Fundamentos da linguagem PHP – inserindo PHP no HTML

- Usam-se as tags **<?php** e **?>**

- Cada região compreendida por estas tags é utilizada para a inserção de comandos e declarações da linguagem PHP;

- Um arquivo HTML pode conter diversas regiões de código em PHP, isto é, podemos abrir e fechar estas tags quantas vezes quisermos, desde que elas estejam sempre aos pares;

- Estes blocos onde o interpretador PHP é chamado podem aparecer antes de <html> ou após </html>. Virtualmente, seções PHP podem estar em qualquer lugar do documento;

- Sempre que o PHP produzir saída para o navegador, é aconselhável que tais códigos estejam dentro da área <body> do documento HTML, facilitando a renderização do conteúdo pelo navegador;

Execução de código em PHP

- Quando, num documento web, o PHP é chamado para executar os comandos da linguagem, ele devolve a reposta para o navegador no formato HTML somente ou em texto puro. Mesmo que o usuário tente visualizar o código-fonte da página recebida, está não mostrará nenhum comando em PHP contido no documento;

- Sendo assim, o navegador só recebe do servidor linhas de código em HTML ou texto puro, mesmo que a resposta da requisição tenha sido produzida por um script em PHP no servidor;

- Todo o trabalho do PHP é realizado no servidor;

- Quando o arquivo contendo comandos em PHP é salvo com a extensão html, qualquer comando da linguagem é simplesmente ignorado pelo servidor e não é executado;

- Todo arquivo que contiver, nem que seja apenas uma única linha em PHP, dever ser salvo com a extensão **.php**.

Comentários

- Forma 1: comentário de uma linha

```
//comentário em PHP
```

- Forma 2: comentário de uma linha

```
#comentário em PHP
```

- Forma 3: comentário de múltiplas linhas

```
/* Um comentário
 de diversas
 linhas */
```

Enviando uma resposta para o cliente

- Todo o resultado do processamento feito pelo PHP no servidor é devolvido ao navegador através do construtor da linguagem chamado **echo**;

- Algumas formas de se usar echo:

```
echo $variável1, $variável2;

echo CONSTANTE;

echo "$variável1 $variável2";

echo '$variável';

echo "Texto ou <html> ou <script>";

echo 'Texto ou <html> ou <script>';

echo ("Com parênteses");
```

- Podemos usar aspas "" ou apóstrofos '' em PHP;
- Tudo o que você colocar entre apóstrofos em um echo é enviado para o cliente literalmente, como texto puro. Com apóstrofos, variáveis **não são interpoladas**. Se você quiser que o navegador receba **O VALOR** de uma variável, use sempre aspas com echo;
- Qualquer resposta produzida por echo deve estar dentro de <body>.

- **print** tem a mesma função de echo.

Criação de variáveis em PHP

- PHP é uma linguagem *fracamente tipada*;
- Sendo assim, não há a necessidade de se definir um tipo de dado para a variável antes de ser utilizada;
- Uma variável pode, no decorrer do script, armazenar valores de diversos tipos. Quem deve controlar este uso é o próprio desenvolvedor;
- Uma variável em PHP não exige declaração em uma região específica do script;
- Variáveis em PHP não necessitam nem mesmo serem inicializadas com um valor, embora o interpretador mostre um aviso se isso acontecer;
- É sempre uma boa prática de desenvolvimento criar variáveis no início de um script e também inicializá-las.

Regras para nomear variáveis

- **Nomes de variáveis:**
  - Devem sempre começar com o cifrão $;
  - Seguindo o cifrão, deve aparecer ou uma letra (a-z, A-Z) ou o sublinhado _. É proibido usar um dígito na segunda posição;
  - O restante do nome da variável pode conter qualquer combinação de letras, dígitos e/ou sublinhado;
  - Espaços em branco, sinais de pontuação e outros símbolos como *, %, #, /, etc... são proibidos no nome;
  - Como qualquer outro identificador em PHP, nomes de variáveis admitem caracteres acentuados, pois a linguagem usa o padrão Unicode de caracteres (UTF-8). *Mas esta prática não é aconselhável*.

