

# Caracteres de escape

- ▶ Para imprimir (escrever) alguns caracteres, pode ser necessário *escapá-los*. Para isso, utilizamos a barra invertida (\)
- ▶ Além disso, a barra invertida, se utilizada em conjunto com certos caracteres, passa a ter significado especial

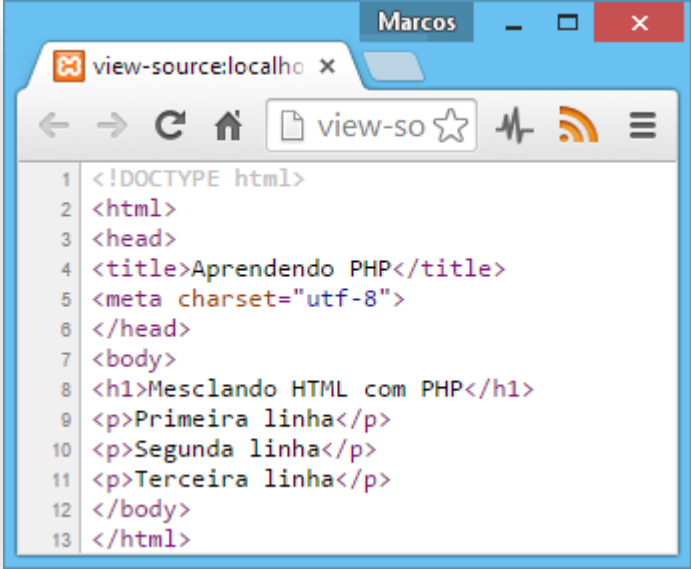
Sintaxe	Significado
\n	Nova linha
\r	Retorno de carro (semelhante a \n)
\t	Tabulação horizontal
\\	A própria barra (\)
\\$	O símbolo \$
\'	Aspa simples
\"	Aspa dupla

# Exercício 3

- a) Crie uma cópia do arquivo `ex2.php`, chamada `ex3.php`.  
Adicione `\n` ao final de cada linha do comando `echo` em `ex3.php`, conforme abaixo:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Aprendendo PHP</title>
5 <meta charset="utf-8">
6 </head>
7 <body>
8 <h1>Mesclando HTML com PHP</h1>
9 <?php
10     echo "<p>Primeira linha</p>\n";
11     echo "<p>Segunda linha</p>\n";
12     echo "<p>Terceira linha</p>\n";
13 ?>
14 </body>
15 </html>
```

- b) Acesse o arquivo pelo navegador e visualize o código-fonte



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Aprendendo PHP</title>
5 <meta charset="utf-8">
6 </head>
7 <body>
8 <h1>Mesclando HTML com PHP</h1>
9 <p>Primeira linha</p>
10 <p>Segunda linha</p>
11 <p>Terceira linha</p>
12 </body>
13 </html>
```

- c) Teste com outros caracteres de escape

# Variáveis

- ▶ São identificadores para representar valores mutáveis e voláteis
- ▶ Só existem durante a execução do programa
  - ▶ Armazenadas na memória RAM
- ▶ Para criar uma variável em PHP, devemos atribuir um nome, precedido do caractere cifrão (\$)

```
<?php
    $nome = "João";
    $sobrenome = " da Silva";
    echo $nome . $sobrenome;
?>
```

**Concatenação:** Unir o conteúdo de duas *strings*.

Em PHP, o operador de concatenação é o ponto (.)

## Observação

PHP é uma linguagem **fracamente tipada**. Isto significa que não é necessário expressar o tipo das variáveis.

# Variáveis *(continuação)*

Algumas dicas para nomenclatura de variáveis:

- ▶ Nunca inicie com números
- ▶ Nunca utilize espaços em brancos no meio
- ▶ Nunca utilize caracteres especiais (!@#\$%&\* () [] {})
- ▶ Nomes de variáveis devem ser significativos e transmitir a ideia de seu conteúdo
- ▶ Utilizar preferencialmente letras em minúsculo
  - ▶ Em caso de mais de uma palavra, separe pelo caractere *underline* (\_) ou use as iniciais em maiúsculo

```
<?php
    $codigo_cliente = 12345;
    $codigoCliente = 12345;
?>
```

## Observação

O PHP é *case sensitive*, isto é,  
\$codigo é diferente de \$Codigo

# Exercícios 4 e 5

- 4) Teste a criação e concatenação de variáveis. Salve como `ex4.php`

## Exemplo 1:

```
<?php
$primeiro_nome = "Marcos";
$nome_do_meio = "Alves";
$sobrenome = "Vieira";
echo $primeiro_nome . $nome_do_meio . $sobrenome;
?>
```

## Exemplo 2:

```
<?php
$primeiro_nome = "Marcos";
$nome_do_meio = "Alves";
echo $primeiro_nome . " " . $nome_do_meio;
?>
```

Use sua criatividade para fazer mais testes.

Aproveite para ver o que acontece quando você esquece um ponto-e-vírgula ou o cifrão.

- 5) Crie o arquivo `ex5.php` e teste o uso de variáveis, mesclando código PHP com HTML. Experimente fazer diversos trechos PHP (`<?php ... ?>`) e responda a si mesmo a pergunta: É possível utilizar uma variável criada em outro trecho PHP, dentro do mesmo arquivo?

