

PHP

O básico

Alisson G. Chiquitto¹

¹Tecnologia em Análise e Desenvolvimento de Sistemas
Universidade Paranaense

Cianorte, 2016

Outline

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

A linguagem PHP

- **PHP**: Hypertext Preprocessor;
- Criado por **Rasmus Lerdorf** em 1995;
- 5º linguagem no **github.com** (Fevereiro/2016);
- 6º linguagem no **Tiobe Index** (Fevereiro/2016);
- 20 anos em 2015.

Fontes de pesquisa

- Documentação oficial (php.net) - Usar o manual em inglês; /pause
- W3Schools (www.w3schools.com/php/); /pause
- Stack Overflow (stackoverflow.com);

Fontes de pesquisa

- Documentação oficial (php.net) - Usar o manual em inglês; /pause
- W3Schools (www.w3schools.com/php/); /pause
- Stack Overflow (stackoverflow.com);

Conteúdo em português bom e atualizado? **Isto non-existe!**

Sites feitos em PHP

- Yahoo;
- Facebook;
- Wikipedia;
- Flickr;
- Wordpress.com (Plataforma para blogs);

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Tag PHP

- Todo o código PHP deve ser escrito entre as tags `<?php` e `?>`;
- Arquivos PHP tem a extensão `.php` - Ex.: `cliente.php`

Tag PHP

- Todo o código PHP deve ser escrito entre as tags `<?php` e `? >`;
- Arquivos PHP tem a extensão `.php` - Ex.: `cliente.php`

PHP e HTML:

As tags PHP podem se misturar com tags HTML;

Tag PHP

- Todo o código PHP deve ser escrito entre as tags `<?php` e `?>`;
- Arquivos PHP tem a extensão `.php` - Ex.: `cliente.php`

PHP e HTML:

As tags PHP podem se misturar com tags HTML;

```
<html>
<?php

// Todo o código PHP vai aqui

?>
</html>
```

PHP e HTML:

Um comentário é um trecho de texto no seu programa que serve apenas para explicar (documentar) o código, sem executar nenhum tipo de comando no programa. Muitas vezes, os comentários são usados também para desabilitar comandos no código. Nesse caso, dizemos que o código foi comentado.

Comentários

No PHP há três tipos de fazer comentários:

- Comentário de 1 linha com `//`;
- Comentário de 1 linha com `#`;
- Comentário de várias linhas com `/*` e `*/`

Comentários

No PHP há três tipos de fazer comentários:

- Comentário de 1 linha com `//`;
- Comentário de 1 linha com `#`;
- Comentário de várias linhas com `/*` e `*/`

```
<?php
```

```
// Comentario de 1 linha
```

```
# Comentario de 1 linha
```

```
/*
```

```
Comentario de varias linhas
```

```
Comentario de varias linhas
```

```
*/
```

```
?>
```

Gerar saída para o usuário

Geração de saídas:

Do que adiantaria processar algo se ninguém soubesse o resultado?

Gerar saída para o usuário

Geração de saídas:

Do que adiantaria processar algo se ninguém soubesse o resultado?

Na internet, o papel principal do PHP é a geração dinâmica de HTML. O PHP utiliza o construtor de linguagem **echo** para escrever Strings na saída do processo.

Gerar saída para o usuário

Geração de saídas:

Do que adiantaria processar algo se ninguém soubesse o resultado?

Na internet, o papel principal do PHP é a geração dinâmica de HTML. O PHP utiliza o construtor de linguagem **echo** para escrever Strings na saída do processo.

```
void echo ( string $arg1 [, string $... ] )
```

Envia para a saída uma ou várias *strings*.

Gerar saída para o usuário

```
void echo ( string $arg1 [, string $... ] )
```

Envia para a saída uma ou várias *strings*.

