

2016.2

Introdução ao PHP

Programação Web com Software Livre

1 - Histórico

- Criação em 1994 por Rasmus Lerdorf;
- 1995 - PHP/FI (Personal Home Pages/Forms Interpreter);



Início da contribuição dos internautas

- 1997 – Segunda versão do PHP;



Neste momento, aproximadamente 50 mil domínios ou 1% da internet já utilizava PHP.

- Andi Gustmans e Zeev Suraski → Começaram a contribuir com o código fonte do PHP e em 1998 SURTIU o **PHP3**

1 - Histórico

- PHP3:
 - Extensibilidade;
 - Conexão com banco de dados;
 - Novos protocolos;
 - API que possibilitava a criação de novos módulos;
 - Alteração do significado da sigla para Hypertext Preprocessor;
 - Integração natural com o HTML;

Nesta época cerca de 10% dos domínios da internet já utilizava o PHP

1 - Histórico

- Em 2000 → PHP4: Melhoria do núcleo do PHP por Zeev e Andi.
 - Batizado como Mecanismo Zend (Zeev + Andi);
 - Variáveis de Sessão;
 - Suporte a diversos servidores Web;
 - Conexões com muitos bancos de dados;
 - Disponível para Sistemas Operacionais Linux e Windows;
 - Aderência a programação orientada a objetos.