Variáveis – note bem

- Nomes de variáveis devem ser únicos em todo o documento;
- Nomes de variáveis são sensíveis ao caso em PHP;
- Não é obrigatório, mas as seguintes convenções devem ser seguidas:
  - Use letras minúsculas;
  - Use nomes descritivos, dando significado à informação que a variável armazena;
  - Se a variável tiver nome composto, escreva todos os nomes grudados e escreva somente a letra inicial de cada nome, a partir do segundo, em maiúscula. Assim: `$umNomeGrandeDeVariavel`. Esta técnica é conhecida como camelCase;

Constantes em PHP

- Armazenam valores que não podem ser mudados dentro do script;
- Seguem as mesmas regras de nomes de variáveis, mas não começam com cifrão ($);
- Nomes de constantes são escritos em maiúsculo (convenção);
- Com nome composto, separe-o com sublinhado (convenção);
- Criam-se constantes assim:

```
define('NOME_DA_CONST', valor);
```

Para escrever o valor de uma constante:

```
echo TAXA;

print TAXA;
```

- Sem aspas ou apóstrofos, ok?

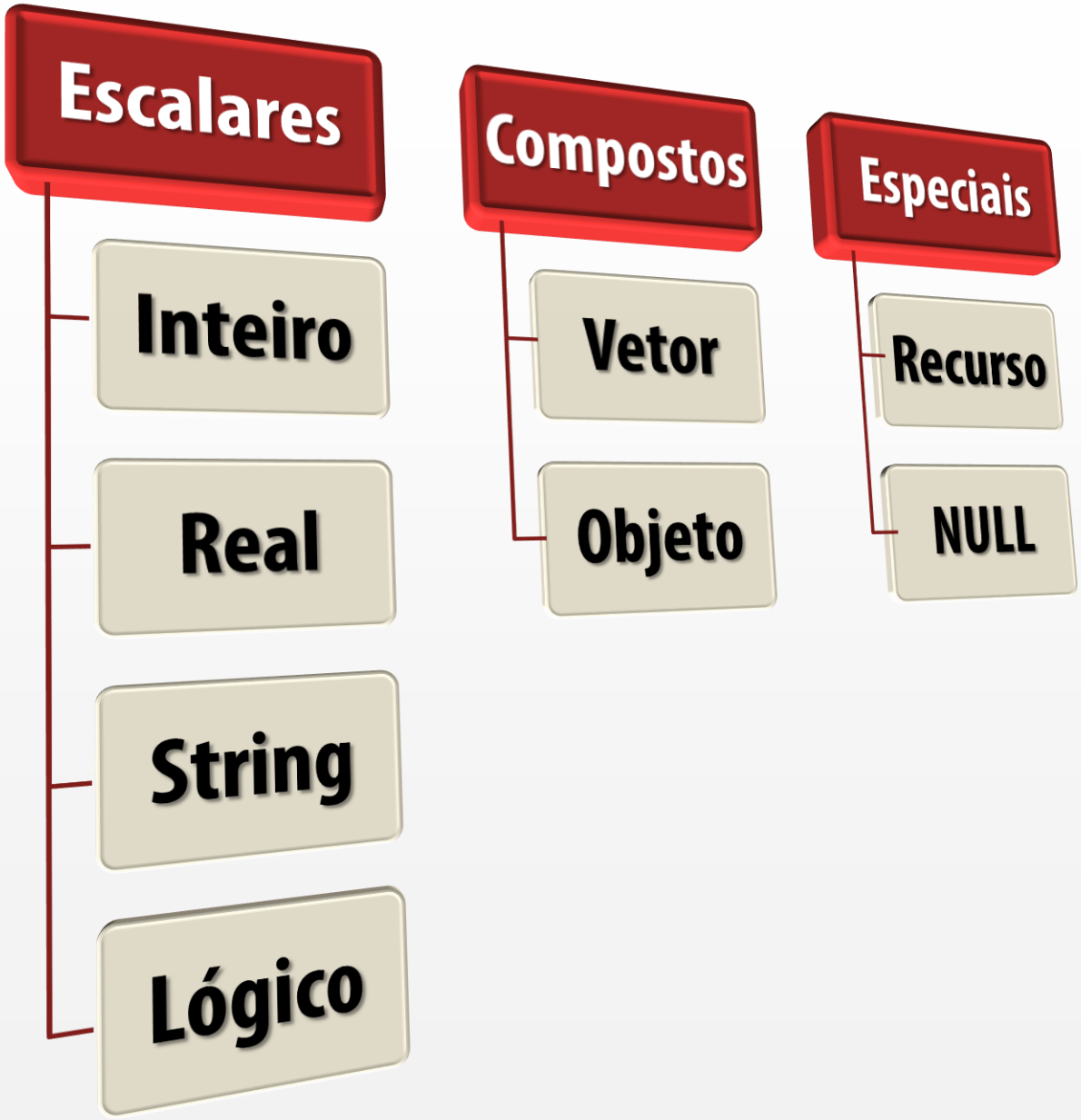
```
define('TAXA', 0.75);
define("ENDERECO", "Rua das Acácias, 12");
define("NOME", 'Pedro dos Santos');
define('IDADE', 46);
define('PODE_VOTAR', false);
```

