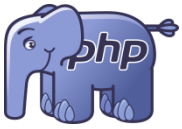

# Programação em PHP

Prof. Wanderlei Silva do Carmo

**Constantes e variáveis**

**Variáveis** são usadas para armazenar valores em PHP. O valor de uma variável pode ser alterado durante a execução do script. Variáveis têm escopo limitado, o que significa que elas só existem dentro do bloco de código em que foram definidas. Variáveis são definidas usando o sinal de igual (=). O nome de uma variável sempre começa com um cifrão ($).

**Constantes** são usadas para armazenar valores em PHP. Uma vez definido, o valor de uma constante não pode ser alterado durante a execução do script. Constantes têm escopo global e podem ser acessadas de qualquer lugar do script. Elas podem ser definidas usando a função define() ou a palavra-chave const. O nome de uma constante não tem um cifrão.

Constantes e variáveis são usadas para armazenar valores em PHP, mas existem algumas diferenças importantes entre elas:

- **Valor**: O valor de uma variável pode ser alterado durante a execução do script, enquanto o valor de uma constante não pode ser alterado depois de definido.
- **Escopo**: Variáveis têm escopo limitado, o que significa que elas só existem dentro do bloco de código em que foram definidas. Constantes, por outro lado, têm escopo global e podem ser acessadas de qualquer lugar do script.
- **Definição**: Variáveis são definidas usando o sinal de igual (=), enquanto constantes podem ser definidas usando a função define() ou a palavra-chave const.
- **Notação**: O nome de uma variável sempre começa com um cifrão ($), enquanto o nome de uma constante não tem um cifrão.

Essas são algumas das principais diferenças entre constantes e variáveis em PHP. É importante entender essas diferenças para usar corretamente cada tipo de dado em seus scripts.

```php
<?php
// variável do tipo String
$nome = 'Meu nome 123';

// variável do tipo inteiro
$ano = 2017;

// variável do tipo float
$pi = 3.14159265;

// variável do tipo booleano
$sim = true;
?>
```

```php
<?php

// Definindo uma constante usando a função define()
define('PI', 3.14);

// Definindo uma constante usando a palavra-chave const
const SITE_NAME = 'Meu Site';

// Acessando os valores das constantes
echo 'O valor de PI é: ' . PI;
echo 'O nome do site é: ' . SITE_NAME;
```

```
// Tentando alterar o valor de uma constante (gera um erro)
PI = 3.1415; // Erro: Cannot reassign a constant

?>
```

Variáveis globais em PHP são variáveis que são definidas fora de uma função e podem ser acessadas de qualquer parte do script. Existem algumas maneiras de usar variáveis globais em PHP:

**Usando a palavra-chave** global:
Você pode usar a palavra-chave global dentro de uma função para tornar uma variável global acessível dentro dessa função. Por exemplo:

```
<?php
$var_global = 'Olá';

function minhaFuncao() {
  global $var_global;
  echo $var_global;
}

minhaFuncao(); // imprime 'Olá'
?>
```

**Usando a variável superglobal** $GLOBALS: A variável superglobal $GLOBALS é um array associativo que contém referências para todas as variáveis definidas no escopo global do script. Você pode usar essa variável para acessar variáveis globais de dentro de uma função. Por exemplo:

```
<?php
$var_global = 'Olá';

function minhaFuncao() {
  echo $GLOBALS['var_global'];
}

minhaFuncao(); // imprime 'Olá'
?>
```

**Variáveis Superglobais**

Variáveis superglobais em PHP são variáveis pré-definidas que estão sempre disponíveis em todos os escopos do script. Isso significa que você pode acessá-las de dentro de funções ou métodos sem precisar usar a palavra-chave global. Algumas das principais variáveis superglobais em PHP incluem:

- $GLOBALS: um array associativo que contém referências para todas as variáveis definidas no escopo global do script.
- $_SERVER: um array contendo informações sobre o servidor e o ambiente de execução.
- $_GET: um array contendo as variáveis da requisição HTTP GET.
- $_POST: um array contendo as variáveis da requisição HTTP POST.
- $_FILES: um array contendo informações sobre os arquivos enviados via HTTP POST.
- $_COOKIE: um array contendo os cookies definidos na requisição HTTP.
- $_SESSION: um array contendo as variáveis de sessão.

