PHP Array: O que é, como utilizar e as principais funções

Neste artigo vamos desvendar sobre o PHP Array, o que são, como funciona, e como utilizar. Inicialmente, recomendamos a leitura do nosso artigo sobre o que é PHP, onde você poderá aprender mais sobre a linguagem PHP, como funciona e ainda como instalar um servidor local para poder estar fazendo os testes na sua própria máquina.

Portanto, veremos nesse artigo os seguintes tópicos:

O Que é a PHP array?

No PHP, um array é, na verdade, um mapa ordenado. Ou seja, é um tipo que relaciona valores a chaves. Portanto, é uma lista de valores que serão armazenados na memória. O PHP array é um tipo de dado, assim como **integer**, **float**, **string** ou **boolean**. Porém, ele pode armazenar mais de um valor, relacionando-os a suas chaves.

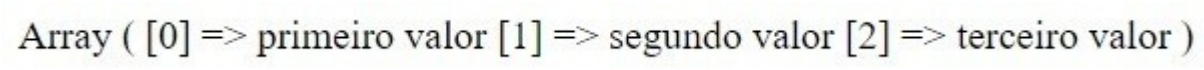
De uma forma geral, podemos dizer que a PHP array é uma variável do php, porém com a possibilidade de dentro dela possuirmos diversos valores. Academicamente falando, o PHP array é equivalente ao conceito de **vetor**. Também considerando uma array do PHP, existem as PHP arrays multidimensionais (também conhecidas como PHP Multi Arrays), que são arrays compostas por outras arrays. Portanto, outro conceito acadêmico que se assemelha a array do PHP é o conceito de **matriz**.

Como Criar uma Array?

Inicialmente, para criar uma PHP array, primeiramente você vai criar uma variável. Após isso, deve-se atribuir para ela um par de chaves[ ]. Por fim, dentro dessas chaves, você armazena os valores que quiser, separados por vírgula. Veja o exemplo abaixo:

```
<?php
$arr=["primeiro valor", "segundo valor", "terceiro valor"];
print_r($arr);
?>
```

Através da função print_r conseguimos imprimir nossa array no navegador, tendo como resultado conforme a imagem abaixo:



Outra forma de declarar uma array é através da função array(), onde incluímos os valores dentro dos parenteses, separados por vírgula. Vejamos o mesmo exemplo anterior só que com essa sintaxe:

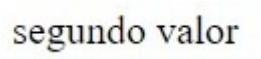
```
<?php
$arr=array("primeiro valor", "segundo valor", "terceiro valor");
print_r($arr);
?>
```

Utilizando as chaves da array

Nos exemplos que utilizamos acima, você pode observar que os valores da PHP array são armazenados dentro de chaves. Dessa forma podemos utilizar um valor específico da array. Observe ainda, que quando as chaves são enumeradas automaticamente, a array inicia-se pela chave [0], e assim por diante. Portanto, a primeira chave é a chave [0] e não a chave [1]. Por exemplo, ainda utilizando a array dos exemplos acima, caso eu queira pegar o segunda valor, ou seja, o valor da chave [1], basta eu chamar pela $arr[1]. Vejamos no exemplo abaixo

```
<?php
$arr=array("primeiro valor", "segundo valor", "terceiro valor");
echo $arr[1];
?>
```

Dessa forma, o resultado no nosso navegador será conforme a imagem abaixo:



Outra coisa que podemos estar fazendo é criar chaves personalizadas para nossa array. Para isso, basta utilizar “=>“, onde a esquerda teremos o nome da chave e a direita o valor da chave. Vejamos o exemplo a seguir:

```
<?php
$arr=array( "nome"=>"Rafael","sobrenome"=>"Marques", "idade"=>"25");
print_r($arr);

echo "<br/><br/>";
echo "Nome: <b>".$arr["nome"]."</b><br/>";
echo "Sobrenome: <b>".$arr["sobrenome"]."</b><br/>";
echo "Idade: <b>".$arr["idade"]."</b>";
?>
```

Com o exemplo acima, vemos por completo como utilizar as chaves personalizadas e ainda como chamar as mesmas através de suas chaves. Portanto, o resultado no navegador será como na imagem abaixo:

PHP Array Multidimensional (ou PHP Multi Array)