Atribuição de valores a variáveis

- Em PHP, usa-se o operador = para associar um valor a uma variável;
- Exemplos:

```
$idade              =  46;
$pagamentoDoMes     =  2175.14;
$telefone           =  '30909017';
$temContaCorrente   =  false;
$nome               =  "Maria das Graças";
$novoNome           =  $nome;
```

Tipos de dados



Inteiros

- Representa quaisquer números inteiros, positivos ou negativos. Admite números nos sistemas decimal, octal e hexadecimal;
- Faixa de valores aceitáveis dependente da plataforma.

Sistema	Iniciar com	Valor válido	Valor inválido
octal	***0***	***014***	***09*** – 9 não existe no sistema octal
hexadecimal	***0x ou 0X***	***0x90af3***	***0x4m67*** – m não existe no sistema hexadecimal

Faixa de valores aceitáveis	
Sistemas de 32 bits	**Sistemas de 64 bits**
–2.147.483.648 até +2.147.483.647	***–9.223.372.036.854.775.808 até 9.223.372.036.854.775.807***

Reais

- Representam quaisquer números que tenham uma parte decimal. O separador é o **(.)**
- Admite representação por meio de notação científica;

  `$realEmNotacaoCientifica = 1.02E+05,` que equivale a 102000.0

  `$outroRealEmNotacaoCientifica = 9.045E-03`, que é igual a 0.009045
- Sua faixa de valores, como os inteiros, depende de detalhes do sistema. Geralmente, fica entre **1,7E–308** e **1,7E+308**
- A precisão é de **15 casas decimais**;
- Números reais têm representação imprecisa. Por exemplo, o valor 3,5 de fato, é representado como 3,4999999999999. Isso pode conduzir a resultados não exatos em cálculos matemáticos;
- Também, devido à imprecisão, evite fazer comparações lógicas (igual, diferente, menor, maior, etc...) com números reais.

Lógico ou booleano

- Tipo de dado que representa um de dois estados possíveis somente;
- Admite os valores **true** ou **false**. Estes valores podem ser escritos em maiúsculo ou minúsculo, mas a norma dita a escrita toda em minúsculo. Exemplo:

```
$temCarro        =  TRUE;
$estaFrio        =  false;
$vouViajar       =  FalSE;
$vouPassarEmPHP  =  tRuE;
```

Equivalentes lógicos do PHP

- Para o PHP, qualquer situação abaixo representa um valor equivalente a **FALSE**:
  - A palavra-chave false;
  - O inteiro 0;
  - O real 0.0;
  - O string vazio '';
  - O string vazio "";
  - O string '0' ou "0";
  - Um vetor com nenhum elemento;
  - Um objeto com nenhuma propriedade ou método;
  - O valor NULL.
- Qualquer valor diferente destes é tratado pelo PHP como **TRUE**.

