**01 Projeto da aula anterior**

Caso queira, você pode [baixar o projeto do curso](https://caelum-online-public.s3.amazonaws.com/1241-php/01/1241-aula01.zip) no ponto em que paramos na aula anterior.

**02 O que são variáveis**

**Transcrição**

Na última aula nós conversamos bastante sobre o que é o PHP e como executar um código PHP, e acabamos exibindo uma mensagem para o usuário. Essa mensagem é um dado fixo, e não conseguimos, depois de exibi-la, manipulá-la de alguma forma. Então, como é que conseguimos lidar com dados que mudam...ou seja, dados variáveis?

Uma variável, em uma linguagem de programação, é um espaço no programa que fica salvo na memória RAM, espaço esse a qual podemos adicionar alguma informação. A partir do momento em que a informação é guardada, podemos realizar algumas operações, modificar o valor dessa variável e utilizá-la de outras formas posteriormente - por exemplo, exibindo-a para o usuário.

Esse conceito de variável é utilizado em qualquer linguagem de programação, e até mesmo na matemática (quando estudamos o conceito de funções e equações). Quando precisamos de um dado que não seja fixo e inalterável, utilizamos as variáveis. Mas como criamos variáveis no PHP?

No nosso projeto, criaremos um novo arquivo chamado variavel.php. Queremos armazenar a minha idade para realizarmos algumas operações com ela. Sabemos que o nome desse dado é idade, e que a minha idade (no caso eu, Vinicius) é 21 anos. Queremos informar ao PHP que ele deve armazenar na memória, em um espaço chamado idade, o valor 21.

No PHP, não precisamos informar que uma variável existe antes de utilizá-la. Ao invés disso, no momento da utilização, podemos informar que queremos que a variável idade tenha o valor 21. Para informarmos que determinado nome é uma variável, precisamos adicionar um símbolo de cifrão $ ao início dele. Para que a variável receba um valor, usamos o sinal de igual =.

$idade = 21

Dessa forma, o valor 21 será atribuído à variável $idade. A partir de agora, em qualquer lugar que quisermos utilizar esse valor, poderemos chamar a variável $idade. Por exemplo, podemos exibi-la na tela chamando echo $idade, o que nos retornará...21! Se mudarmos o valor para 22 e executarmos novamente, esse será o valor exibido na tela.

Nós ainda podemos, inclusive, alterar esse valor depois que ele é definido. Por exemplo:

$idade = 22;

$idade = 21;

echo $idade;

Dessa forma, o valor exibido na tela será 21. Com isso, aprendemos o conceito de variáveis: um valor que, mesmo depois de definido, pode ser modificado. Atualmente temos uma variável cujo valor é um número, e podemos fazer operações com números - por exemplo, descobrir quantos anos terei daqui a 10 anos. Como fazemos esse tipo de operação matemática com o PHP?

Entenderemos mais sobre isso no próximo vídeo.

# 03 Variáveis

**Ao desenvolver um programa, certamente precisaremos armazenar dados. Seja para realizar cálculos, processamentos ou somente para exibir para o usuário. Para armazenar estes dados, utilizamos variáveis.**

**Qual a melhor definição do termo variável?**

Top of Form

* Alternativa correta



Um arquivo temporário que o PHP lê sempre que é executado

Bottom of Form

  Alternativa correta



Um espaço na RAM onde através de um nome, podemos manipular e acessar algum valor.

 Alternativa correta! Além disso, como o próprio nome já diz, variáveis podem variar, ou seja, seus valores podem ser alterados durante a execução do programa.

 Alternativa correta



Um número que representa a idade da pessoa que está programando

# 04 Operações matemáticas

## Transcrição

No último vídeo estávamos pensando sobre como realizar operações matemáticas com as nossas variáveis. Para entendermos esse processo, criaremos um arquivo operacoes.php.