As PHP arrays são estruturas heterogêneas que permitem que você salve múltiplos dados de tipos diferentes em um mesmo lugar. Dessa forma, é permitido incluir uma ou mais arrays, dentro de uma mesma PHP array. Portanto, obtemos assim a forma de estrutura de uma matriz. Veja então o exemplo abaixo:

```
<?php
$arr=array(
array("Primeiro valor","Segundo valor"),
array("Terceiro valor","Quarto valor")
);
print_r($arr);
echo "<br/>";
print_r($arr[0]);
echo "<br/>";
print_r($arr[1]);
?>
```

Dessa forma, teremos como resultado impresso no nosso navegador conforme imagem abaixo:

Caso que queira acessar valores específicos dentro das PHP arrays internas, podemos chamar utilizando a estrutura $arr[][]. Vejamos então o exemplo a seguir:

```
<?php
$arr=array(
array("Primeiro valor","Segundo valor"),
array("Terceiro valor","Quarto valor")
);

print_r($arr[0][0]);
echo "<br/>";
print_r($arr[0][1]);
echo "<br/>";
print_r($arr[1][0]);
echo "<br/>";
print_r($arr[1][1]);
?>
```

Dessa forma, teremos em nosso navegador conforme a imagem abaixo:

Contando os elementos de uma PHP Array

Principalmente quando utilizamos chaves personalizadas, as vezes se torna difícil identificar a grosso modo quantos elementos uma PHP Array possui. Portanto, para isso existe a função count() que faz essa verificação e retorna para você o valor de elementos contidos dentro dela.

Vejamos então o exemplo abaixo:

```
<?php
$numeros = [1,2,700,13,6,78,100,212,15,2,3,1000,412,6];
echo count($numeros);
?>
```

Observe que dessa forma, teremos como retorno em nosso navegador o valor 14. Ou seja, há um total de 14 elementos dentro da nossa PHP array $numeros.

Adicionando um valor a array dinamicamente

Além das formas anteriores de declarar uma array, também podemos criar uma array e adicionar seu valores de uma maneira dinâmica. Para isso, basta estar chamando o nome da array com [] ao final, e automaticamente, o valor será incrementado a uma última chave, ao fim da array. Vejamos o exemplo abaixo para melhor entendimento:

```
<?php
$arr = array();
$arr[] = 'azul';
$arr[] = 'amarelo';
$arr[] = 'vermelho';
$arr[] = 'rosa';

print_r($arr)
?>
```

Dessa forma, teremos como resultado conforme a imagem abaixo:

Também é possível adicionar um valor a uma PHP array já existente, da mesma forma. Lembre-se sempre que dessa forma, o valor sempre será adicionado a uma chave posterior a última chave já existente. Vamos então criar uma array com 3 valores a ela. Posteriormente, acrescentaremos valores de forma dinâmica, e verificaremos então o resultado.

```
<?php
        $arr=array("primeiro valor", "segundo valor", "terceiro valor");
        echo "valor incial da minha array é:<br>";
        print_r($arr);
        echo "<br/><br/>";

        $arr[] = 'quarto valor';
```

```
        $arr[] = 'quinto valor';

        echo "Agora a nossa array é:<br/>";
        print_r($arr);
        echo "<br/>";
?>
```

E dessa forma, teremos então como resultado conforme imagem abaixo:

Adicionando ou removendo um valor na PHP array

Através de algumas funções pode se acrescentar valores dentro de uma array. Podemos também estar removendo valores da array já existente. Portanto, veremos nesse tópico algumas maneiras para poder realizar essas funções:

- 1. Adicionar um Novo Elemento no início do Array
- 2. Adicionar um Novo Elemento no final do Array
- 3. Remover o Primeiro Elemento de um Array
- 4. Remover o Último Elemento do Array

1. Adicionar um Novo Elemento no Inicio do Array

É possível adicionar um novo elemento ao início de uma PHP array com a função **array_unshift()**. Vejamos então o exemplo abaixo:

```
<?php
$frutas=array('maçã','banana','limão');
array_unshift($frutas, 'abacaxi');
print_r($frutas);
?>
```

Observe que dessa forma, teremos como resultado a Array ([0] => abacaxi [1] => maçã [2] => banana [3] => limão). O elemento “abacaxi” foi adicionado ao índice [0], e os demais elementos tiveram suas chaves alteradas para o próximo valor.