Strings

- Sequência de caracteres com tamanho ilimitado. São escritos dentro de aspas ou apóstrofos.

- Variáveis dentro de aspas são interpoladas. Dentro de apóstrofos, não são. Veja:

```
$nome = 'Joana';

echo '$nome'; //o navegador mostra $nome

echo "$nome"; //o navegador mostra Joana
```

- E se quisermos escrever aspas ou apóstrofo como parte da string? Exemplo:

```
echo "...e Deus disse: "Faça-se a luz!""; //erro
```

- Usamos o caractere de escape \

Caracteres de escape

Sequência de escape enviada pelo echo	O que o navegador recebe
\"	"
\'	'
\n	Nova linha
\r	Retorno de carro
\t	Um espaço de tabulação
\\	\
\$	$

```
$caminho =  'C:\\WINDOWS\\SYSTEM\\Tim O\'Reilly';
echo $caminho; //o navegador mostrará C:\WINDOWS\SYSTEM\TIM O'Reilly
```

O tipo NULL

- Uma variável criada mas não inicializada contém o valor **NULL**;
- Podemos atribuir null diretamente à variável: `$texto = null;`
- NULL significa uma variável com nenhum dado válido, mas, ainda assim armazenando NULL. Null pode ser escrito independentemente usando-se maiúsculas ou minúsculas;
- Se o PHP escrever na página web uma variável com o valor null, o navegador mostrará um **espaço em branco**;

☞**NOTA:**

- O tipo de dado recurso (resource) é manipulado e criado internamente pelo PHP para controle de elementos externos ao próprio interpretador, como conexão a um banco de dados, acesso a um arquivo em disco, etc... Não utilizaremos este tipo;
- Vetores e matrizes serão cobertos detalhadamente mais a frente neste curso.

Variáveis de variável

- São variáveis que fazem referência ao valor de uma variável cujo nome está armazenado em outra variável. Veja:

```
$pessoa = 'estudante';

$$pessoa = 'João';  (variável de variável)
```

- Depois da segunda declaração, o PHP cria, automaticamente , uma variável de nome **$estudante** cujo valor é **'João'**;

- Esta técnica será muito útil, por exemplo, para facilitar o recebimento de dados de um formulário que apresente muitos campos de informação. Veremos exemplos práticos aplicando este método no decorrer do curso.

Escopo de variáveis

- Escopo de variáveis está relacionado ao local onde uma variável foi criada e quais partes do programa podem ou não ter acesso ao conteúdo da variável. Em PHP, temos 4 tipos de escopo: local, global, estático e parâmetros de funções. Não utilizaremos, no nosso curso, o escopo estático;
- **Escopo global:** em PHP, variáveis declaradas fora de um função são globais a todo o script, mas não são visíveis dentro da própria função. Uma variável global criada em um seção do documento é visível em qualquer seção subsequente;
- **Escopo local:** qualquer variável criada dentro de uma função só existe e tem seu valor disponível dentro da própria função. Acessos externos à variável não são possíveis. Uma variável local é extinta e perde todo seu conteúdo logo que uma função termina sua execução;
- **Parâmetros de funções** também são locais e só estão disponíveis dentro da própria função. Assim como variáveis locais, qualquer parâmetro é extinto logo que a função termina.

Escopo de variáveis - esquema

```php
<?php
 //seção 1
 $idade = 46; //esta variável é global e pode ser alcançada de qualquer lugar deste documento, mesmo em outras seções de PHP abaixo destas, exceto de dentro de uma função
 ?>
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="utf-8">
  <title>  </title>
</head>

<body>
 <?php
 //seção 2 do PHP
 echo $idade; //mostra 46

 //tentar acessar aqui $soma e $valor resulta em erro

 function somaIdade($valor)//$valor é um parâmetro e só está acessível dentro da função. Nenhum outro lugar deste documento pode utilizar $valor
  {
  echo $valor; //parâmetro de função é local. Mostra 46
  $soma = 0;
  echo $soma;

  //tentar acessar a variável $idade aqui gera erro, pois ela é global

  //neste momento, a função irá terminar e tanto a variável $soma quanto o parâmetro $valor serão destruídos e seus conteúdos perdidos
  }

 //tentar acessar $soma e $valor aqui também dá erro
 somaIdade($idade); //chamada de uma função
 ?>
</body>
</html>
```