Gerar saída para o usuário

```
void echo ( string $arg1 [, string $... ] )
```

Envia para a saída uma ou várias *strings*.

```
<?php
```

```
echo "<h1>Data de hoje</h1>";
```

```
echo "<p>", "01/abr/2000", "</p>";
```

```
?>
```

Função print_r()

print_r() exibe informação sobre uma variável de forma legível para humanos.

mixed print_r(mixed \$expression)

Imprime informação sobre uma variável de forma legível

```
<?php
print_r("string"); // string
print_r(1); // 1
print_r(array('a','b')); // Array ( [0] => a [1] => b )
print_r(true); // 1
print_r(false); //
?>
```

Aviso

Não usar esta função para exibir dados para o usuário.

Função var_dump()

Esta função mostrará uma representação estruturada sobre uma ou mais expressões, incluindo o tipo e o valor.

```
mixed var_dump( mixed $expression )
```

Mostra informações sobre a variável.

```
<?php
var_dump("string"); // string(6) "string"
var_dump(1); // int(1)
var_dump(array('a','b')); // array(2) { [0]=> string(1) "a" [1]=> string(1) "b" }
var_dump(true); // bool(true)
var_dump(false); // bool(false)
?>
```

Aviso

Não usar esta função para exibir dados para o usuário.

print_r() x var_dump()

```
<?php
print_r("string"); // string
print_r(1); // 1
print_r(array('a','b')); // Array ( [0] => a [1] => b )
print_r(true); // 1
print_r(false); //
?>
```

```
<?php
var_dump("string"); // string(6) "string"
var_dump(1); // int(1)
var_dump(array('a','b')); // array(2) { [0]=> string(1) "a" [1]=> string(1) "b" }
var_dump(true); // bool(true)
var_dump(false); // bool(false)
?>
```

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
 - Variáveis
 - Constantes
 - Conversão de tipos
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
 - Variáveis
 - Constantes
 - Conversão de tipos
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Nomes de variáveis

Use as seguintes regras para criar variáveis:

\$

Variáveis tem que começar com o símbolo de dolar (\$);

Case-sensitive

Os nomes das variáveis são *case-sensitive*, assim *\$primeiroNome* é diferente de *\$PrimeiroNome*;

Letra ou *underscore*

Depois do **dolar** (\$), o próximo caractere deve ser uma letra ou o símbolo *underscore* (). Depois disto, o restante do nome da variável pode ter qualquer combinação de letras, números e *underscores*;

Variável *\$this*

A variável *\$this* é reservada para uso em Orientação a Objetos do PHP. Ela não pode ser utilizada em outros lugares;

Tipos de variáveis

escalar *boolean, integer, float e string;*

composto *array e objeto;*

especial *NULL e resource;*

Tipos de variáveis

escalar *boolean, integer, float e string;*

composto *array e objeto;*

especial *NULL e resource;*

Tipagem fraca

No PHP o tipo de uma variável é determinado pelo contexto em que a variável é utilizada. Isto significa que, se você atribuir uma *string* para a variável `$var`, `$var` se torna uma *string*. Se você atribuir um inteiro para `$var`, ela se torna um inteiro.

Tipos de variáveis

escalar *boolean, integer, float e string;*

composto *array e objeto;*

especial *NULL e resource;*

Tipagem fraca

No PHP o tipo de uma variável é determinado pelo contexto em que a variável é utilizada. Isto significa que, se você atribuir uma *string* para a variável `$var`, `$var` se torna uma *string*. Se você atribuir um inteiro para `$var`, ela se torna um inteiro.

Leia mais

Tipos de variáveis

Manipulação de tipos

Tipo Boolean

Um booleano expressa um valor verdade. Ele pode ser TRUE ou FALSE.

```
<?php
$verdadeiro = true;
$falso = false;

var_dump($verdadeiro); // bool(true)
var_dump($falso); // bool(false)
?>
```

Tipo Integer

Um inteiro é um número do conjunto $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$.

```
<?php
$a = 1234;
$b = -123; // um número negativo
$c = 0123; // número octal
$d = 0x1A; // número hexadecimal

echo $a; // 1234
var_dump($a); // int (1234)
?>
```

Tipo Float

Números de ponto flutuante (também conhecidos como *floats*, *doubles* ou números reais) podem ser especificados utilizando qualquer uma das seguintes sintaxes:

```
<?php  
$a = 1.234;  
$b = 1.2e3;  
$c = 7E-10;  
?>
```

