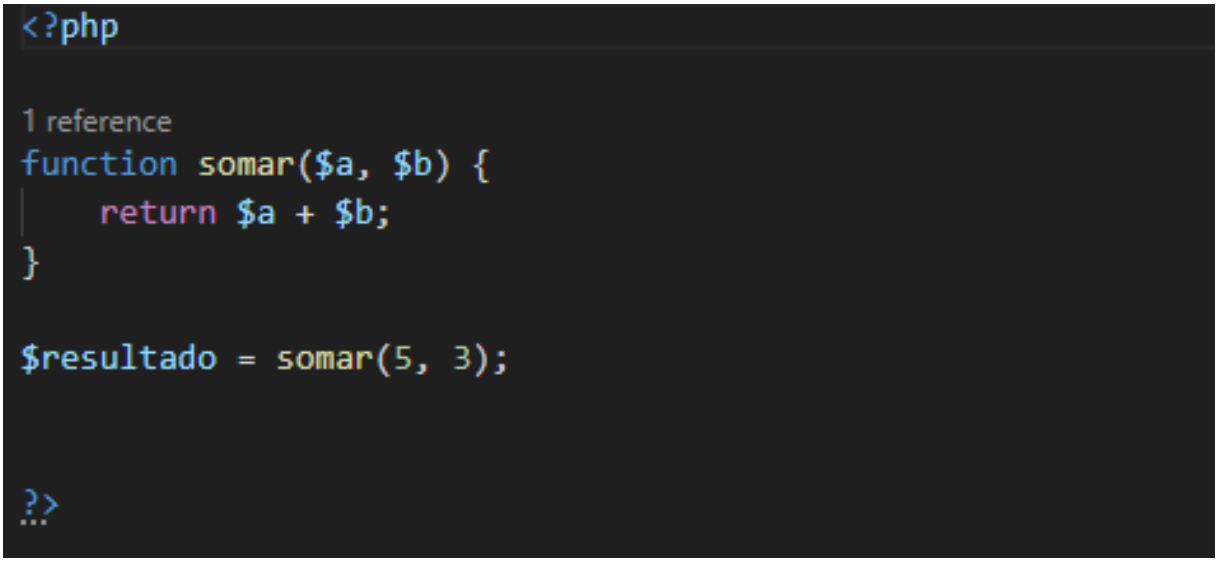
Diferença entre funções nativas e funções definidas pelo usuário

As funções nativas são aquelas que já vêm prontas no PHP, desenvolvidas pela própria linguagem. Elas realizam tarefas comuns e economizam tempo de desenvolvimento. Algumas das funções nativas são a **array_push()** para adicionar elementos a um array as funções de **session**, que servem para criar, manipular, excluir uma sessão ou as variáveis contidas nela, entre outras.

Já as funções definidas pelo usuário são criadas pelo próprio programador, com o objetivo de realizar tarefas específicas de um projeto. Elas permitem personalizar o comportamento do código e adaptá-lo às necessidades do sistema.

Parâmetros e retorno

As funções podem receber parâmetros, que são valores passados a elas no momento da chamada. Esses valores são utilizados dentro da função para realizar cálculos ou manipulações. Além disso, uma função pode retornar um valor usando a instrução return. Esse retorno pode ser armazenado em uma variável ou usado diretamente em outra parte do código.



Boas práticas no uso de funções

- Utilize **nomes descritivos** para as funções, indicando claramente sua finalidade, por exemplo, calcularSoma, multiplicar, exibir etc...

- Evite criar funções muito longas ou que façam várias coisas diferentes.

- Prefira **funções reutilizáveis**, que possam ser usadas em outras partes do código.

- Documente suas funções com comentários explicando os parâmetros e o retorno.

- Use **retornos consistentes**, evitando misturar tipos diferentes de valores (por exemplo, retornar às vezes um número e outras vezes uma string).

Fontes

PHP Manual. **array_merge**. Disponível em: https://www.php.net/manual/en/function.array-merge.php. Acesso em: 19 out. 2025.

PHP Manual. **array_push**. Disponível em: https://www.php.net/manual/en/function.array-push.php. Acesso em: 19 out. 2025.

PHP Manual. **count**. Disponível em: https://www.php.net/manual/en/function.count.php. Acesso em: 19 out. 2025.

PHP Manual. **foreach**. Disponível em: https://www.php.net/manual/en/control-structures.foreach.php. Acesso em: 19 out. 2025.

PHP Manual. **sort**. Disponível em: https://www.php.net/manual/en/function.sort.php. Acesso em: 19 out. 2025.

https://www.php.net/manual/pt_BR/indexes.functions.php