**01 Projeto da aula anterior**

Caso queira, você pode [baixar o projeto do curso](https://caelum-online-public.s3.amazonaws.com/1241-php/02/1241-aula02.zip) no ponto em que paramos na aula anterior.

**02 Tipo string**

**Transcrição**

Nessa aula vamos trabalhar um pouco mais com a manipulação de textos no PHP. No nosso projeto, criaremos um arquivo chamado strings.php. Começaremos exibindo, como já fizemos antes, um Olá mundo.

<?php

echo 'Olá mundo';

No terminal, executaremos php strings.php e a mensagem "Olá mundo" será exibida. Porém, repare que no arquivo ola-mundo.php, utilizamos aspas duplas (") em volta da mensagem:

<?php

echo "Olá mundo";

Dessa vez, entretanto, utilizamos apóstrofes, também chamadas de aspas simples. Mas qual a diferença? Vamos supor que temos uma variável $idade recebendo o valor 21. Agora, queremos colocar esse valor na nossa mensagem:

<?php

$idade = 21;

echo 'Olá mundo! Minha idade é $idade';

Porém, se executarmos esse código, teremos como retorno "Olá mundo! Minha idade é $idade". Não é isso que queremos, certo? Para concatenarmos (ou seja, juntarmos) o valor de uma string com uma variável, precisaremos de outra construção. Fecharemos as aspas simples ao final do nosso texto, indicando que a string terminou, e usaremos o . para essa concatenação:

<?php

$idade = 21;

echo 'Olá mundo! Minha idade é ' . $idade;

Dessa forma, a mensagem exibida será "Olá mundo! Minha idade é 21". E se quisermos adicionar o texto "anos." ao final? Basta concatenarmos mais uma string:

<?php

$idade = 21;

echo 'Olá mundo! Minha idade é ' . $idade . ' anos.';

Quando temos somente uma variável, essa construção é bastante funcional. Porém, imagine o seguinte caso:

echo 'Olá! Meu nome é ' . $nome . '. Meu e-mail é ' . $email . '. Tenho ' . $idade . ' anos ';

Mesmo em uma frase tão pequena, não parece interessante concatenar as variáveis dessa forma. Seria ideal se tivéssemos uma forma de passá-las diretamente no texto...e isso é possível justamente com as aspas duplas:

<?php

$idade = 21;

echo "Olá mundo! Minha idade é $idade anos.";

Dessa forma, o PHP irá reconhecer a variável iniciada pelo $ e, após a leitura dela, continuará interpretando o texto normalmente. Se modificarmos o valor da variável, o PHP também conseguirá lê-la com sucesso. Com isso, é possível formar textos contendo valor de outros textos e variáveis, incluindo resultados de operações, e por aí vai.

No nosso próximo exemplo, queremos pular uma linha:

<?php

$idade = 21;

echo "Olá mundo!";

echo "Eu tenho $idade anos.";

Executando esse código, o texto será exibido todo junto:

"Olá mundo!Eu tenho 21 anos"

Ou seja, o PHP não tem quebra de linhas. Conversaremos mais sobre esse problema no próximo vídeo.

# 03 Variáveis em textos

**Vimos que há uma diferença entre aspas simples (’) e aspas duplas (”) no PHP. Esta diferença está na forma como ele interpreta os dados presentes no texto.**

**Qual das alternativas a seguir NÃO exibirá o valor da variável $idade corretamente?**

Top of Form

* Alternativa correta



echo “Eu tenho $idade anos”;

Bottom of Form

  Alternativa correta



echo ‘Eu tenho $idade anos’;

 Alternativa correta! Esta linha exibiria o texto Eu tenho $idade anos, sem interpretar o valor da variável $idade.

 Alternativa correta



echo ‘Eu tenho ‘ . $idade . ‘ anos’;

# 04 Caracteres especiais

## Transcrição

No último vídeo, conseguimos juntar um texto com o valor de uma variável, mas não conseguimos quebrar linhas. No PHP temos as strings, sequências de caracteres que formam um texto. O echo simplesmente exibe as strings e os valores que definimos, e não faz nada além disso. Além disso, os espaços e quebras de linha entre uma instrução do PHP e outra não são interpretados. Isso é uma vantagem, afinal, do contrário, teríamos que escrever todo o nosso código em uma única linha.

Se quisermos realizar uma quebra de linha, precisaremos instruir o PHP a incluir, além do texto, um caractere especial que quebra linhas. Para informarmos um caractere especial, utilizamos contra-barra (\) e o código que referencia esse caractere - nesse caso, n, de "new line".

<?php

$idade = 21;

echo "Olá mundo!\n";

echo "Eu tenho $idade anos.";

Dessa vez, a nossa mensagem será exibida da forma correta:

Olá mundo!

Eu tenho 21 anos

Devemos tomar cuidado com o uso de aspas simples ('), pois, nesses casos, o PHP entende que tudo compreendido por elas é uma string e não precisa ser interpretado. Dessa forma, o caractere especial \n não funcionará. Já com o uso de aspas duplas ("), o PHP entende que pode precisar interpretar algo, seja uma variável ou um caractere especial.

