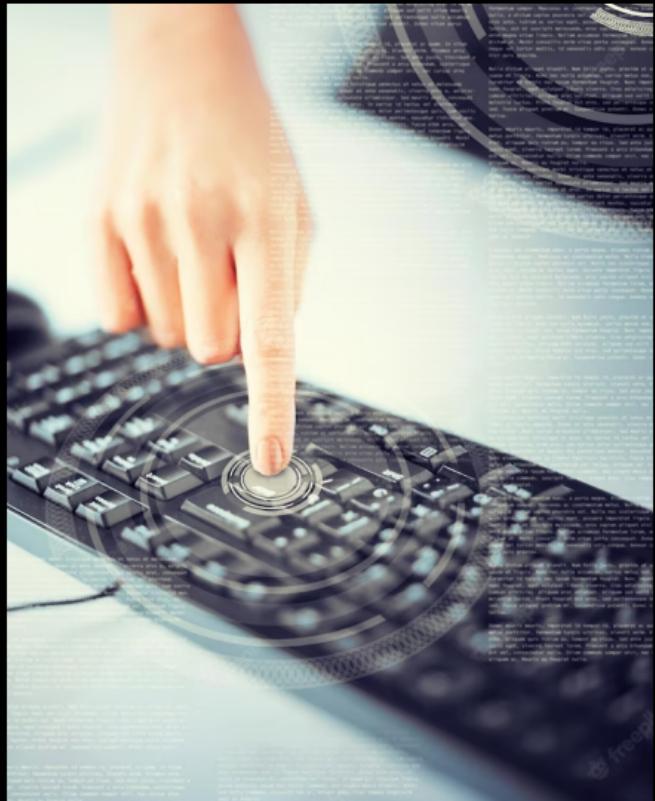


Fonte: Marcoratti

Explorando o Poder das Funções Anônimas em PHP

Prof. Wanderlei Silva do Carmo



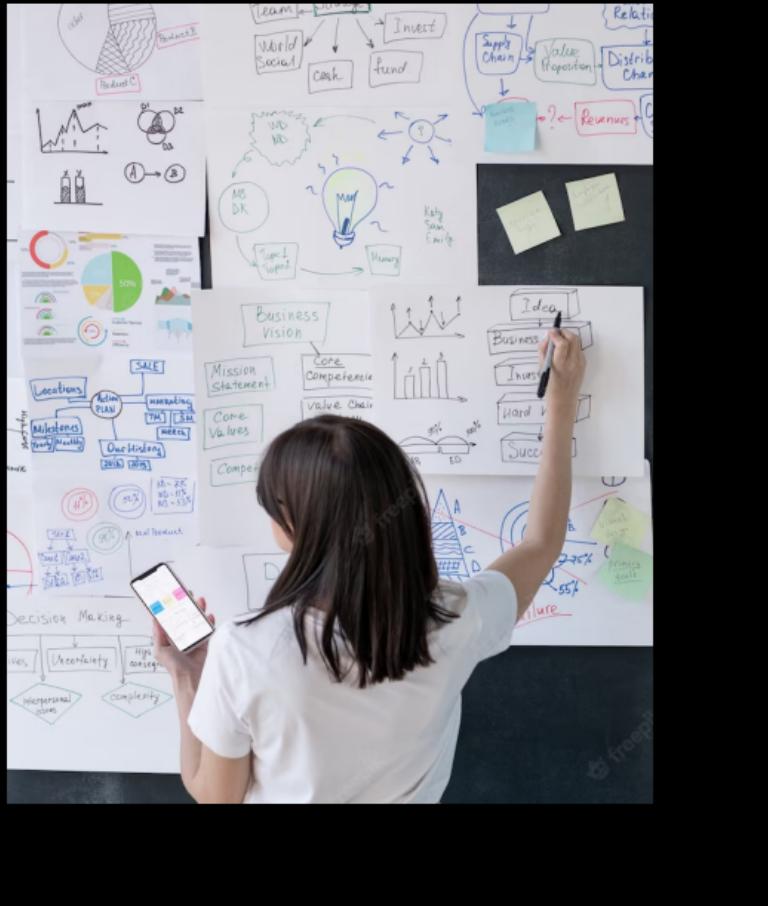
Introdução

As **funções anônimas** são uma poderosa ferramenta em **PHP**. Elas permitem criar blocos de código que podem ser usados como **callbacks** ou **closures**. Nesta apresentação, exploraremos o que são funções anônimas e como usá-las para criar código mais limpo e eficiente.