2. Adicionar um Novo Elemento no Final do Array

Assim como já vimos anteriormente, podemos estar acrescentando de forma dinâmica, apenas chamando a array com [] e automaticamente ela irá ser acrescentada ao final da array. Porém, outra forma de estar realizando isso, é através da função **array_push()**. Observe o exemplo abaixo utilizando a mesma array do exemplo anterior:

```
<?php
$frutas=array('maçã','banana','limão');
array_push($frutas, 'abacaxi');
print_r($frutas);
?>
```

Dessa forma, teremos como resultado a Array ([0] => maçã [1] => banana [2] => limão [3] => abacaxi), ou seja, mantemos todos os elementos da PHP array em suas chaves originais, e acrescentamos uma chave posteriormente a última já existente, contendo o valor do nosso novo elemento.

3. Remover o Primeiro Elemento de um Array

Muitas vezes, precisamos estar removendo um elemento de dentro de nossa array. Podemos estar removendo o primeiro valor de um array através da função **array_shift()**. Dessa forma, vejamos o exemplo abaixo:

```
<?php
$frutas=array('maçã','banana','limão','abacaxi');
array_shift($frutas);
print_r($frutas);
?>
```

Observe que teremos como resultado então a Array ([0] => banana [1] => limão [2] => abacaxi). Portanto, perceba que o primeiro valor da array inicial, ou seja, “maçã” foi removido. Dessa forma, todas os demais elementos permanecem na array, porém alteram suas chaves para um anterior a de origem.

4. Remover o Último Elemento de um Array

Ao contrário da função **array_shift()** apresentado anteriormente, também podemos estar removendo o último elemento de uma array. Para isto, basta estar utilizando a função **array_pop()**. Vejamos então o mesmo exemplo acima, só que agora removendo o último elemento da PHP Array:

```
<?php
$frutas=array('maçã','banana','limão','abacaxi');
array_pop($frutas);
print_r($frutas);
?>
```

Dessa forma, teremos como resultado a Array ([0] => maçã [1] => banana [2] => limão). Ou seja, continuamos com a nossa array original, com os elementos compostos na mesmas chaves, porém com a última chave e seu valor removidas da php array.

Ordenando uma PHP Array

Os elementos contidos na PHP array podem ser ordenados, classificados em ordem alfabética ou numérica e de forma crescente, ou decrescente. Para isso, utilizamos as funções envolvendo **sort**. Portanto, podemos organizar as arrays tanto através da ordem crescente ou decrescente de seus valores, como também através da ordem crescente ou decrescente de suas chaves.

Ordenando os elementos de forma crescente

Podemos ordenar os elementos de forma crescente apenas usando a função **sort()**. Dessa forma, podemos utilizar tanto para organizar de forma crescente alfabética, como numérica.

Vejamos no exemplo abaixo um PHP Array com valores alfabéticos. Iremos então ordenar de forma crescente alfabeticamente.

```
<?php
```

```
$carros = array("Volvo", "BMW", "Toyota");
sort($carros);
print_r($carros);
?>
```

Observe que dessa forma, obteremos então como resultado a Array ([0] => BMW [1] => Toyota [2] => Volvo).

Vamos agora fazer a mesma coisa, mas para uma PHP array contendo valores numéricos:

```
<?php
$numeros = array(3, 6, 2, 52, 11);
sort($numeros);
print_r($numeros);
?>
```

Observe que dessa forma, obteremos então como resultado a Array ([0] => 2 [1] => 3 [2] => 6 [3] => 11 [4] => 52).

Ordenando os elementos de forma decrescente

Ao contrário da função **sort()**, podemos organizar nossa PHP array através de seus valores em ordem decrescente. Para isso basta utilizar a função **rsort()**. Vejamos então os mesmos exemplos das PHP arrays utilizadas anteriormente, só que dessa vez, usaremos a função **rsort().**

Iremos então ordenar de forma decrescente alfabeticamente:

```
<?php
$carros = array("Volvo", "BMW", "Toyota");
rsort($carros);
print_r($carros);
?>
```

Dessa forma, obteremos então como resultado a Array ([0] => Volvo [1] => Toyota [2] => BMW).