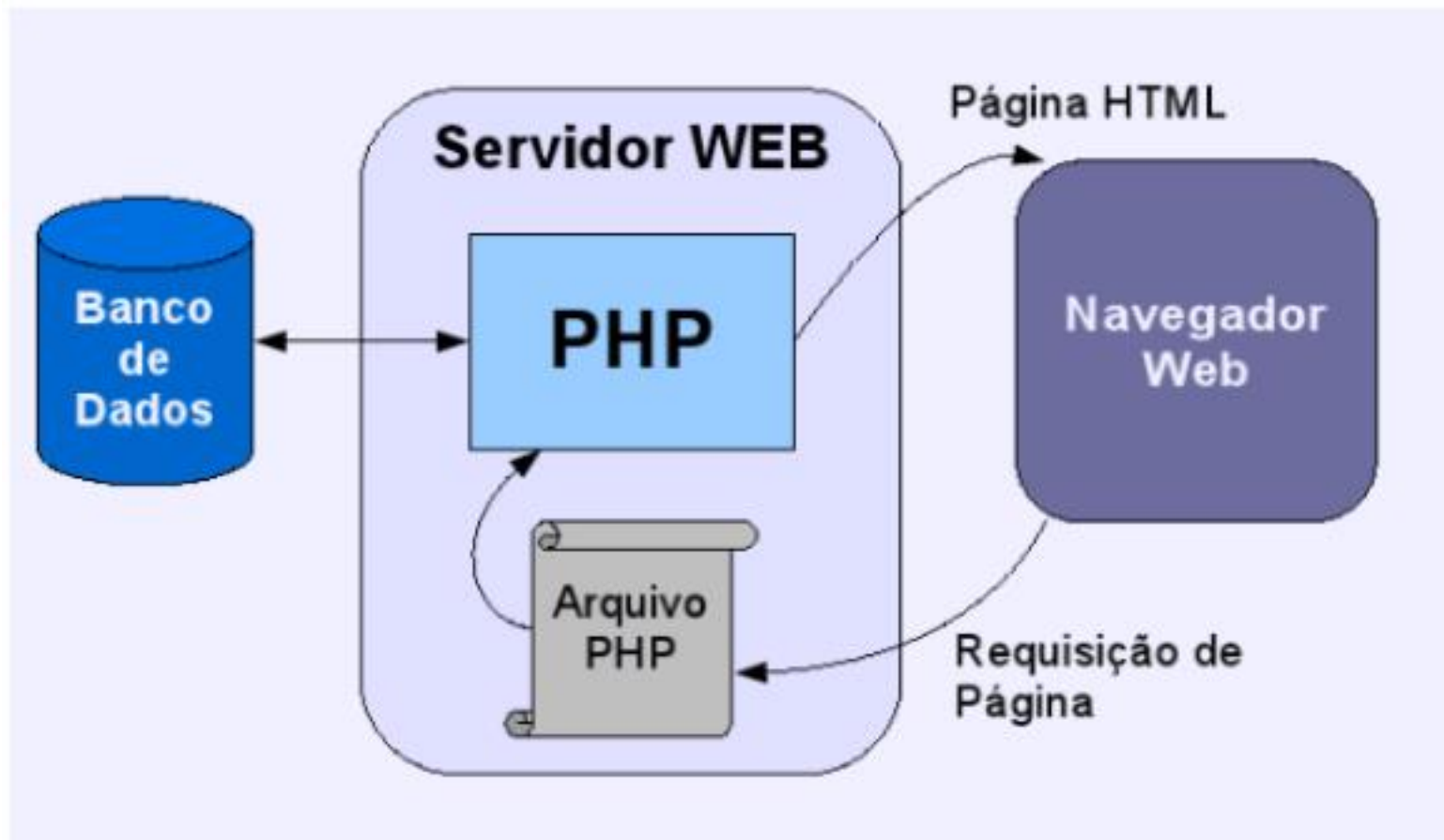
1 - Histórico

- Em 2004 → PHP5
 - Programação Orientada a Objetos;
 - Otimização e surgimento de funções e bibliotecas para trabalhar com banco de dados (Ex: MySQLi, PDO);
 - Nova API para trabalhar com XML;
 - Suporte a SOAP.

2 – O que é PHP?

- O PHP é uma linguagem de programação interpretada de código aberto publicado sob a PHP License;
- **Características:**
 - Velocidade;
 - Portabilidade - independência de plataforma - escreva uma vez, rode em qualquer lugar;
 - Tipagem dinâmica;
 - Sintaxe similar a C/C++ e o Perl;
 - Open-source
 - Server-side (O cliente manda o pedido e o servidor responde em página HTML)

3 – Como funciona?



3 – Extensões de arquivos

►.php

- Arquivo PHP contendo um programa;

►class.php

- Arquivo PHP contendo uma classe;

►inc.php

- Arquivo a ser incluído, pode incluir constantes ou configurações;

4 – Delimitadores de Código

- O código de um programa do php deve ser delimitado:

```
<?php
//código ;
//código ;
//código ;
?>
```

→ Todos os comandos são delimitado por ponto-e-vírgula ;

4 – Delimitadores de Código

```
<?php  
comandos  
?>
```

```
<script language="php">  
comandos  
</script>
```

```
<?  
comandos  
?>
```

```
<%  
comandos  
%>
```

5 – Comentários

- Para comentar uma única linha de código

```

// echo "a" ;
# echo "a" ;

```

- Para comentar várias linhas de código

```

/* echo "a" ;
echo "b" ;
*/

```

6 – Variáveis (Regras e boas práticas)

- Toda variável em PHP tem seu nome composto pelo caractere \$ e uma string, que deve iniciar por uma **letra** ou o **caractere** “_”;
- **Nunca** utilize espaços no meio do identificador da variável;
- **Nunca** utilize caracteres especiais: ! @ # \$ % ^ & * / [] { }
- Nomes das variáveis devem ser significativos e transmitir a ideia de seu conteúdo;
- Evite criar variáveis com mais de 20 caracteres para manter a clareza do código;
- Utilize preferencialmente palavras em minúsculo (separadas por _) ou somente a primeira letra em maiúscula, caso de ter mais de uma palavra.
- O PHP é case sensitive

6 – Variáveis (Regras e boas práticas)

O PHP 5 oferece um outro meio de atribuir valores a variáveis: a atribuição por referência. Isto significa que a nova variável simplesmente referencia (em outras palavras, "torna-se um apelido para" ou "aponta para") a variável original.

Exemplo:

```
<?php
$item1 = 'Banana'; // Atribui o valor 'Banana' a variável $item1
$item2 = &$item1; // Referencia $item1 através de $item2.
$item2 = "O produto é $item2"; // Altera $item2...
echo $item2.'<br>';
echo $item1; // $item1 é alterado também.
?>
```

6 – Variáveis (Regras e boas práticas)

Quais as variáveis abaixo possuem declaração válida em PHP?

- ▶ \$5cliente
- ▶ 123
- ▶ \$computador
- ▶ \$#estabilizador
- ▶ \$cod Cliente
- ▶ \$_cliente
- ▶ \$computadorSemEstabilizadorDeRede
- ▶ \$idPessoa

7 – Comandos de saída

- **echo**

Imprime uma ou mais variáveis no console.

Exemplo: <?php
 echo 'a', 'b', 'c';
 ?>

- **print**

Imprime uma string no console.

Exemplo: <?php
 print('abc');
 ?>

7 – Comandos de saída

Outra diferença em relação ao **echo** e o **print** é que o **print** sempre retorna o valor inteiro 1, enquanto o **echo** não retorna nada.

Exemplo:

```
$minhavariavel = print "aeiou";
```

Isto irá retornar '**aeiou**' e colocará em minha variável \$minhavariavel o valor **1**

```
$minhavariavel = echo "aeiou"; (incorreto)
```

O correto seria:

```
echo "aeiou";
```


7 – Comandos de saída

- **var_dump:**

- A função `var_dump` exibe informações estruturadas sobre as variáveis / expressões, incluindo o tipo e o valor.
- Muito comum para realizar debug.
- Se o parâmetro for um objeto, ele imprimirá todos os seus atributos; Se for um array, imprimirá o conteúdo das respectivas posições, com seus tipos de dados.