Expressões e operadores

- Uma expressão é um trecho de código em PHP que produz um valor;
- As expressões mais simples são as variáveis e os valores literais atribuídos a uma variável. Exemplo:

```
$soma = 8 + 6;    //esta linha é uma expressão do PHP

if($idade <= 45)  //o conteúdo dentro do parêntese também é uma expressão
```

- Um operador toma um, dois ou mais valores e realiza uma determinada operação com os mesmos;
- Dependendo do que um operador faz, podemos ter:
  - **Operadores aritméticos** – realizam operações matemáticas
  - **Operadores relacionais** – comparam grandezas
  - **Operadores lógicos ou booleanos** – manipulam expressões lógicas

Precedência de operadores

- Quando temos, numa expressão, diversos operadores de tipos diferentes, o PHP deve escolher quais operadores e operandos serão avaliados primeiro. Para isso, existe a precedência de operadores. Suponha a expressão abaixo:

```
$operador = 8 + 3 * 4 < 5 / 12 && (7 - $aumento++);
```

- O PHP deve avaliar esta expressão e, ao final, atribuir um valor à variável $operador;
- Isto é possível com a aplicação de duas simples regras:
  - *Multiplicação, divisão e resto vêm antes de soma e subtração;*
  - *Utilize parênteses para qualquer situação adicional que aparecer;*
- O uso de parênteses muda a precedência natural de um operador;
- Na expressão acima, usando parênteses, teríamos:

```
$operador = ((8 + 3 * 4) < (5 / 12)) && (7 -
$aumento++); // o resultado é verdadeiro
```

Tabela de precedência – da maior para a menor

Precedência	Operador	Descrição
1	- ++ –	Inversor de sinal, autoincremento, autodecremento
2	!	Inversor lógico
3	* / %	Multiplicação, divisão, resto
4	+ - .	Soma, subtração, concatenação de string
5	> < >= <=	Maior, menor, maior ou igual, menor ou igual
6	== !=	Igual, diferente
7	&&	E lógico (and)
8	||	OU lógico (or)
9	= += -= *= /= %=	Operadores compactos de atribuição
10	AND	E lógico (and)
11	OR	OU lógico (or)

☞**NOTA:** numa mesma linha, operadores têm precedência igual

Associatividade de operadores

- Quando, numa expressão, os operadores têm a mesma precedência, o PHP começa a avaliar o resultado da expressão da esquerda para a direita. Use parênteses para alterar esta ordem natural. Veja:

```
$resultado = 2 / 2 * 2;
```
//resulta em 2, divisão primeiro

```
$resultado = 2 * 2 /2;
```
//resulta 2, a multiplicação é feita antes

```
$resultado = 2 * (2 / 2);
```
//resulta 2, a divisão é feita antes

```
$resultado = 2 * (2 + 2);
```
// 8, pois parêntese altera a ordem

Operadores aritméticos ou matemáticos

- Realizam uma operação matemática sobre os operandos.

Operador	Descrição
*	Multiplicação
/	Divisão – pode resultar um inteiro (4/2) ou um real (2/4)
+	Adição
-	Subtração
%	Resto de uma divisão inteira (12 % 5 é igual a 2)
-	Oposto – equivale a inverter o sinal do número

Conversão de tipos implícita

- Quando o PHP precisa manipular operandos com tipos de dados diferentes, ele tenta converter ambos para o mesmo tipo. Para **operadores aritméticos**, a conversão segue as regras abaixo:

Tipo do primeiro operando	Tipo do segundo operando	Conversão aplicada
inteiro	***real***	O inteiro é convertido para real
inteiro	***string***	O string é convertido para um número. Se o número, depois da conversão, for um real, o outro inteiro também é convertido para real
real	***string***	O string é convertido para real