Se você estiver utilizando algum editor de texto ou IDE que não o PhpStorm, lembre-se sempre de adicionar a extensão .php ao final do arquivo!

No exemplo anterior, a ideia era calcularmos a idade que eu teria em 10 anos. Portanto, criaremos uma variável $idadeDaqui10Anos. Sabendo que hoje eu tenho 21 anos, podemos adicionar a esse valor + 10:

$idadeDaqui10Anos = 21 + 10;

Com isso, realizaremos uma soma. Dentro de operações matemáticas, podemos inclusive utilizar outras variáveis:

$idade = 21;

$idadeDaqui10Anos = $idade + 10;

Dessa forma, o PHP entenderá que o valor de idadeDaqui10Anos é o valor de $idade, que definimos como 21, mais o valor 10, que não foi armazenado em outra variável. Podemos finalizar com echo $idadeDaqui10Anos para verificarmos se o PHP realiza essa operação corretamente.

Executando php operacoes.php no terminal, receberemos o valor "31". Se mudarmos o valor de $idade para 25, receberemos "35". Assim, aprendemos que uma operação matemática é muito simples de ser executada com o PHP, e ainda podemos utilizar variáveis dentro de uma operação.

Além da soma, podemos fazer outras operações, como a subtração:

$idade = 21;

$idadeHa5Anos = $idade - 5;

echo $idadeHa5Anos;

Dessa forma, o valor "16" será exibido na tela. Antes de continuarmos com outras operações, vamos apagar todo o código que escrevemos até agora. Em seguida, informaremos que $soma recebe 2 + 2; que $subtracao recebe 2 - 2; que $multiplicacao recebe 2 \* 2 (o asterisco é utilizado para essa operação); e $divisao recebe 2 / 2.

É interessante não utilizar cedilhas, tils e outros acentos em nomes de variáveis, pois às vezes elas podem atrapalhar o funcionamento da aplicação!

$soma = 2 + 2;

$subtracao = 2 - 2;

$multiplicacao = 2 \* 2;

$divisao = 2 / 2;

Essa são as quatro operações matemáticas básicas a partir das quais conseguiremos chegar em outros resultados. Por exemplo, se quisermos a potência de um número, por exemplo, 2 elevado ao cubo, poderíamos fazer 2 \* 2 \* 2. Existe ainda o operador de potência, representado por dois asteriscos \*\*.

$doisAoCubo = 2 \*\* 3;

//ou

$doisAoCubo = 2 \* 2 \* 2;

Ambas as construções, obviamente, retornam o mesmo resultado (8). Outra operação que podemos buscar é o $restoDaDivisao. Por exemplo, se dividirmos 10 por 3 e quisermos o quanto sobrou dessa operação, utilizamos, ao invés de /, o %:

$restoDaDivisao = 10 % 3;

echo $restoDaDivisao;

Executando esse código, teremos como retorno o valor "1". Isso porque 10 dividido por 3 é 3, e o resto é 1 (não estamos contando, claro, divisões que aceitam números decimais).

Repare, então, que fizemos todas essas operações com números inteiros. Será que é possível trabalharmos com números decimais? Que outros tipos de dados o PHP consegue trabalhar em uma variável?

Conversaremos sobre isso no próximo vídeo.

# 05 Matemática com PHP

**É mais do que comum nós precisarmos realizar operações matemáticas durante a execução de nosso programa. Quais os operadores matemáticos disponíveis no PHP?**

Top of Form

* Alternativa correta



+ (soma), - (subtração), \* (multiplicação), / (divisão)

Bottom of Form

  Alternativa correta



+ (soma), - (subtração), x (multiplicação), / (divisão), ^ (exponenciação), % (módulo ou resto da divisão)

  Alternativa correta



+ (soma), - (subtração), \* (multiplicação), / (divisão), \*\* (exponenciação), % (módulo ou resto da divisão)

Top of Form

* Alternativa correta! Com estes operadores nós conseguimos realizar as operações matemáticas com PHP

Bottom of Form

Parabéns, você acertou!