**Vetores em PHP**

Nesta aula, vamos aprender sobre vetores em PHP. Vetores são estruturas de dados que permitem armazenar vários valores em uma única variável. Por exemplo, se quisermos guardar os nomes de cinco alunos, podemos usar um vetor chamado $alunos e atribuir os nomes a cada posição do vetor:

```
$alunos = array("Ana", "Bruno", "Carlos", "Daniela", "Eduardo");
```

Para acessar os valores do vetor, podemos usar o índice de cada posição, que começa em zero. Por exemplo, para imprimir o nome do primeiro aluno, podemos fazer:

```
echo $alunos[0]; // imprime Ana
```

Para percorrer todos os valores do vetor, podemos usar um laço for ou foreach. Por exemplo, para imprimir todos os nomes dos alunos, podemos fazer:

```
foreach ($alunos as $aluno) {
  echo $aluno . "\n";
}
```

Vetores em PHP podem armazenar valores de diferentes tipos, como números, strings ou até mesmo outros vetores. Além disso, podemos usar chaves associativas para dar nomes às posições do vetor, em vez de usar índices numéricos. Por exemplo, se quisermos guardar as notas de cada aluno, podemos usar um vetor associativo chamado $notas e atribuir as notas a cada nome:

```
$notas = array(
  "Ana" => 8.5,
  "Bruno" => 7.0,
  "Carlos" => 9.0,
  "Daniela" => 6.5,
  "Eduardo" => 10.0
);
```

Para acessar os valores do vetor associativo, podemos usar as chaves correspondentes. Por exemplo, para imprimir a nota da Daniela, podemos fazer:

```
echo $notas["Daniela"]; // imprime 6.5
```

Para percorrer todos os valores do vetor associativo, podemos usar um laço foreach com duas variáveis: uma para a chave e outra para o valor. Por exemplo, para imprimir o nome e a nota de cada aluno, podemos fazer:

```
foreach ($notas as $nome => $nota) {
  echo $nome . ": " . $nota . "\n";
}
```

Podemos ainda adicionar novos elementos ao final do vetor usando a função array_push(). Essa função recebe como primeiro argumento o nome do vetor e como segundo argumento o valor do novo elemento. Por exemplo, para adicionar a string "manga" ao final do vetor $frutas, podemos usar o seguinte código:

```
array_push($frutas, "manga"); // adiciona "manga" ao final de $frutas
```

**Pesquisa em vetor**

Uma forma de pesquisar um elemento no vetor em php é usar a função in_array, que retorna verdadeiro se o elemento estiver no vetor e falso caso contrário. Por exemplo:

```
$vetor = array(1, 2, 3, 4, 5); // cria um vetor com 5 elementos

$elemento = 3; // define o elemento a ser pesquisado

if (in_array($elemento, $vetor)) { // verifica se o elemento está no vetor
  echo "O elemento $elemento está no vetor."; // imprime uma mensagem positiva
} else {
  echo "O elemento $elemento não está no vetor."; // imprime uma mensagem negativa
}
```

**Filtrar elementos em um vetor**

Uma maneira de filtrar elementos em um vetor com php é usar a função array_filter. Essa função recebe um vetor como primeiro argumento e uma função de callback como segundo argumento. A função de callback deve retornar um valor booleano que indica se o elemento deve ser mantido ou descartado do vetor filtrado. Por exemplo, se quisermos filtrar um vetor de números e manter apenas os números pares, podemos fazer o seguinte:

```
$vetor = [1, 2, 3, 4, 5, 6];

$vetor_filtrado = array_filter($vetor, function($numero) {

  return $numero % 2 == 0; // retorna verdadeiro se o número for par

});
```

O vetor_filtrado terá os elementos [2, 4, 6]. Podemos usar qualquer critério de filtragem que quisermos na função de callback, desde que ela retorne um valor booleano.

Uma forma de filtrar os elementos de um array em PHP é usar a função array_filter. Essa função recebe um array e uma função callback que retorna verdadeiro ou falso para cada elemento do array. Se a função retornar verdadeiro, o elemento é mantido no array resultante. Se retornar falso, o elemento é descartado.

A função array_filter pode receber um terceiro argumento opcional que indica quais argumentos são passados para a função callback: o valor do elemento, a chave do elemento ou ambos. O valor padrão é 0, que passa apenas o valor do elemento.