Exemplo:

```
<?php
```

```
$array = array('123',123);  
var_dump($array);
```

```
?>
```

Resultará em:

```
array(2) {  
    [0]=>  
    string(3) "123"  
    [1]=>  
    int(123)  
}
```

7 – Comandos de saída

- **print_r:**

- O `print_r ()` Imprime o conteúdo de uma variável de forma explicativa, assim como no `var_dump()`, mas com um formato mais legível para o programador, com os conteúdos alinhados e suprimindo os tipos de dados.

Exemplo:

```
<?php  
  
$array = array('123',123);  
print_r($array);  
  
?>
```

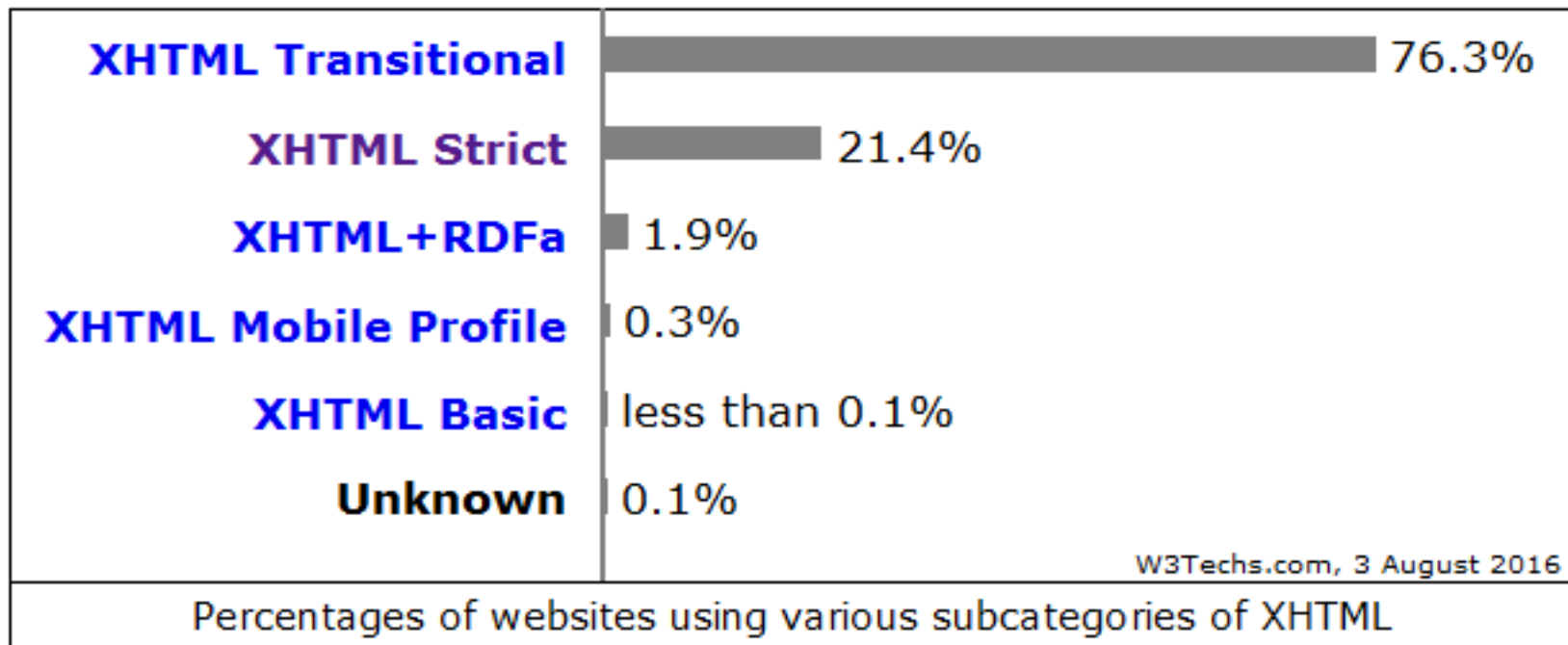
Resultará em:

```
Array  
(  
    [0] => 123  
    [1] => 123  
)
```

8 – XHTML

- XHTML é a sigla em inglês para EXtensible HyperText Markup Language que em tradução livre resulta em Linguagem Extensível para Marcação de Hipertexto, uma aplicação XML, escrita para substituir o HTML.
- As tags e atributos da XHTML foram criadas ("inventadas") aproveitando-se as nossas conhecidas tags e atributos da HTML 4.01 e suas regras.
- Escrevemos um código XML, onde as tags e atributos já estão definidos .
- XHTML é uma recomendação separada; a W3C continua a recomendar o uso de XHTML 1.1 para publicação na web, assim como o HTML5.