# 06 Tipos de dados

## Transcrição

Nós já entendemos por alto como o PHP funciona, o que são variáveis, como atribuí-las e como fazer operações com essas variáveis. Porém, até o momento, só conseguimos manipular números inteiros. Para essa etapa do curso, criaremos um novo arquivo chamado tipos.php. Conversaremos, então, sobre tipos de dados em PHP.

Sabemos que a variável $idade é um número inteiro, afinal, é assim que representamos idades no mundo real. Mas e valores em dinheiro? Além de R$1000,00 ou R$1001,00, um salário também pode ser R$1000,50, certo? Antes de nos aprofundarmos mais nisso, vamos voltar aos números inteiros.

Como podemos confirmar que o PHP entende 21 como um número inteiro? O PHP nos fornece uma funcionalidade para descobrir o tipo de uma variável. Por enquanto, não se atenha à sintaxe, pois ainda conversaremos sobre. Executaremos então echo gettype($idade). Esse método exibirá o tipo da variável que passamos por parâmetro.

$idade = 21;

echo gettype($idade);

Agora, se executarmos php tipos.php no terminal, teremos como retorno "integer" - ou seja, "inteiro". Criaremos, então, uma variável $salario recebendo 1000. Se exibirmos essa variável, também receberemos um "integer". Mas o que acontece se passarmos um número com casas decimais? E como devemos representar esse número?

Os números quebrados, na programação, são chamados de "números de ponto flutuante". Na prática, isso significa que são números decimais representados com pontos, ao invés de vírgulas como estamos acostumados no Brasil.

$salario = 1000.30;

echo gettype($salario);

Dessa vez, se executarmos php tipos.php, receberemos "double" como resposta. "Double" não significa que existem dois números após a vírgula, mas é como o PHP chama um número decimal. Outro nome a esse tipo de número, no PHP, é "float". No momento, trataremos esses nomes como sinônimos.

Em alguns casos, quando estivermos estudando mais a fundo, "double" significa algo como um número decimal com precisão dupla.

Já sabemos representar um número inteiro e um número decimal. Agora, tentaremos pegar o tipo do resultado de uma operação matemática:

$divisao = 10/3;

echo gettype($divisao);

Executando no terminal, o PHP retornará novamente um "double". Inclusive, se exibirmos somente o resultado dessa divisão (echo $divisao), teremos o valor dessa divisão, que é 3.333333 - que é uma dízima periódica. Portanto, o PHP consegue entender qual tipo de dados ele irá utilizar dependendo do seu valor, e não precisaremos declarar o tipo das variáveis da seguinte forma:

int $idade = 21;

float $salario = 1000.3;

float $divisao = 10 / 3;

Justamente por isso, o PHP é o que chamamos de uma **linguagem dinamicamente tipada**. Inclusive, uma variável $divisao do tipo double que mais tarde recebe um valor inteiro, por exemplo 3, passará a ser tratada como do tipo inteiro.

Além de números, temos outro tipo de dado, que é o texto - por exemplo, o nosso "Olá mundo!". Se executarmos a função gettype() para uma variável do tipo texto, teremos como retorno "string".

$texto = "Olá mundo!";

echo gettype($texto);

Em PHP, e em qualquer outra linguagem de programação, uma string é um conjunto de caracteres (formando uma palavra, uma frase, um texto, etc.). Imagine agora que queremos executar alguma ação dependendo de uma condição ser verdadeira ou falsa. Em programação, temos exatamente um tipo de dado que é $verdadeiro (representado pela palavra true) ou $falso (representado por false).

$verdadeiro = true;

$falso = false;

echo gettype($verdadeiro);

echo gettype($falso);

O PHP chama esse tipo de variável de "boolean" (ou booleana). Os únicos valores possíveis para uma variável booleana são verdadeiro ou falso. Com esse tipo de variável, conseguiremos tomar decisões nos nossos programas.