**Dica:** Utilize como base o código do arquivo `ex3.php`

# Tipos de dados

## Booleano

- ▶ Expressa um valor lógico que pode ser *verdadeiro* ou *falso*
- ▶ Para especificar um valor booleano, deve-se utilizar as palavras TRUE ou FALSE

```
<?php
    $estou_feliz = TRUE;
    $vai_chover = ($umidade > 90);
?>
```

Repare que o valor TRUE (ou FALSE) do tipo booleano não deve estar entre parênteses.

- ▶ Também são considerados valores *falsos* em comparações booleanas:
  - ▶ Inteiro 0
  - ▶ Ponto flutuante 0.0
  - ▶ Uma *string* vazia "" ou "0"
  - ▶ Um *array* vazio
  - ▶ Um objeto sem elementos
  - ▶ Tipo NULL
- ▶ Qualquer valor diferente dos acima é considerado como *verdadeiro*

# Tipos de dados

## Numérico

- Tipos numéricos podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), precedido ou não de sinal (+ ou -)

```
<?php
//número decimal
$a = 1234;
//número negativo
$a = -1234;
//número octal (equivalente a 83 em decimal)
$a = 0123;
//número hexadecimal (equivalente a 26 em decimal)
$a = 0x1A;
//ponto flutuante
$a = 1.234;
//notação científica
$a = 4e23;
?>
```

# Tipos de dados

## *String*

- ▶ Uma *string* é uma cadeia de caracteres alfanuméricos
- ▶ Para declarar uma *string*, podemos usar aspas simples ( ' ') ou duplas ( " ")

```
<?php
    $variavel = 'Isto é um teste';
    $variavel = "Isto é um teste";
?>
```



# Exercício 6

Pesquise a(s) diferença(s) entre o uso de aspas simples e aspas duplas em PHP. Faça alguns exemplos e salve seu arquivo como `ex6.php`

# Tipos de dados

## *Array* ou vetor

- ▶ *Array* é uma lista de valores
- ▶ Cada elemento de um *array* pode ser de um tipo diferente (números, *strings*, objetos)
- ▶ Um *array* pode crescer dinamicamente

```
<?php
    $carros = array("Palio", "Corsa", "Gol");
    echo $carros[1];      //resultado = "Corsa"
?>
```

### Nota

*Arrays* serão tratados com maiores detalhes futuramente.

# Tipos de dados

## NULL

- ▶ A utilização do valor especial NULL significa que a variável não tem valor
- ▶ NULL é o único valor possível do tipo NULL

```
<?php
    $valor1 = NULL;    //não há valor algum
    $valor2 = 100;     //valor decimal inteiro 100
    $valor3 = 0;       //valor decimal inteiro 0
?>
```

# Tipos de dados

## Constante

- ▶ Valor que não sofre modificações durante a execução do programa
- ▶ Só pode conter valores escalares
  - ▶ Booleano, inteiro, ponto flutuante e *string*
  - ▶ Não pode conter outros valores, como vetores e objetos
- ▶ Seguem as mesmas regras de nomenclatura das variáveis
  - ▶ Com exceção de que não devem ser precedidas de \$
  - ▶ Geralmente utiliza-se nomes em maiúsculo

```
<?php
define("NUMERO_PI", 3.14159265359);
$raio = 10;
echo "A circunferência é " . 2 * NUMERO_PI * $raio;
?>
```

# Operadores

## Atribuição

- ▶ Utilizado para atribuir valor a uma variável
- ▶ O operador básico de atribuição é o sinal de igual (=)

```
<?php
    $var = 10; //atribui o valor 10 a $var
    $var += 5; //soma 5 em $var
    $var -= 5; //subtrai 5 em $var
    $var *= 5; //multiplica $var por 5
    $var /= 5; //divide $var por 5
    $var %= 3; //$var recebe o resto de sua divisão por 3
?>
```

# Operadores

## Atribuição (*continuação*)

### ► Pré e pós incremento/decremento:

Operadores	Descrição
++\$a	Pré-incremento: incrementa \$a em 1 e, então, retorna \$a
\$a++	Pós-incremento: retorna \$a e, então, incrementa \$a em 1
--\$a	Pré-decremento: decrementa \$a em 1 e, então, retorna \$a
\$a--	Pós-decremento: retorna \$a e, então, decrementa \$a em 1

```
<?php
    $teste1 = 10;      //atribui o valor 10 a $teste1
    $teste2 = 10;      //atribui o valor 10 a $teste2

    echo --$teste1;    //imprime 9 e $teste1 vale 9
    echo $teste2--;    //imprime 10 e $teste2 vale 9
?>
```

# Exercício 7

Teste livremente TODOS os operadores de atribuição apresentados, incluindo pré e pós incremento/decremento.

Salve seu arquivo como `ex7.php`

Exemplo 1:

```
<?php
    $nome = "Felipe";
    $idade = 17;
    echo "<p>Eu sou $nome e tenho $idade anos</p>";
?>
```

Exemplo 2:

```
<?php
    $var = 100;
    $var += 25; //soma 25 em $var
    echo $var++;
    echo $var;
?>
```

## Dica

Use diferentes tipos de dados, como números inteiros, ponto flutuante, constantes, *strings*...