Veja alguns exemplos de uso da função array_filter:

```
// Filtrar os números ímpares de um array

$array = [1, 2, 3, 4, 5];

$impares = array_filter($array, function ($valor) {

    return $valor % 2 == 1;

});

print_r($impares); // [1, 3, 5]

// Filtrar os países que começam com a letra B de um array associativo

$paises = [

    'BR' => 'Brasil',

    'US' => 'Estados Unidos',

    'PT' => 'Portugal',

    'AR' => 'Argentina'

];

$comecam_com_b = array_filter($paises, function ($chave) {

    return $chave[0] == 'B';

}, ARRAY_FILTER_USE_KEY);

print_r($comecam_com_b); // ['BR' => 'Brasil']

// Filtrar os nomes que têm mais de 5 letras de um array indexado

$nomes = ['Ana', 'Carlos', 'Beatriz', 'Daniel', 'Eduardo'];

$mais_de_5_letras = array_filter($nomes, function ($valor, $chave) {

    return strlen($valor) > 5;

}, ARRAY_FILTER_USE_BOTH);

print_r($mais_de_5_letras); // [2 => 'Beatriz', 4 => 'Eduardo']
```

**Excluir elementos de um vetor**

Para excluir elementos de um array em PHP, existem algumas funções que podem ser úteis, dependendo do caso. Uma delas é a função unset(), que remove o valor armazenado em uma variável ou em um índice de um array. Por exemplo:

Outra função que pode ser usada para excluir elementos de um array em PHP é a função array_splice(), que remove uma parte de um array e pode substitui-la por outra parte. A sintaxe dessa função é:

array_splice($array, $inicio, $tamanho, $substituto);

Onde $array é o array original, $inicio é o índice do primeiro elemento a ser removido, $tamanho é o número de elementos a serem removidos e $substituto é um array opcional que contém os elementos que serão inseridos no lugar dos removidos. Por exemplo:

```
//Declarar o array
$frutas = array("maçã", "banana", "laranja", "manga");
//Excluir dois elementos a partir do índice 1 e substituir por outro elemento
array_splice($frutas, 1, 2, "abacaxi");
//Imprimir o array
print_r($frutas);
```

O resultado seria:

```
Array
(
[0] => maçã
[1] => abacaxi
[2] => manga
)
```

Essas são algumas das funções que podem ser usadas para excluir elementos de um array em PHP. Existem outras funções que também podem fazer isso, como array_diff(), array_filter() e array_slice(), mas elas têm propósitos e comportamentos diferentes. O importante é saber qual função usar de acordo com a necessidade e o objetivo do código.

Nesta aula, vimos como criar e manipular vetores em PHP, usando índices numéricos ou chaves associativas. Vetores são muito úteis para armazenar e organizar dados em uma única variável. Para saber mais sobre vetores em PHP, consulte a documentação oficial: https://www.php.net/manual/pt_BR/language.types.array.php

**Manipulação de arquivos de texto em PHP:**

Para manipular arquivos de texto em PHP, você pode usar várias funções. Primeiro, para abrir um arquivo, você pode usar a função fopen(), que aceita dois argumentos: o nome do arquivo e o modo em que ele será aberto. Por exemplo, para abrir um arquivo chamado arquivo.txt no modo de escrita, você pode usar o seguinte código:

```
<?php
$arquivo = 'arquivo.txt';
$fp = fopen($arquivo, 'w');
?>
```

Depois de abrir o arquivo, você pode escrever nele usando a função fwrite(). Por exemplo, para escrever a string “Olá mundo!” no arquivo aberto anteriormente, você pode usar o seguinte código:

```
<?php
$conteudo = "Olá mundo!\n";
fwrite($fp, $conteudo);
?>
```

Para ler o conteúdo de um arquivo de texto, você pode usar a função fgets(), que lê uma linha do arquivo por vez. Você também pode usar a função feof() para verificar se chegou ao final do arquivo. Por exemplo, para ler todo o conteúdo do arquivo arquivo.txt e imprimir na tela, você pode usar o seguinte código:

```php
<?php
$arquivo = 'arquivo.txt';
$fp = fopen($arquivo, 'r');
while (!feof($fp)) {
   $linha = fgets($fp);
   echo $linha;
}
?>
```

Finalmente, depois de terminar de manipular o arquivo, é importante fechá-lo usando a função fclose(). Isso libera os recursos usados pelo arquivo e evita problemas futuros. Por exemplo, para fechar o arquivo aberto anteriormente, você pode usar o seguinte código:

```php
<?php
fclose($fp);
?>
```

**Funções em PHP**

Uma função é um bloco de código nomeado que executa uma tarefa específica, possivelmente processando um conjunto de valores fornecidos a ela e/ou retornando algum valor. Funções são úteis para reutilizar código e tornar o código mais legível e fácil de manter.