O que são funções anônimas?

As **funções anônimas** são funções sem nome que podem ser armazenadas em variáveis e passadas como argumentos para outras funções. Elas são especialmente úteis para **callbacks** e **closures**. As funções anônimas são definidas usando a palavra-chave `function`.





Exemplos de uso

As **funções anônimas** podem ser usadas de muitas maneiras em PHP. Alguns exemplos incluem: **callbacks** para funções de ordenação, **closures** para encapsular estado e comportamento, e **callbacks** para chamadas de API assíncronas. Com as funções anônimas, você pode criar código mais limpo e conciso.

Funções anônimas

Em PHP, funções anônimas são funções sem nome que podem ser criadas dinamicamente e armazenadas em variáveis, passadas como argumentos de função ou retornadas como valores de outra função. Essas funções são também conhecidas como closures (fechamentos) ou lambda functions (funções lambda).

A sintaxe básica para criar uma função anônima em PHP é a seguinte:

php

 Copy code

```
$funcaoAnonima = function($parametro1, $parametro2) {  
    // corpo da função  
};
```

Nesse exemplo, `'\$funcaoAnonima'` é uma variável que armazena uma função anônima que recebe dois parâmetros (`'\$parametro1'` e `'\$parametro2'`) e executa algum código dentro do corpo da função.

Funções anônimas

As funções anônimas em PHP são frequentemente usadas em conjunto com funções de ordem superior, que são funções que aceitam outras funções como argumentos ou as retornam como valores. Um exemplo comum é a função `'array_map()'`, que aplica uma função a cada elemento de um array e retorna um novo array com os resultados. Aqui está um exemplo de como usar uma função anônima com `'array_map()'`:

bash

 Copy code

```
$numeros = [1, 2, 3, 4, 5];
$quadrados = array_map(function($numero) {
    return $numero * $numero;
}, $numeros);

// $quadrados agora contém [1, 4, 9, 16, 25]
```

Nesse exemplo, a função anônima é usada para calcular o quadrado de cada elemento do array `'$numeros'` passado como segundo argumento para `'array_map()'`. A função anônima recebe um parâmetro `'$numero'`, que representa cada elemento do array, e retorna o resultado do cálculo `'$numero * $numero'`. O resultado é armazenado em um novo array `'$quadrados'`.

Funções anônimas

Outro exemplo comum de uso de funções anônimas é com a função `usort()`, que ordena um array de acordo com um critério definido por uma função de comparação. Aqui está um exemplo:

bash

 Copy code

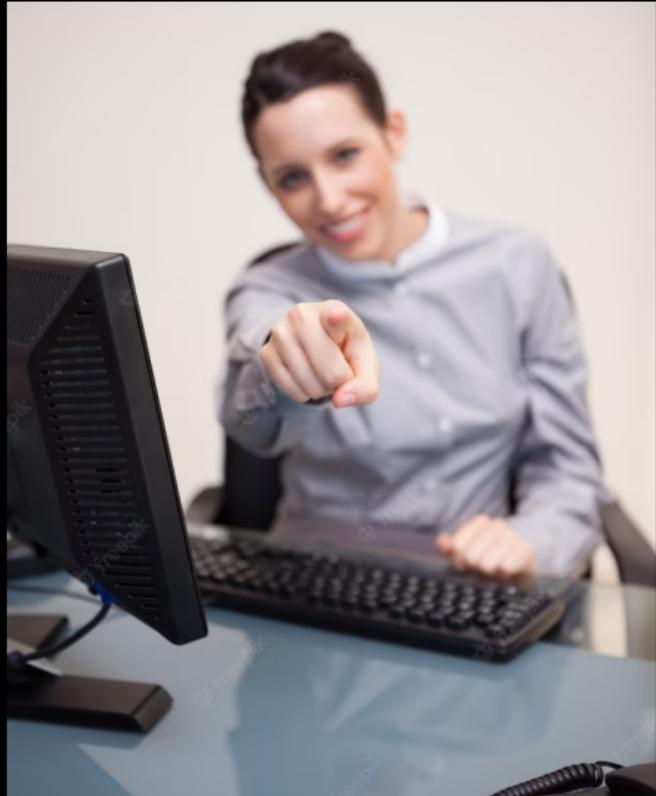
```
$nomes = ['Maria', 'João', 'Pedro', 'Ana'];
usort($nomes, function($nome1, $nome2) {
    return strcmp($nome1, $nome2);
});

// $nomes agora contém ['Ana', 'João', 'Maria', 'Pedro']
```

Nesse exemplo, a função anônima é usada para definir o critério de ordenação dos elementos do array `\$nomes`. A função anônima recebe dois parâmetros `\$nome1` e `\$nome2`, que representam dois elementos a serem comparados, e retorna o resultado da função `strcmp()`, que compara as strings `\$nome1` e `\$nome2`. O resultado é usado para ordenar o array em ordem alfabética crescente.

