**Avançando com PHP**

**Depois de aprender um pouco sobre o PHP, já estamos preparados para avançar mais na tecnologia.**

**Se ficou alguma dúvida sobre o documento (Introdução ao PHP), faça algumas pesquisas e exercícios antes de iniciar essa documentação.**

Nome: Kaio Soares Alves

# O primeiro assunto que falaremos na nossa documentação são os arrays.

O que são arrays?

Os arrays ou vetores são o conjunto de dados dentro de uma variável.

Ex:

Texto

Descrição gerada automaticamente

Quando você executar esse programa, o número que aparecera e o **19**.

Porque o 19 se o 18 e o primeiro?

No array, quando você não define um índice para ele iniciar, ele necessariamente começa no índice 0, por isso apareceu o 19 e não o 18.

Para que consiga acessar os valores do array, e necessário usar um loop, que nem no exemplo abaixo.

Texto

Descrição gerada automaticamente

Nesse exemplo aparecera na saída todos os elementos do array em ordem **crescente**.

18 19 23 45 21 23 43

Mas como bom hábito devemos conhecer e utilizar a função **count**.

**O count tem como função “mostra” o número de elementos dentro do array.**

Se usarmos o count poderemos adicionar e retirar os elementos do array a vontade, sem precisar alterar o valor da condição no loop.

Segue o exemplo abaixo:

Texto

Descrição gerada automaticamente

Nesse exemplo aparecera na saída todos os elementos do array em ordem **crescente**.

17 19 23 45 21 23 43

# Falando de array, temos outro tipo de array, os arrays associativos. O que são esses arrays associativos?

Arrays associativos são estruturas onde cada elemento é identificado por uma chave única, podendo assim trazer diversas novas formas de usar os arrays.

Ex:

Texto

Descrição gerada automaticamente

**O programa exibira os seguintes dados:**

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Porém o PHP tem uma função que simplifica esse processo. Ela se chama

**Foreach**.

O**foreach** é uma estrutura de repetição da linguagem de programação PHP. Ele é usado para facilitar a iteração de estruturas como arrays, objetos e outros tipos que são iteráveis.

Texto

Descrição gerada automaticamente

No exemplo acima, peguei uma parte da estrutura do código anterior e a modifiquei, trocando o For pelo Foreach.

Repare como ficou mais simples navegar pelos arrays associativos usando o Foreach.

Usando essa estrutura de repetição não precisamos necessariamente saber qual o índice do array para acessar seu conteúdo.

Também temos outra forma de diminuir mais esse código.

Podemos pegar o bloco de execução {} do código anterior e substitui-lo pelas variáveis que definimos os índices.

Texto

Descrição gerada automaticamente

Assim o código fica menor.

Próximo assunto: Subrotina ou Function

# Subrotina ou Função (Function)

Uma função é uma pequena rotina nomeada que permite que um certo algoritmo seja reutilizado pela aplicação, garantindo que seu código não será duplicado e que a sua manutenção será feita em apenas um lugar.

**Qual a diferença entre Subrotina e função?**

**Uma função retorna um valor, ou seja, a chamada da função representa um valor após sua execução. Uma subrotina apenas executa um código isolado.**

**Sintaxe:**

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

Ou

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Para poder se aproximar mais no assunto Função, estarei disponibiliando uns links com artigos sobre o assunto.

Link:

<https://www.devmedia.com.br/php-funcoes/40688>

<https://www.php.net/manual/pt_BR/functions.variable-functions.php>

<https://www.w3schools.com/php/php_functions.asp>

# String complexas

Apesar do nome (String complexa), ela não tem nada de complexo, apenas leva esse nome por trabalhar com expressões complexas.

Pegando a imagem da Página 6 de exemplo, vemos um código com uma String escrita nele.

Texto

Descrição gerada automaticamente

Nessa imagem vemos um exemplo de hábitos ruins de como se escrever uma String complexa de maneira errada.

Não que tenha um jeito certo, mas podem ser malvistas pela comunidade.

Agora colocarei duas formas mais bem vistas de se escrever um String complexa.

**1° Forma-**

**Texto

Descrição gerada automaticamente**

Nessa forma simplismente adicionamos um par de **aspas duplas em volta de todo conteúdo**, retiramos os pontos e as aspas duplas que usamos para espaçar e separar os elementos, depois retiramos as **aspas simples** que foram usadas dentro dos **arrays** para indicar os **indices**.

**2° Forma-**

**Texto

Descrição gerada automaticamente**

Na segunda forma, fizemos um pouco diferente, adicionamos um par de **aspas duplas em volta de todo conteúdo**, retiramos os pontos e as aspas duplas que usamos para espaçar e separar os elementos, depois envolvemos os arrays com **chaves {}** e deixamos as aspas simples.