Agora utilizando a array numérica, vamos ordená-la de forma decrescente:

```
<?php
$numeros = array(3, 6, 2, 52, 11);
rsort($numeros);
print_r($numeros);
?>
```

Portanto, teremos como resultado a Array ([0] => 52 [1] => 11 [2] => 6 [3] => 3 [4] => 2).

Ordenando uma array com chaves personalizadas

Podemos ordenar de 4 formas uma array com chaves personalizadas:

- 1- Através do valor de forma crescente;
- 2- Através do valor de forma decrescente;
- 3- Através da Chave de forma crescente;
- 4- Através da Chave de forma decrescente.
- 5- Remover Elementos Repetidos de um Array

Vamos então estudar esses 4 métodos de forma resumida.

1- Ordenando uma array com chaves personalizadas através do valor de forma crescente

O exemplo a seguir classifica uma PHP arrays em ordem **crescente**, de acordo com o **valor**. Para isso usamos a função **asort()**.

```
<?php
$idade = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($idade);
print_r($idade);
?>
```

Observe então, que o resultado levará em consideração a ordem alfabética dos valores, e não das chaves. Portanto, teremos como resultado a Array ([Peter] => 35 [Ben] => 37 [Joe] => 43)

2- Ordenando uma array com chaves personalizadas através do valor de forma decrescente

O exemplo a seguir classifica uma PHP arrays em ordem **decrescente**, de acordo com o **valor**. Para isso usamos a função a **arsort()**.

```
<?php
$idade = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($idade);
print_r($idade);
?>
```

Dessa forma, teremos como resultado a Array ([Joe] => 43 [Ben] => 37 [Peter] => 35).

3- Ordenando uma array com chaves personalizadas através da Chave de forma crescente

O exemplo a seguir classifica uma PHP arrays em ordem **crescente**, de acordo com a **chave**. Para isso usamos a função **ksort()**.

```
<?php
$idade = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($idade);
print_r($idade);
?>
```

Observe então, que dessa vez, os valores não foram considerados para ordenar a php array, mas sim as suas chaves. Portanto obtemos como resultado a Array ([Ben] => 37 [Joe] => 43 [Peter] => 35)

4- Ordenando uma array com chaves personalizadas através da Chave de forma decrescente

O exemplo a seguir classifica uma PHP arrays em ordem **decrescente**, de acordo com a **chave**. Para isso usamos a função a **krsort()**.

```
<?php
$idade = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($idade);
print_r($idade);
?>
```

Portanto, teremos como resultado a Array ([Peter] => 35 [Joe] => 43 [Ben] => 37).

5- Remover Elementos Repetidos de um Array

Às vezes, é comum que nossa PHP Array possua valores repetidos, principalmente se ela for muito grande. Portanto, caso você queira filtrar os elementos que se repetem, ou seja, que possuem o

mesmo valor, basta utilizar a função **array_unique()**. Dessa forma, teremos como resultado uma nova array onde todos os elementos repetidos serão removidos, permanecendo na ordem apenas o que apareceu na primeira chave de mesmo valores. Vejamos então o exemplo abaixo contendo 7 itens, onde alguns se repetem.

```
<?php
$frutas=array('maçã','banana','limão','banana','abacaxi','banana','limão');
$frutas=array_unique($frutas);
print_r($frutas);
?>
```

Observe que inicialmente temos os valores da Array ([0] => maçã [1] => banana [2] => limão [3] => banana [4] => abacaxi [5] => banana [6] => limão). Porém, após utilizarmos a função array_unique(), passamos então a ter a Array ([0] => maçã [1] => banana [2] => limão [4] => abacaxi), removendo assim todos os itens repetidos.

Combinando PHP Arrays

Muitas vezes necessitamos combinar duas ou mais arrays e transforma-las em uma só. Ou seja, pegar todos os elementos de uma array e combina com outra array diferente. Para isso, podemos utilizar a função **array_merge()** para estarmos combinando as arrays em uma única php array. Vejamos o exemplo a seguir:

```
<?php
$frutas=array('maçã','banana','limão');
$legumes=array('batata','cenoura');
$compras=array_merge($frutas,$legumes);
print_r($compras);
?>
```

Observe então que ao final, criamos uma array, $compras, onde realizamos a combinação da array $frutas com a array $legumes. Portanto, nosso resultado será Array ([0] => maçã [1] => banana [2] => limão [3] => batata [4] => cenoura). Podemos estar fazendo isso com duas ou mais arrays.