Tipo String

Uma *string* é uma cadeia de caracteres. Utiliza-se apóstrofos ou aspas para delimitar uma *string*.

```
<?php  
$a = 'String';  
$b = "String";  
?>
```

Leia mais

<http://php.net/manual/en/language.types.string.php>

http://www.w3schools.com/php/php_string.asp

Tipo String

Apóstrofos x Aspas

apóstrofo A *string* é utilizada como "ela é";

aspas O conteúdo da *string* é interpretada, buscando por sequências de escape (ex.: \n) ou outras variáveis.

```
<?php  
  
$pais = 'Brasil';  
  
echo 'Estamos no $pais'; // Estamos no $pais  
echo "Estamos no $pais"; // Estamos no Brasil
```

Leia mais

Interpretação de strings

Tipo Array

Tipo que relaciona valores a chaves. Um array pode ser criado com o construtor de linguagem `array()` com um certo número de pares (chave => valor) separados por vírgula.

```
<?php
// Esquema para criacao de um array
// $var = array( chave => valor, ...);
// chave pode ser tanto string ou um integer
// valor pode ser qualquer tipo de valor

// Array ( [0] => 1 [1] => 2 )
$ex1 = array(1,2);
$ex2 = array(0=>1, 1=>2);
```

Tipo Array

Exemplo 1

Criação de um array com valores, e uso destes valores.

```
<?php
$a = array(1,2,'a','b');

echo $a[0]; // 1
echo $a[2]; // a

// Array ( [0] => 1 [1] => 2
//      [2] => a [3] => b )
print_r($a);
```

Tipo Array

Exemplo 2

Criação de um array com valores, alteração e uso dos valores.

```
<?php
$a = array(1,2,'a','b');

echo $a[0]; // 1
echo $a[2]; // a

$a[2] = 'c';
echo $a[2]; // c

$a[4] = 'd';

// Array ( [0] => 1 [1] => 2 [2] => c
//      [3] => b [4] => d )
print_r($a);
```

Tipo Array

Exemplo 3

Criação de um array vazio, inclusão e uso dos valores.

```
<?php
$a = array();

$a[0] = 'a';
$a[1] = 'c';
$a[2] = 'e';

echo $a[0]; // a
echo $a[2]; // e

// Array ( [0] => a [1] => c [2] => e )
print_r($a);
```

Tipo Array

Exemplo 4

Criação de um array vazio, inclusão em chaves fora da ordem natural.

```
<?php
$a = array();

$a[0] = 'a';
$a[5] = 'c';
$a[2] = 'e';

echo $a[5]; // c

// Array ( [0] => a [5] => c [2] => e )
print_r($a);
```

Tipo Array

Exemplo 5

Criação de um array com valores em chaves personalizadas.

```
<?php
$a = array(10=>100,11=>200);

echo $a[10]; // 100

// Array ( [10] => 100 [11] => 200 )
print_r($a);
```

Tipo Array

Exemplo 6

Criação de um array com valores em chaves personalizadas.

```
<?php
$a = array(
    10 => 100,
    11 => 200
);

// Array ( [10] => 100 [11] => 200 )
print_r($a);
```

Tipo Array

Exemplo 7

Array com chaves do tipo *string*.

```
<?php
$cliente = array(
    'nome' => 'Zezinho',
    'idade' => 18,
    'altura' => 170
);

// Array ( [nome] => Zezinho
//      [idade] => 18
//      [altura] => 170 )
print_r($cliente);
```


Tipo Array

Exemplo 8

Array em array (matriz).

```
<?php
$pos = array(
    array('x'=>10,'y'=>15),
    array('x'=>20,'y'=>30),
);

echo $pos[0]['x']; // 10
echo $pos[1]['x']; // 20

/* Array (
    [0] => Array ( [x] => 10 [y] => 15 )
    [1] => Array ( [x] => 20 [y] => 30 )
) */
print_r($pos);
```

Tipo Array

Exemplo 9

Matriz (Exemplo 2)

```
<?php
$frutas = array('banana', 'laranja');
$folhas = array('alface', 'couve');
$alimento = array($frutas, $folhas);

echo $frutas[0]; // banana
echo $alimento[0][0]; // banana

echo $folhas[1]; // couve
echo $alimento[1][1]; // couve

/* Array (
  [0] => Array ( [0] => banana [1] => laranja )
  [1] => Array ( [0] => alface [1] => couve )
) */
print_r($alimento);
```

Tipo Objeto

Em orientação a objetos, representa a instância de uma classe.