8 – XHTML



8 – XHTML

➤ As diferenças entre XHTML e HTML

1 - Todas as tags devem ser escritas com letras minúsculas

Errado:

```
<DIV><P>Aqui um texto</P></DIV>
```

Certo:

```
<div><p>Aqui um texto</p></div>
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

2 - As tags devem estar convenientemente aninhadas

Errado:

```
<div><em><p>Aqui um texto negrito</em></p></div>
```

Certo:

```
<div><em><p>Aqui um texto negrito</p></em></div>
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

3 - Os documentos devem ser bem formados

```
<html>  
<head>  
...  
</head>  
<body>  
...  
</body>  
</html>
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

4 - O uso de tags de fechamento é obrigatório

Errado:

`<p>Um parágrafo.<p>Outro parágrafo.`

Certo:

`<p>Um parágrafo.</p><p>Outro parágrafo.</p>`

8 – XHTML

➤ As diferenças entre XHTML e HTML

5 - Elementos vazios devem ser fechados

Errado: Elementos vazios sem terminação

```
<hr>
```

```

```

Certo: Elementos vazios com terminação

```
<br />
```

```
<hr />
```

```

```

8 – XHTML

➤ As diferenças entre XHTML e HTML

6 - Diferenças para a sintaxe dos atributos

Assim como as tags, os atributos também são sensíveis ao tamanho de caixa e então deve-se escrever nomes de atributos em minúsculas;

Errado:

```
<td ROWSPAN="3">
```

Certo:

```
<td rowspan="3">
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

7 - Valores de atributos

Os valores de atributos devem estar entre "aspas duplas " ou 'aspas simples';

Errado:

`<td rowspan=3>`

Certo:

`<td rowspan="3">` ou `<td rowspan='3'>`

8 – XHTML

➤ As diferenças entre XHTML e HTML

8 - Valores de atributos

Todos os atributos devem receber um valor;

Errado:

```
<input checked />
```

Certo:

```
<input checked="checked" />
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

8 - Valores de atributos

Todos os atributos devem receber um valor;

```
compact compact="compact"
checked checked="checked"
declare declare="declare"
readonly readonly="readonly"
disabled disabled="disabled"
selected selected="selected"
defer defer="defer"
ismap ismap="ismap"
nohref nohref="nohref"
noshade noshade="noshade"
nowrap nowrap="nowrap"
multiple multiple="multiple"
noresize noresize="noresize"
```

8 – XHTML

➤ As diferenças entre XHTML e HTML

9 - O atributo alt para imagens

*Em XHTML o uso do atributo **alt** para imagens é obrigatório;*

```

```

Se tratar-se de uma imagem decorativa pode-se usar o atributo **alt** vazio:

```

```

8 – XHTML

➤ As diferenças entre XHTML e HTML

10 - Códigos gerados por editores

Cuidado com os códigos gerados por editores!

Errado:

`onMouseOver=function()`

Certo:

`onmouseover=function()`

8 – XHTML

➤ As diferenças entre XHTML e HTML

11 - Caracter & (ampersand)

Codifique o & (e comercial)

Errado:

Comércio & Exportação

Certo:

Comércio & Exportação

8 – XHTML

➤ Elementos obrigatórios em um documento XHTML

É obrigatório a declaração do DOCTYPE assim como a existência dos elementos `<html>` `<head>` `<title>` e `<body>`

```
<!DOCTYPE bla..bla..bla>  
<html  
  xmlns="http://www.w3.org/1999/xhtml">  
  <head>  
    <title>Título do documento</title>  
  </head>  
  <body>  
    Conteúdo do documento  
  </body>  
</html>
```

8 – XHTML

➤ DOCTYPE

A DTD é usada pelas aplicações SGML (tais como HTML) para identificar as regras que se aplicam a linguagem de marcação usada no documento bem como o conjunto de elementos e entidades válidas naquela linguagem.

O DOCTYPE deve ser sempre a primeira declaração em um documento web.

8 – XHTML

➤ Os tipos de DOCTYPE para o XHTML

STRICT:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

TRANSITIONAL

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

FRAMESET

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

9 – Tipos de Dados

➤ O PHP suporta oito tipos de dados primitivos divididos em três grupos:

1 - Quatro tipos básicos, os dados escalares

integer

float (número de ponto flutuante, ou também double)

string

boolean

2 - Dois tipos compostos

array

object

3 - Dois tipos especiais:

resource

NULL

9 – Tipos de Dados

➤ O tipo **integer**, inteiro, no PHP

Um inteiro é qualquer numero sem decimais, positivo ou negativo. Englobando todos os números do conjunto Z (os números inteiros).