Filtrando Elementos de um Array

Caso seja necessário utilizar um filtro para pegar valores específico de uma array, podemos estar e utilizando a função array_filter(). Para isso, inicialmente é necessário criar uma função, que será utilizada para realizar o filtro. Posteriormente, chamamos então pela sintaxe:

```
array_filter($array,'função')
```

Vamos criar uma função que irá filtrar nossa PHP array, para que sejam considerados apenas os valores maiores ou iguais a 15. Posteriormente, utilizaremos a função array_filter() e realizaremos o filtro. Vejamos então o exemplo abaixo:

```
<?php
function filtro($value)
{
    return $value >= 15;
}
$numeros = [1,2,700,13,6,78,100,212,15,2,3,1000,412,6];
$filtrado = array_filter($numeros, 'filtro');
print_r($filtrado);
```

```
?>
```

Observe então que após aplicar o filtro, teremos então como resultado a Array ([2] => 700 [5] => 78 [6] => 100 [7] => 212 [8] => 15 [11] => 1000 [12] => 412).

Caso queira aprender sobre como criar funções e outros assuntos, acesse nosso curso introdutório de php.

Iterando Arrays

Na computação, ou melhor, na programação, o termo Iteração significa a repetição de uma ou mais ações. Cada iteração refere-se a apenas uma instância da ação, ou seja, cada repetição possui uma ou mais iterações. Dessa forma, iterar uma array significa então passar por todos os elementos de uma array e executar um bloco de comandos. Para isso, utilizamos alguns laços de repetições.

Um laço de repetição nada mais é que um recurso que permite executar mais de uma vez trechos de código de acordo com uma condição. O PHP possui quatro estruturas de laços de repetição: **for**, **foreach**, **while** e **do-while**.

Os laços de repetição também são comumente conhecidos como **Loop**, ou **Looping**. Também são considerados estruturas de controle. Você pode estar vendo mais sobre esses laços de repetição no nosso curso introdutório de PHP e também no nosso artigo sobre foreach.

Iterando arrays com o Foreach

Existem vários modos para percorrer uma PHP array, no entanto, o mais simples deles é utilizando o laço de repetição foreach em PHP. Este comando funciona só com arrays e objetos, e retorna um erro quando utilizado com outros tipos de expressões.

O **PHP foreach** é um laço de repetição especializada na iteração de Arrays. Ou seja, ele funciona como uma estrutura que está projetada para percorrer todos os elementos de uma Array. Dessa forma, além de melhorar a legibilidade do código, também evitamos alguns problemas, como o acesso a elementos não existentes. Este é um problema que poder ocorrer quando trabalhamos com uma estrutura do laço de repetição **for** em sua definição básica.

Com o **PHP Foreach** temos acesso a todos os elementos, da mesma forma que teríamos se trabalhássemos com o **for** normal. Dessa forma, temos um laço de repetição que percorrerá todos os elementos e a cada ciclo será definido o próximo elemento contido na estrutura que está sendo iterada. Dessa forma, podemos entender o **foreach** como uma função que, a cada elemento de uma array, executa um bloco de ações definidas.

A sintaxe básica do PHP Foreach é conforme o código abaixo:

```
foreach ($array as $value) {
    //código a ser executado;
}
```

Dessa forma, para cada iteração do laço de repetição, o valor do elemento atual da Array é atribuído ao valor $value. Consequentemente o ponteiro da array é movido um a um, até atingir seu último elemento.

Vejamos então o exemplo abaixo:

```
<!DOCTYPE html>
<html>
<body>

<?php
    $cores = array("azul", "vermelho", "amarelo", "verde");

    foreach ($cores as $value) {
        echo "$value <br>";
    }
?>

</body>
</html>
```

Dessa forma, a nosso laço de repetição percorrerá cada elemento da nossa array $cores e irá imprimi-lo na tela, conforme na imagem abaixo:

Você pode estar aprofundando seus conhecimentos sobre o PHP Foreach em nosso artigo onde explicamos detalhadamente sobre ele e suas formas de utilização.