Callbacks

Os **callbacks** são funções que são passadas como argumentos para outras funções. Eles são comumente usados em funções de **callback** de eventos, como `addEventListener` em **JavaScript**. As funções anônimas são perfeitas para **callbacks**, pois permitem que você defina a função no local onde ela é usada.



Callbacks em PHP

Em PHP, um callback é uma referência a uma função que é passada como argumento para outra função. O termo "callback" refere-se ao fato de que a função passada como argumento será "chamada de volta" ou executada posteriormente pela função que a recebeu.

Os callbacks são muito úteis em situações em que queremos fornecer à função uma maneira flexível de executar algum código que depende de uma lógica específica. Ao passar uma função como callback, estamos permitindo que a função que a recebe execute essa lógica de forma personalizada.

Por exemplo, a função `usort()` recebe como argumentos um array e uma função de comparação que é usada para ordenar os elementos do array. Aqui está um exemplo:

bash

 Copy code

```
$frutas = array("laranja", "banana", "maçã", "limão");
usort($frutas, function($a, $b) {
    return strcmp($a, $b);
});
```

Callbacks em PHP

Outro exemplo de uso de callbacks é com a função `array_filter()`, que filtra os elementos de um array de acordo com uma função de filtro que é passada como callback. Aqui está um exemplo:

bash

 Copy code

```
$numeros = array(1, 2, 3, 4, 5);
$impares = array_filter($numeros, function($numero) {
    return $numero % 2 != 0;
});
```

Nesse exemplo, a função `array_filter()` recebe como argumentos o array `\$numeros` e uma função anônima que é usada como callback para filtrar os elementos do array que são ímpares. A função anônima recebe um parâmetro `\$numero`, que representa cada elemento do array, e retorna `true` se o número for ímpar e `false` caso contrário. O resultado dessa filtragem é armazenado em um novo array `\$impares`.

Closures

As **closures** são funções que capturam variáveis do escopo em que são definidas. Isso permite que elas mantenham o estado entre as chamadas de função. As funções anônimas são frequentemente usadas para criar **closures** em PHP.



Closures em PHP

Em PHP, closures são funções anônimas que podem ser armazenadas em variáveis e passadas como parâmetros para outras funções ou métodos. Uma closure pode "capturar" variáveis do escopo externo em que foi definida, o que a torna útil para criar funções que possuem um estado interno persistente.

Aqui está um exemplo de uma closure em PHP que adiciona um valor a uma variável definida no escopo externo:

```
php

function incrementador($valorInicial) {
    return function($valorAdicional) use (&$valorInicial) {
        $valorInicial += $valorAdicional;
        return $valorInicial;
    };
}

$adiciona5 = incrementador(5);
echo $adiciona5(3); // Output: 8
echo $adiciona5(2); // Output: 10
```

 Copy code

Closure em PHP

Neste exemplo, a função `incrementador()` retorna uma closure que "captura" a variável `\$valorInicial` definida no escopo da função externa. A closure recebe um parâmetro `\$valorAdicional` que é adicionado ao `\$valorInicial`. A cada chamada da closure, o `\$valorInicial` é atualizado e seu novo valor é retornado.

Ao chamar a função `incrementador(5)`, a closure é criada com o `\$valorInicial` definido como 5. A variável `\$adiciona5` é atribuída a esta closure e pode ser chamada múltiplas vezes, incrementando o `\$valorInicial` a cada chamada.

Neste exemplo, a closure é usada como uma função de incremento que mantém um estado interno persistente. Esse é apenas um exemplo de uso de closures em PHP, elas podem ser utilizadas em diversas situações, como funções de callback e gerenciamento de estados em loops.

Conclusão

As **funções anônimas** são uma ferramenta poderosa em **PHP**. Elas permitem criar código mais limpo e eficiente, especialmente quando usadas para **callbacks** e **closures**. Esperamos que esta apresentação tenha ajudado você a entender melhor o poder das funções anônimas em **PHP**.