```
01 <?php
02
03 // Declaração de uma variavel como inteiro
04 $ano_nascimento = 1989;
05
06 ?>
07 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
08 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
09 <html xmlns="http://www.w3.org/1999/xhtml">
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12 <title>Tipo Inteiro no PHP</title>
13 </head>
14 <body>
15 <p>Eu nasci no ano de <?php echo $ano_nascimento; ?></p>
16 </body>
17 </html>
```

9 – Tipos de Dados

- O tipo **float** ou **double**, número de ponto flutuante, no PHP

Float, Double ou ainda números de ponto flutuante são os números reais.

```

01  <?php
02
03  // Declaração de uma variavel como float, armazenando o valor de pi
04  $pi = 3.14159265;
05
06  ?>
07  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
08  <html xmlns="http://www.w3.org/1999/xhtml">
09  <head>
10  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
11  <title>Tipo float, double ou número de ponto flutuante no PHP</title>
12  </head>
13
14  <body>
15  <p>O valor de pi é <?php echo $pi; ?></p>
16  </body>
17  </html>
    
```

9 – Tipos de Dados

➤ O tipo string no PHP

Uma string é uma série de caracteres, um texto por exemplo. Para declararmos strings podemos utilizar as aspas simples (apóstrofes) e as aspas duplas.

```
01 <?php
02
03 // Declaração de uma variavel com string com aspas simples
04 $texto = 'O PHP é uma linguagem server-side.';
05
06 ?>
07 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
08 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
09 <html xmlns="http://www.w3.org/1999/xhtml">
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12 <title>String com aspas simples no PHP</title>
13 </head>
14 <body>
15 <p><?php echo $texto; ?></p>
16 </body>
17 </html>
```

9 – Tipos de Dados

➤ O tipo string no PHP

A grande diferença entre as strings declaradas com aspas simples e as declaradas com aspas **duplas** está no fato de que as strings declaradas com aspas duplas interpretam as variáveis.

```

01  <?php
02  // Declaração de um produto
03  $produto = 'pizza';
04  // Declaração de uma variavel com string com aspas duplas
05  $texto = "Ele \"comprou\" uma $produto no Joey's.";
06
07  ?>
08  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
09  <html xmlns="http://www.w3.org/1999/xhtml" >
10  <head>
11  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12  <title>String com aspas duplas no PHP</title>
13  </head>
14
15  <body>
16  <p><?php echo $texto; ?></p>
17  </body>
18  </html>
    
```


9 – Tipos de Dados

- O tipo boolean, booleano, no PHP

O tipo booleano é muito simples pois aceita apenas os valores verdadeiro(TRUE) ou falso(FALSE).

```

1  <?php
2  // Declaração de uma variável booleana com o valor verdadeiro
3  $sim = TRUE;
4  // Declaração de uma variável booleana com o valor falso
5  $nao = FALSE;
6
7  ?>

```

10 – Operadores

➤ Informa ao PHP o que deve ser executado

◦Ex: Atribuir um valor a uma variável, realizar operações aritméticas, comparações de valores, ...

□ Tipos:

- Operadores Aritméticos
- Operadores Binários
- Operadores de Comparação
- Operadores de atribuição
- Operadores lógicos
- Operador ternário

10 – Operadores

➤ Aritméticos

Exemplo	Nome	Resultado
$-\$a$	Negação	Oposto de $\$a$.
$\$a + \b	Adição	Soma de $\$a$ e $\$b$.
$\$a - \b	Subtração	Diferença entre $\$a$ e $\$b$.
$\$a * \b	Multiplicação	Produto de $\$a$ e $\$b$.
$\$a / \b	Divisão	Quociente de $\$a$ por $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.

10 – Operadores

➤ Aritméticos

Exemplo	Nome	Efeito
-\$a	Troca Sinal	Trocar o sinal de \$a
++\$a	Pré-incremento	Incrementa \$a em um, e então retorna \$a.
\$a++	Pós-incremento	Retorna \$a, e então incrementa \$a em um.
--\$a	Pré-decremento	Decrementa \$a em um, e então retorna \$a.
\$a--	Pós-decremento	Retorna \$a, e então decrementa \$a em um.

10 – Operadores

➤ Aritméticos

```
echo "<h3>Pós-incremento</h3>";
$a = 5;
echo "Deve ser 5: " . $a++ . "<br />\n";
echo "Deve ser 6: " . $a . "<br />\n";
```

```
echo "<h3>Pré-incremento</h3>";
$a = 5;
echo "Deve ser 6: " . ++$a . "<br />\n";
echo "Deve ser 6: " . $a . "<br />\n";
```

```
echo "<h3>Pós-decremento</h3>";
$a = 5;
echo "Deve ser 5: " . $a-- . "<br />\n";
echo "Deve ser 4: " . $a . "<br />\n";
```