Iterando arrays com o For

O laço de repetição do PHP for executa um bloco de código um número especificado de vezes.

Dentre as estruturas de laço de repetição do PHP, ela é a mais complexa, porém ainda assim, será de fácil compreensão. Vamos analisar abaixo sua sintaxe básica:

```
for (contador; condição ; incremento ou decremento) {
    # code...
}
```

Vamos compreende-la agora.

O **Contador**, é como uma variável, geralmente utilizamos algo como $i. Também nessa etapa já colocamos o valor inicial do nosso contador, por exemplo, 0.

Já a **condição**, representa até quando o nosso laço de repetição irá funcionar.

O **incremento ou decremento** nada mais é que o valor que será adicionado para cada vez que o laço de repetição for executado. Geralmente utilizamos a termologia **$var++** para que adicione 1 ao valor a cada repetição, ou **$var–**, para que reduza em 1 o valor a cada repetição.

Portanto, vamos fazer a seguinte estrutura para poder realizar a iteração. Através da função **count()**, vamos definir o valor máximo ao qual o contador irá alcançar, ou seja, o número de elementos que compõem a nossa PHP array. Com isso, vamos aplicar então o seguinte código:

```
<?php
  $cores = array("azul", "vermelho", "amarelo", "verde");
  $valormax=count($cores);
for ($i = 0; $i < $valormax; $i++) {
```

```
  echo "$cores[$i] <br>";
}
?>
```

Dessa forma, a nosso laço de repetição percorrerá cada elemento da nossa PHP array $cores e irá imprimi-lo na tela, conforme na imagem abaixo:

Iterando arrays com o While

O comando de repetição **While**, nada mais é que “enquanto”. É o laço de repetição mais simples. Ele testa uma condição e caso verdadeira, executa um comando ou bloco de comandos. E repete isso enquanto a condição continuar sendo verdadeira.

A expressão só é testada a cada vez que o bloco de instrução termina, além do teste inicial. Portanto, se o valor da expressão passar a ser falsa no meio do bloco de instrução, ele ainda executará até o final este bloco, e só então, quando refazer o teste da condição, que finalizará o laço de repetição. Isso acontece, pois, se no teste inicial a condição for avaliada como false, o bloco de comandos não será executado.

Vejamos então a sintaxe abaixo do laço de repetição **while**:

```
while (condição) {
    #código
}
```

Portanto, vamos criar um laço de repetição com um contador dentro dele para realizar a iteração da nossa PHP Array. Observe que a estrutura é similar ao que fizemos no exemplo anteriormente, com o **for**.

```
<?php
  $cores = array("azul", "vermelho", "amarelo", "verde");
  $valormax=count($cores);
  $i=0;
while($i < $valormax){
  echo "$cores[$i] <br>";
  $i++;
}
?>
```

Dessa forma, nosso resultado será:

Conclusão: o que são arrays PHP

Com todas as informações apresentadas nesse tópico, com certeza você terá uma grande fonte de pesquisa e conhecimento para poder utilizar PHP array sempre que necessário. Portanto, recomendamos a leitura de todos os tópicos, e que experimentem executar na práticas os exemplos. Também recomendamos experimentar criar seus próprios códigos e criar suas próprias funções. Caso já possua conhecimento avançados, experimente executar funções envolvendo arrays e banco de dados. Continue seus estudos.

Segue algumas recomendações de estudos que você pode estar lendo em nosso blog para se tornar um desenvolvedor web e complementar seus estudos:

- O que é HTML
- HTML Básico
- HTML CSS
- O Que é PHP
- Curso Introdutório de PHP
- PHP Foreach
- PHP Explode: separando strings em arrays

Artigos relacionados

- PHP Isset: Tudo sobre essa função nativa do PHP
- Curso de PHP Introdutório: inicie seus estudos em PHP
- O que é PHP: entenda de forma descomplicada
- PHP 8: conheça as melhorias da nova versão do PHP
- Como Criar um Site com Cadastro de Clientes em WordPress
- PHP Date Format: Exibindo a modificando o padrão de Datas

https://www.homehost.com.br/blog/tutoriais/php/php-array/