A sintaxe geral para definir uma função em PHP é a seguinte:

```
function nome_da_funcao($parametro1, $parametro2, ...) {
   // código da função
   return $valor;
}
```

O nome da função deve começar com uma letra ou sublinhado e pode conter letras, números e sublinhados. Os parâmetros são opcionais e são usados para passar valores para a função. A palavra-chave return é usada para especificar o valor que a função deve retornar.

Para chamar uma função, basta usar o nome da função seguido de parênteses contendo os valores dos parâmetros (se houver). Por exemplo, para chamar a função soma() com os valores 3 e 4 como parâmetros, você pode usar o seguinte código:

```
$resultado = soma(3, 4);
```

Neste exemplo, a função soma() é chamada com os valores 3 e 4 como parâmetros e o valor retornado pela função é armazenado na variável $resultado.

Existem vários tipos de funções em PHP. Algumas das principais categorias incluem:

1. **Funções de String**: Essas funções têm uma funcionalidade predefinida no PHP para trabalhar com strings. Exemplos incluem strlen(), que retorna o comprimento de uma string, e strpos(), que encontra a posição da primeira ocorrência de uma substring em uma string.
2. **Funções de Data**: Essas funções são funcionalidades predefinidas no PHP para trabalhar com datas e horas. Exemplos incluem date(), que formata uma data/hora local, e strtotime(), que analisa qualquer descrição de data/hora em texto em inglês em um timestamp Unix.
3. **Funções Numéricas**: Essas funções têm sua própria lógica predefinida fornecida pelo PHP, que é usada para operações numéricas. Exemplos incluem abs(), que retorna o valor absoluto de um número, e round(), que arredonda um número para o número inteiro mais próximo.

**Enviando arquivos via formulário HTML para o PHP**

Para enviar um arquivo e processá-lo com PHP, você precisará de duas coisas: um formulário para o usuário escolher o arquivo e um script PHP para receber os dados e salvar o arquivo enviado em uma pasta escolhida.

O formulário deve ter um campo de entrada do tipo file para permitir que o usuário selecione o arquivo a ser enviado. Além disso, o formulário deve ter o atributo enctype definido como multipart/form-data para permitir o envio de arquivos. Aqui está um exemplo de formulário para envio de arquivos:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  Selecione o arquivo:
  <input type="file" name="arquivo">
  <input type="submit" value="Enviar">
</form>
```

No exemplo acima, o formulário envia os dados para um script PHP chamado upload.php usando o método POST. O script PHP que recebe o arquivo enviado deve implementar qualquer lógica que for necessária para determinar o que deve ser feito com o arquivo enviado. Você pode, por exemplo, usar a variável $_FILES['arquivo']['size'] para descartar qualquer arquivo que seja muito pequeno ou muito grande.

Aqui está um exemplo de script PHP que recebe um arquivo enviado e salva-o em uma pasta chamada uploads:

```
<?php
if (isset($_FILES['arquivo'])) {
    $nome_arquivo = $_FILES['arquivo']['name'];
    $caminho_temp = $_FILES['arquivo']['tmp_name'];
    $caminho_salvar = 'uploads/' . $nome_arquivo;
    move_uploaded_file($caminho_temp, $caminho_salvar);
}
?>
```

**Include e Require em PHP**

include e require são duas instruções em PHP usadas para incluir o conteúdo de um arquivo em outro arquivo. A principal diferença entre elas é a forma como lidam com erros: se o arquivo a ser incluído não for encontrado ou não puder ser lido, require gerará um erro fatal e interromperá a execução do script, enquanto include apenas emitirá um aviso e continuará a execução do script.

Aqui está um exemplo de como usar include e require:

```
<?php
// usando include
include 'arquivo.php';

// usando require
require 'arquivo.php';
?>
```

Além de include e require, existem também as instruções include_once e require_once, que funcionam de maneira semelhante, mas verificam se o arquivo já foi incluído anteriormente e, em caso afirmativo, não o incluem novamente.