Até o momento vimos os tipos inteiro (integer), decimal (float/double), texto (string) e booleano (boolean), que são conhecidos como "tipos primitivos". Ainda existem outros tipos, chamados de "tipos complexos", que podem ser criados a partir deles.

Já entendemos bastante coisa sobre PHP, manipulação de variáveis e tipagem. Nesse processo, aprendemos o tipo string, de texto. Mas o que mais podemos fazer com esse tipo? E se quiséssemos exibir a minha idade junto ao texto Olá Mundo!, por exemplo?

Conversaremos melhor sobre isso na próxima aula.

# 07 Tipos

**Conversamos neste vídeo sobre os tipos de dados existentes no PHP.**

**Quais são os possíveis tipos de dados que uma variável PHP pode armazenar?**

Top of Form

* Alternativa correta



Inteiro, float, double, long, short, char, booleano e string

Bottom of Form

  Alternativa correta



Inteiro, decimal (chamado de float ou double), booleano (verdadeiro ou falso) e string (texto).

 Alternativa correta! Estes são os tipos primitivos do PHP. Por enquanto serão suficientes, mas há ainda tipos mais complexos que estudaremos mais adiante na segunda parte deste treinamento.

 Alternativa correta



Números e textos

# 08 Para saber mais: Dinheiro

Agora que já sabemos armazenar dados e realizar operações matemáticas com eles, você pode querer implementar programas que façam contas com valores monetários. Aplicar juros, descontos, etc.

Na computação existe um problema muito famoso de precisão. Existem vários guias que explicam como contornar este problema em linguagens diferentes, e um deles é o site [The Floating-Point Guide](https://floating-point-gui.de/).

Logo, caso você precise realizar operações em valores monetários (com qualquer linguagem), tome bastante cuidado!

# 09 Consolidando o seu conhecimento

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, segue uma passo a passo:

1) No PHPStorm, dentro do seu projeto, crie um novo arquivo variavel.php com o conteúdo abaixo:

<?php

$idade = 22;

$idade = 21;

echo $idade;

2) Agora execute o arquivo na linha de comando:

php variavel.php

3) De volta no PHPStorm crie mais um arquivo operacoes.php:

<?php

$soma = 2 + 2;

$subtracao = 2 - 2;

$multiplicacao = 2 \* 2;

$divisao = 2 / 2;

$doisAoCubo = 2 \*\* 3;

$restoDaDivisao = 10 % 3;

echo $restoDaDivisao;

4) Novamente teste e rode o arquivo na linha de comando:

php operacoes.php

5) Por fim, teste os tipos (a tipagem) no novo arquivo tipos.php:

<?php

$idade = 21;

$salario = 1000.301;

$divisao = 10 / 3;

$divisao = 3;

$texto = "Olá mundo";

$verdadeiro = true;

$falso = false;

echo gettype($verdadeiro);

6) Aqui também, execute o arquivo na linha de comando:

php tipos.php

### Opinião do instrutor

Continue com os seus estudos, e se houver dúvidas, não hesite em recorrer ao nosso fórum!

# 10 O que aprendemos?

Nessa aula, aprendemos:

* Uma variável guarda um valor
* Uma variável tem um $ na frente e para atribuir um valor usamos =
* O nome da variável não deve usar caracteres especiais
* Para imprimir o valor de uma variável também usamos o comando echo
* PHP dá suporte às operações matemáticas através dos seguintes operadores:
  + Soma: +
  + Subtração: -
  + Multiplicação: \*
  + Divisão: /
  + Resto da divisão (módulo): %
  + Potência: \*\*
* PHP possui tipos de dados como integer, double, string ou boolean (tipos primitivos)
* PHP descobre os tipos de variáveis dinamicamente (tipagem dinâmica)
* Utilizando gettype conseguimos descobrir o tipo de uma variável