```
echo "<h3>Pré-decremento</h3>";
$a = 5;
echo "Deve ser 4: " . --$a . "<br />\n";
echo "Deve ser 4: " . $a . "<br />\n";
```

10 – Operadores

➤ Binários

Trabalham diretamente com os bits

Exemplo	Nome	Resultado
$\$a \& \b	E	Os bits que estão ativos tanto em $\$a$ quanto em $\$b$ são ativados.
$\$a \b	OU	Os bits que estão ativos em $\$a$ ou em $\$b$ são ativados.
$\$a \wedge \b	XOR	Os bits que estão ativos em $\$a$ ou em $\$b$, mas não em ambos, são ativados.
$\sim \$a$	NÃO	Os bits que estão ativos em $\$a$ não são ativados, e vice-versa.
$\$a \ll \b	Deslocamento à esquerda	Desloca os bits de $\$a$ $\$b$ passos para a esquerda (cada passo significa "multiplica por dois")
$\$a \gg \b	Deslocamento à direita	Desloca os bits de $\$a$ $\$b$ passos para a direita (cada passo significa "divide por dois")

10 – Operadores

➤ Atribuição

Exemplo	Efeito
<code>\$a = \$b</code>	<code>\$a</code> recebe o valor de <code>\$b</code>
<code>\$a += \$b</code>	Equivalente a <code>\$a = \$a + \$b</code>
<code>\$a -= \$b</code>	Equivalente a <code>\$a = \$a - \$b</code>
<code>\$a *= \$b</code>	Equivalente a <code>\$a = \$a * \$b</code>
<code>\$a /= \$b</code>	Equivalente a <code>\$a = \$a / \$b</code>
<code>\$a %= \$b</code>	Equivalente a <code>\$a = \$a % \$b</code>
<code>\$a .= \$b</code>	Concatenação: equivalente a <code>\$a = \$a . \$b</code>
<code>\$a &= \$b</code>	Equivalente a <code>\$a = \$a & \$b</code>
<code>\$a = \$b</code>	Equivalente a <code>\$a = \$a \$b</code>
<code>\$a ^= \$b</code>	Equivalente a <code>\$a = \$a ^ \$b</code>
<code>\$a <<= \$b</code>	Equivalente a <code>\$a = \$a << \$b</code>
<code>\$a >>= \$b</code>	Equivalente a <code>\$a = \$a >> \$b</code>

10 – Operadores

➤ Comparação

Exemplo	Nome	Resultado
\$a == \$b	Igual	Verdadeiro (TRUE) se \$a é igual a \$b.
\$a === \$b	Idêntico	Verdadeiro (TRUE) se \$a é igual a \$b, e eles são do mesmo tipo (introduzido no PHP4).
\$a != \$b	Diferente	Verdadeiro se \$a não é igual a \$b.
\$a <> \$b	Diferente	Verdadeiro se \$a não é igual a \$b.
\$a !== \$b	Não idêntico	Verdadeiro se \$a não é igual a \$b, ou eles não são do mesmo tipo (introduzido no PHP4).
\$a < \$b	Menor que	Verdadeiro se \$a é estritamente menor que \$b.
\$a > \$b	Maior que	Verdadeiro se \$a é estritamente maior que \$b.
\$a <= \$b	Menor ou igual	Verdadeiro se \$a é menor ou igual a \$b.
\$a >= \$b	Maior ou igual	Verdadeiro se \$a é maior ou igual a \$b.

10 – Operadores

➤ Comparação

Observação:

Uma String é convertida para número quando comparada com um número.

Exemplos:

```
var_dump(0 == "a"); // 0 == 0 -> true  
var_dump("1" == "01"); // 1 == 1 -> true  
var_dump("1" == "1e0"); // 1 == 1 -> true
```

10 – Operadores

➤ Lógicos

São aqueles que retornam o valor verdadeiro ou falso

Exemplo	Nome	Resultado
\$a and \$b	E	Verdadeiro (TRUE) se tanto \$a quanto \$b são verdadeiros.
\$a or \$b	OU	Verdadeiro se \$a ou \$b são verdadeiros.
\$a xor \$b	XOR	Verdadeiro se \$a ou \$b são verdadeiros, mas não ambos.
! \$a	NÃO	Verdadeiro se \$a não é verdadeiro.
\$a && \$b	E	Verdadeiro se tanto \$a quanto \$b são verdadeiros.
\$a \$b	OU	Verdadeiro se \$a ou \$b são verdadeiros.

10 – Operadores

➤ Precedência de operadores

Operador	Informação adicional
++ --	incremento/decremento
!	lógico
* / %	aritmético
+ - .	aritmético e string
<< >>	Bit-a-bit
< <= > >= <>	comparação
== != === !==	comparação
&	Bit-a-bit e referências
^	Bit-a-bit
	Bit-a-bit
&&	lógico
	lógico
? :	ternário
= += -= *= /= .= %= &= = ^=	atribuição
<<= >>=	
and	lógico
xor	lógico
Or	lógico

11 – Obter Dados de Formulário

- Utilizado para aumentar a interatividade fazendo comunicação entre o usuário e o site

Exemplo:

```
<form =>
  <p>Digite seu e-mail: <input type="text" name="email" size="20"></p>
  <p><input type="submit" value="Enviar!" name="enviar"></p>
</form>
```



As informações vão ser perdidas, pois o navegador não sabe o que fazer com elas

11 – Obter Dados de Formulário

- Utilizado para aumentar a interatividade fazendo comunicação entre o usuário e o site

Exemplo:

Para o tornar útil podemos usar a opção action, informando ao navegador para aonde enviar a informações para serem processadas

```
<form action="recebe_dados.php">  
  <p>Digite seu e-mail: <input type="text" name="email" size="20"></p>  
  <p><input type="submit" value="Enviar!" name="enviar"></p>  
</form>
```

11 – Obter Dados de Formulário

➤ TAG *input*

- Define um campo de entrada
- Acompanhado de algumas opções:

Opção	Descrição
name	Informa qual o nome do campo
value	Informa um valor padrão para o campo
size	Informa o tamanho do campo exibido na tela
maxlength	Informa o número máximo de caracteres do campo
type	Informa qual é o tipo do campo de entrada de dados

11 – Obter Dados de Formulário

➤ TAG *input*

Os possíveis valores da opção **type**:

Valor	Descrição
text	Mostra uma caixa de texto de uma linha que permite a entrada de valores numéricos ou alfanuméricos
password	Usado na digitação de senhas, pois camufla qualquer caractere digitado com (*) mas a informação é enviada normalmente
hidden	Campo escondido, não aparece na tela. Usado para passar informações aos programas que recebem os dados. Muito útil
select	Uma lista de seleção (ou <i>drop-down</i>)
checkbox	Uma caixa de seleção, que pode ser marcada ou desmarcada
radio	Botões de seleção, em que só uma opção é escolhida entre várias

11 – Obter Dados de Formulário

➤ TAG *input*

Os possíveis valores da opção **type**:

Valor	Descrição (Continuação)
textarea	Caixa de texto com várias linhas
file	Permite o envio de arquivos
submit	Botão que aciona o envio dos dados dos formulários
image	Mesma função <i>submit</i> , mas uma imagem substitue o botão tradicional
reset	Limpa todos os campos de um formulário e retorna o valor-padrão (se existir)

11 – Obter Dados de Formulário

➤ Métodos

Existem dois métodos de passagem de parâmetros :

- GET
- POST

Exemplo:

```
<form action="processa.php" method="POST">
```

11 – Obter Dados de Formulário

➤ GET

- Método padrão de envio de dados
- Se não for especificado o método na *tag action*, GET é assumido pelo PHP
- Os dados são enviados juntamente com o nome da página (URL) para o envio de dados

11 – Obter Dados de Formulário

➤ GET

Exemplo:

```
<form action="recebe_dados.php">  
  <p>Digite seu nome: <input type="text" name="nome" size="30"></p>  
  <p>Digite seu e-mail: <input type="text" name="idade" size="3"></p>  
  <p><input type="submit" value="Enviar!" name="enviar"></p>  
</form>
```

http://www.seusistema.com.br/recebe_dados.php?nome=Joaquim&idade=20

? - representa o início da cadeia de variáveis

& - identifica o início de uma nova variável

= - separa as variáveis dos seus respectivos valores

11 – Obter Dados de Formulário

➤ POST

Exemplo:

```
<form action="recebe_dados.php" method="POST">  
  <p>Digite seu nome: <input type="text" name="nome" size="30"></p>  
  <p>Digite seu e-mail: <input type="text" name="idade" size="3"></p>  
  <p><input type="submit" value="Enviar!" name="enviar"></p>  
</form>
```

Diferente do GET, o POST envia os dados por meio do corpo da mensagem encaminhada ao servidor

11 – Obter Dados de Formulário

- A maneira mais comum e segura de acessar os dados recebidos é usar os arrays superglobais predefinidos pelo php:

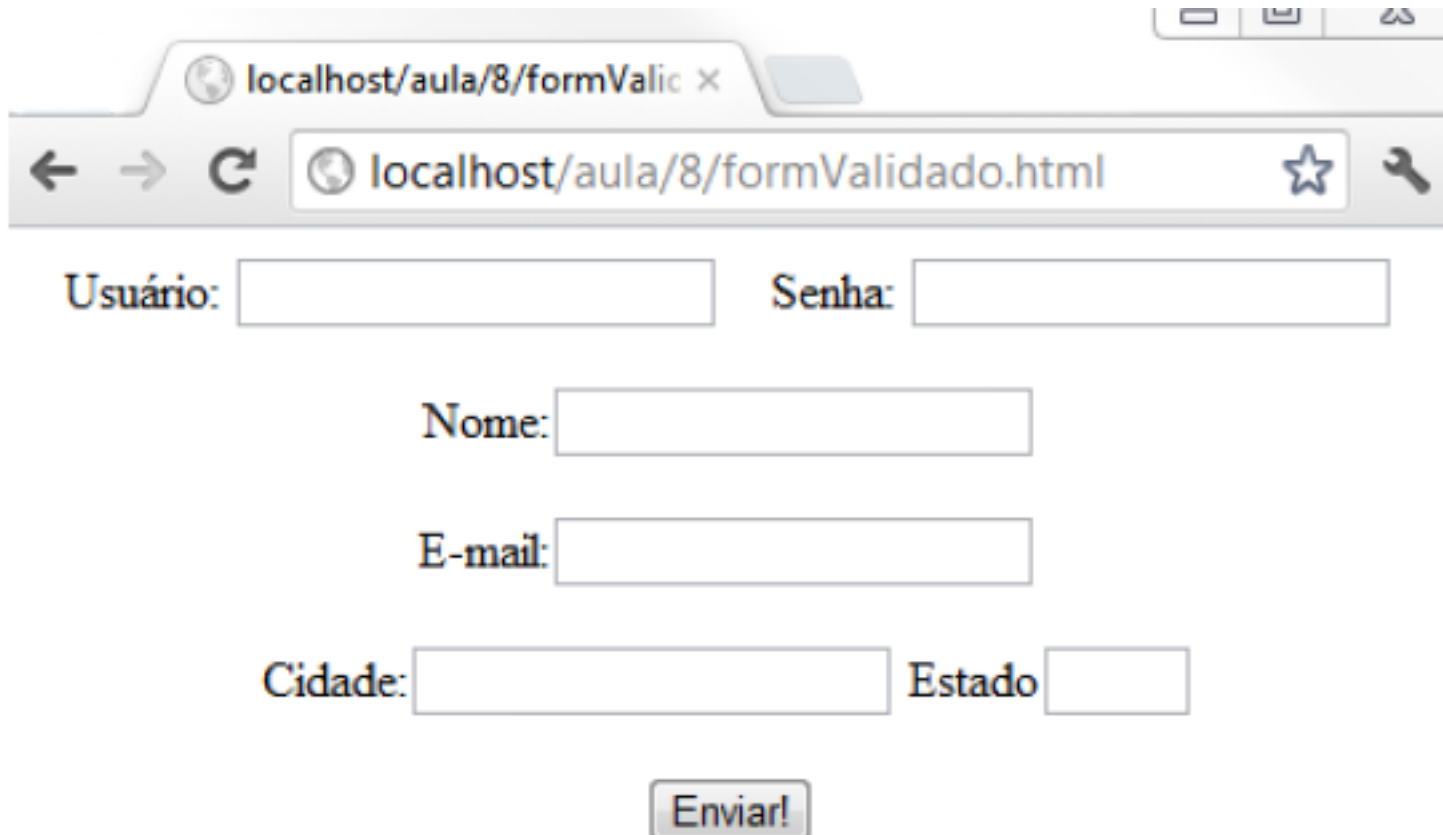
`$_GET` e `$_POST`

Os nome dos campos são usado como chaves associativas:

`$_GET["nome"]` ou `$_POST["idade"]`

11 – Obter Dados de Formulário

➤ Exemplo



localhost/aula/8/formValidado.html

Usuário: Senha:

Nome:

E-mail:

Cidade: Estado:

11 – Obter Dados de Formulário

Exemplo

```
<form action="recebe_dados.php" method="POST">  
    <div align="center"><center>  
        <p>Usuário: <input type="text" name="usuario" size="20">      </p>  
  
        Senha: <input type="text" name="senha" size="20"></p>  
        <p>Nome:<input type="text" name="nome" size="20"></p>  
        <p>E-mail:<input type="text" name="email" size="20"></p>  
        <p>Cidade:<input type="text" name="cidade" size="20">  
        Estado<input type="text" name="estado" size="2" maxlength="2"></p>  
  
        <p><input type="submit" value="Enviar!" name="enviar"></p>  
    </center></div>  
</form>
```

11 – Obter Dados de Formulário

➤ Exemplo

Recebe_dados.php – Recebe e processa os dados da página

```
$usuario = $_POST["usuario"];  
$senha = $_POST["senha"];  
$nome = $_POST["nome"];  
$email = $_POST["email"];  
$cidade = $_POST["cidade"];  
$estado = $_POST["estado"];
```


13 – Referências

- Dall'Oglio, Pablo. Php 5 - PHP: Programando com Orientação a Objetos. 3ª Edição, São Paulo, Novatec, 2015.
- WALLACE, Soares. Php 5 - Conceitos, Programação e Integração com Banco de Dados. 7ª Edição, São Paulo, Erica, 2013.
- http://www.php.net/manual/pt_BR/index.php