Obs: Minha opinião é que você deve usar a **2° forma**, pois você ira manter um padrão sempre que for indicar o indice do array com as aspas simples.

# Arquivo externo do PHP

Quando temos um código muito extenso no PHP, temos a opção de criar um arquivo externo e conectar os dois arquivos, para isso temos dois comandos diferente. Que são eles o include e require.

**Include –**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Require-**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

A diferença entre **include** e**require** é a forma como um erro é tratado. require produz um erro E\_COMPILE\_ERROR, o que encerra a execução do *script*. O include apenas produz um *warning* que pode ser "abafado" com @.

**include\_once** tem a garantia que o arquivo não será incluído novamente se ele já foi incluído antes.

**require\_once** é análogo ao include\_once

require\_once é um *statement* portanto a sintaxe com parênteses pode confundir um pouco. No primeiro olhar pode parecer que é uma função. Por isso não aconselho seu uso, apesar de não trazer consequências prejudiciais ao programa.

# Erros no PHP

O PHP trabalha com níveis de “erros”. Alguns destes níveis interrompem a execução do programa, enquanto outros níveis não.

Esses são os erros mais comuns que podem aparecer para você durante seu período de aprendizado ou até mesmo de uso profissional da linguagem.

**E\_ERROR (int.)** Erros fatais em tempo de execução. Isso indica erros que não podem ser recuperados, como um problema de alocação de memória. A execução do script foi interrompida.

**E\_WARNING** (int.) Avisos de tempo de execução (erros não fatais). A execução do script não é interrompida.

**E\_NOTICE (int.)** Avisos de tempo de execução. Indique que o script encontrou algo que pode indicar um erro, mas também pode acontecer no curso normal da execução de um script.

Para ver com mais detalhes estes níveis, você pode conferir a [documentação](https://www.php.net/manual/en/errorfunc.constants.php) deste assunto.

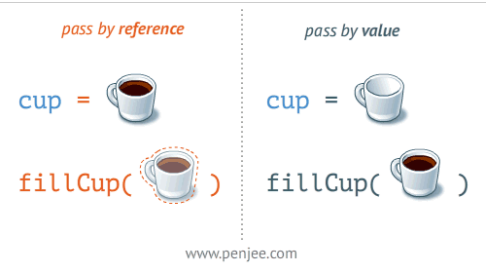
# Passagem de Parâmetros por Valor e Referência no PHP

**Valor**. Quando realizamos uma passagem de parâmetros baseada no valor, significa que uma função irá realizar suas operações baseado no valor do parâmetro informado para ela. Em outras palavras, o valor informado **será alterado apenas no escopo da função**. Essa é a maneira padrão de o PHP informar parâmetros.

**Referência**. Quando realizamos uma passagem de parâmetros baseada na referência, significa que uma função irá realizar suas operações baseado na referência original do parâmetro. Em outras palavras, se informarmos uma variável para uma função, o valor dessa variável **será alterado no escopo global**. Para realizar essa passagem, basta adicionar um **&** (“E” comercial) antes do parâmetro informado.

**Diagrama

Descrição gerada automaticamente**



Link: <https://hcode.com.br/blog/passagem-de-parametros-por-valor-e-referencia-no-php>

# Servidor web embutido

Desde a versão 5.4 o PHP CLI vem com um servidor http embutido. Ele pode ser usado para o desenvolvimento e testes, sem a necessidade de configurar e instalar um outro servidor (apache pu nginx, por exemplo).

Como a própria [documentação do PHP avisa](https://secure.php.net/manual/pt_BR/features.commandline.webserver.php), ele não é recomendado para ser usado em aplicações em produção, pois implementa funcionalidades simples do protocolo http.

**COMANDO BÁSICO**

Para iniciar o servidor localmente, basta acessar a pasta que deseja e rodar o seguinte comando:

$ php -S localhost:8000

**INFORMANDO UMA PASTA**

Como pode ser visto acima, utilizamos o parâmetro -S para indicar que queremos iniciar o servidor e em seguida informamos um endereço e porta desejados.

Para iniciar o servidor em uma pasta específica, basta informar o caminho junto ao comando:

$ php -S localhost:8000 public/

**INFORMANDO UM ARQUIVO DE ROTEAMENTO**

Também é possível informar um script de "roteamento", para onde todas as requisições serão enviadas. Isso é interessante caso você esteja usando algum framework (como Laravel)

$ php -S localhost:8000 public/index.php

**ACESSANDO REMOTAMENTE**

Para fazer com que o servidor seja disponível de máquinas remotas, utilize o ip 0.0.0.0 para funcionar com todas as interfaces do computador:

$ php -S 0.0.0.0:8000 public/index.php

Documentação PHP: <https://www.php.net/manual/pt_BR/features.commandline.webserver.php>