E se, ao invés de uma linha vazia, quiséssemos dar um Tab para frente? Se fizermos echo " Eu tenho $idade anos.";, esse espaçamento será exibido corretamente. Mas será que é possível salvar o valor desse Tab em um caractere? Na verdade sim, por meio de \t:

echo "\t Eu tenho $idade anos.";

Existem diversos caracteres especiais, e o PHP consegue exibí-los por meio da contra-barra e da identificação desse caractere. Na [página sobre strings do PHP](https://www.php.net/manual/pt_BR/language.types.string.php), é possível encontrar uma lista das chamadas "sequências de escape", que são caracteres especiais.

Normalmente, os arquivos do Windows trabalham com \r\n, ou seja, o caractere de retorno de carro somado ao caractere de nova linha, para identificar uma quebra de linha. Para não nos preocuparmos com esses detalhes, ou mesmo precisarmos lembrar de cada referência de caractere especial, o PHP criou uma forma um pouco mais legível de quebrar linhas. Para isso, ao invés de \n, concatenaremos a string com um valor do PHP chamado PHP\_EOL ("end of line"), que tem o valor do caractere especial de quebra de linha do sistema operacional em que é executado.

<?php

$idade = 21;

echo "Olá mundo!" .PHP\_EOL;

echo "Eu tenho $idade anos.";

Agora que já aprendemos bastante sobre o PHP, dedicaremos o próximo vídeo para revisarmos o conteúdo abordado até aqui.

# 05 Quebra de linha

**Aprendemos até agora mais de uma forma de quebrar linha em um texto no PHP. Conseguimos fazer isso com a sequência de escape \n e com a constante chamada PHP\_EOL.**

**Por que utilizar PHP\_EOL?**

Top of Form

* Alternativa correta



Porque esta constante leva em consideração os diferentes caracteres utilizados para quebrar linha em diferentes sistemas operacionais.

Bottom of Form

 Alternativa correta! Alguns sistemas operacionais utilizam o caracter \n para representar uma quebra de linha. Outros, utilizam \r. Já o Windows utiliza ambos (\r\n). Utilizando a constante PHP\_EOL nos é abstraída esta diferença e podemos deixar nosso código funcionando de forma igual em todas as plataformas.

 Alternativa correta



Porque PHP\_EOL é mais legível do que \n

 Alternativa errada! Esta não é a principal vantagem e alguns discordam que desta forma seja mais legível.

 Alternativa correta



Porque PHP\_EOL é lido mais rápido pelo PHP

# 06 Revisão até aqui

## Transcrição

Nesse vídeo vamos recapitular os conteúdos que aprendemos até agora. Começamos a trabalhar escrevendo comandos diretamente no terminal do php, que foi acessado com php -a. Lá, foi possível declarar uma variável, exibir textos, entre outras funcionalidades. Nesse ponto, aprendemos que toda instrução do PHP deve terminar com ;.

Depois dessa execução, passamos a escrever nossos códigos em um arquivo separado, de modo a conseguirmos executar uma sequência de comandos. Então, entendemos o que são variáveis (que sempre se iniciam com $) e como armazenar, manipular e exibir os dados. Também aprendemos a realizar operações com esses dados, como soma, subtração, multiplicação e divisão.

Todos esses dados manipulados eram do tipo inteiro (integer), o que abriu uma brecha para conversarmos a respeito de outros tipos, como decimais (double/float), texto (string) e booleano (boolean). A partir disso, começamos a juntar textos diferentes e a exibir variáveis dentro de um texto.

Vimos que valores entre aspas simples (') são apenas lidos como strings, enquanto valores entre aspas duplas (") podem ser interpretados. Dessa forma, podemos passar variáveis ou referências de caracteres especiais entre aspas duplas. Também aprendemos a concatenar valores próprios do PHP, como o PHP\_EOL:

<?php

$idade = 21;

echo "Olá mundo!" .PHP\_EOL;

echo "Eu tenho $idade anos.";

Agora começaremos a trabalhar mais ativamente no código. Por exemplo, imagine a seguinte situação:

echo "Você só pode entrar se tiver mais de 18 anos\n";

echo "Você tem $idade anos. Pode entrar";

Nesse caso, a segunda mensagem só fará sentido se o valor informado na variável $idade for igual ou maior a 18. Porém, se passarmos valores abaixo deste, a mensagem continuará sendo exibida. Ou seja, o nosso código não consegue tomar decisões antes de executar ações. Na próxima aula, começaremos a trabalhar com decisões, fluxos e fluxos alternativos.

# 07 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, seguem os passos mais importantes:

1) No PHPStorm crie um novo arquivo strings.php com o conteúdo abaixo:

<?php

$idade = 16;

echo "Olá mundo!" . PHP\_EOL;

echo "Eu tenho \"$idade\" anos";

echo PHP\_EOL . PHP\_EOL;

echo "Você só pode entrar, se tiver mais de 18 anos\n";

echo "Você tem $idade anos. Pode entrar";

2) Agora chame o arquivo na linha de comando:

php strings.php

### Opinião do instrutor

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

# 08 O que aprendemos?

Nesta aula, aprendemos:

* Para concatenar uma string com uma outra variável usamos o . (ponto)
* Podemos definir uma string com aspas simples(') ou duplas(")
* Para interpretar um valor da variável ou caractere especial dentro da string devemos usar aspas duplas
* caracteres especias são:
  + Nova linha: \n ou \r
  + Tab: \t
* PHP possui uma constante para indicar uma quebra de linha: PHP\_EOL