☞**NOTA:** nas situações onde o PHP não consegue converter uma string para um inteiro ou real equivalente (tipo "ab567"), o PHP usa o valor 0 no lugar do string problemático. Note que um string do tipo '567.1abf3' é ainda convertido pelo PHP como o real 567.1

Operador de concatenação de strings

- O PHP junta uma string com outra por meio do operador de concatenação (.). Se este operador for usado com valores inteiros ou reais, estes valores são sempre transformados para um **string equivalente**. O retorno deste operador é sempre um tipo string. Veja:

```
$a = 37;

$b = "Faltam ";

$c = ' dias para a festa';

$concatena = $b . $a . $c; //'Faltam 37 dias para a festa'
```

Operadores de autoincremento e autodecremento

- O autoincremento **++** aumenta o valor da variável em uma unidade. Equivale a **$var = $var + 1**.

- O autodecremento **--** diminui o valor em uma unidade. Equivale a **$var = $var – 1**;

- O operador pode ser colocado antes do valor (pré-incremento ou pré-decremento) ou depois (pós-incremento ou pós decremento);

- Colocar antes ou depois produz resultados diferentes. Se colocado antes, ele retorna o novo valor do operando já modificado (incrementado ou decrementado). Se colocado depois, ele retorna o valor original do operando, sem alteração. Exemplos:

```
$a = 5;

$b = $a++ / 2;      // no final, $b vale 2.5 e $a vale 6

$b = ++$a / 2;      //no final da operação, $b é igual a 3 e $a vale 6
```

Operadores relacionais

- Realizam comparações entre os operandos. Sempre que uma expressão contiver um operador relacional, o resultado final é **true** ou **false**;
- Operandos na comparação podem ser ambos numéricos, ambos string ou um número e um string;

Operador	Descrição
==	Igualdade (não compara tipos)
===	Identidade (compara se os valores e tipos dos operandos são iguais)
!= ou <>	Desigualdade (não compara tipos – apenas se os valores são diferentes)
!==	Desigualdade absoluta (verifica se os valores e os tipos são diferentes)
>	Maior que
<	Menor que
<=	Menor que ou igual a
>=	Maior que ou igual a

Tipo de comparação aplicado pelos operadores relacionais

Primeiro operando	Segundo operando	Comparação
número	número	numérica
string numérico	string numérico	numérica
string numérico	número	numérica
string numérico	string não numérico	numérica
string não numérico	número	alfabética
string não numérico	string não numérico	alfabética

```
8.7 > 15 (false)
'8.7' < '15' (true)
'8.7' == 15 (false)
"45" != 45 (false)
'abc45' < 45 (false) – código ASCII de 'a' é 97 e código ASCII de '4' é 42
'maria' > 'ana' (true) – ASCII de 'm' é 109 e ASCII de a é 97
```

Operadores lógicos

- Estes operadores atuam sobre expressões lógicas e, assim como os relacionais, sempre retornam um valor booleano. Qualquer expressão em PHP, por mais complexa que seja, e que apresentar, pelo menos, um operador lógico, terá como resultado final sempre o valor ***true*** ou ***false***.

Operador	Significado	Resultado
&& ou and	E lógico	Retorna true se TODOS os operandos forem true. Caso contrário, retorna false
|| ou or	OU lógico	Retorna true se, NO MÍNIMO, UM operando for true. Se todos forem FALSE, retorna false
!	Negação	Inverte o valor lógico do operando, isto é, se o operando era true antes de aplicar o operador, depois da aplicação ele passa a ter o valor false, e vice-e-versa

Operadores de atribuição combinados

Operador	Significado	Exemplo	Associação equivalente
+=	Soma e atribui	$a += 8;	$a = $a + 8;
-=	Subtrai e atribui	$a -= 8;	$a = $a – 8;
*=	Multiplica e atribui	$a *= 8;	$a = $a * 8;
/=	Divide e atribui	$a /= 8	$a = $a / 8;
.=	Concatena e atribui	$a .= '8'	$a = $a . '8';
%=	Calcula o resto e atribui	$a %= 8;	$a = $a % 8;