Tipo Null

O valor especial NULL representa que a variável não tem valor. NULL é o único valor possível do tipo NULL.

```
<?php
$var = NULL;

echo $var; // vazio
print_r($var); // vazio
var_dump($var); // NULL
?>
```

Tipo Resource

Um recurso é uma variável especial, que mantém uma referência a um recurso externo. Recursos são criados e usados por funções especiais.

```
<?php
// conexao com BD
$con = mysqli_connect();

// link com arquivo aberto
$file = fopen();
?>
```

Leia mais

<http://php.net/manual/en/resource.php>

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
 - Variáveis
 - Constantes
 - Conversão de tipos
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Constantes

- Uma constante após definida, armazena apenas um valor até o término da execução do script;
- O escopo da **constante** é **global**, isto significa que você pode acessar a constante de qualquer lugar do script (dentro e fora de funções, dentro de classes e métodos, dentro de blocos de laços, etc);
- Uma constante é definida pelo uso da função `define()`;

Constantes

Regras para definição de constantes

`define()`

Use esta função para definir constantes;

Case-sensitive

Por convenção, uma constante é sempre criada em caixa alta.

Letra ou *underscore*

Uma constante deve iniciar com um caractere de letra ou um *underscore*, seguido por letras, números e *underscores*;

Restrições de tipo

Somente dados escalares (*boolean*, inteiros, *floats* e *strings*) podem ser armazenados em constantes;

Constantes

Exemplo de uso

```
<?php

define('FIM_ANO', false);
define('ANO', 2014);
define('MES', 'fevereiro');

$fimAno = FIM_ANO;
echo MES;
echo ANO;
```

Constantes

Constantes pré-definidas

"O PHP fornece um grande número de constantes pré-definidas. A maioria dessas constantes, entretanto, são criadas por várias extensões, e somente estarão presentes quando essas extensões estiverem disponíveis."

Leia mais

[PHP Manual - Predefined Constants](#)

Constantes mágicas

Constantes "mágicas" **mudam** de valor dependendo de onde elas são utilizadas.

Exemplos:

`__LINE__`

A linha atual do script;

`__FILE__`

O caminho completo e nome do arquivo. Se utilizado dentro de um include, o nome do arquivo incluído será retornado;

Constantes mágicas

```
<?php

// escreve o nome do arquivo
echo __FILE__;

// escreve a linha da instrucao
echo __LINE__;
```

Leia mais

PHP Manual - Magic Constants

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
 - Variáveis
 - Constantes
 - Conversão de tipos
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Conversão de tipos

A conversão de tipos no PHP funciona como no C: o nome de um tipo desejado é escrito entre parênteses antes da variável em que se deseja a moldagem.

`int/integer` converte para um número inteiro;

`bool/boolean` converte para booleano;

`float/double/real` converte para um número de ponto flutuante;

`string` converte para string;

`array` gera um array com o valor informado;

`object` converte para objeto;

Conversão para Boolean

Ao converter para booleano, os valores a seguir são considerados FALSE:

- o próprio booleano FALSE;
- o inteiro 0 (zero);
- o ponto flutuante 0.0 (zero);
- uma string vazia ("") e a string "0";
- um array sem elementos;
- um objeto sem elementos;
- o tipo especial NULL;
- o objeto SimpleXML vazio;

Qualquer outro valor é considerado TRUE (incluindo qualquer *resource*).

Conversão para Boolean

```
<?php
var_dump((bool) "");           // bool(false)
var_dump((bool) 1);            // bool(true)
var_dump((bool) -2);           // bool(true)
var_dump((bool) "foo");        // bool(true)
var_dump((bool) 2.3e5);         // bool(true)
var_dump((bool) array(12));     // bool(true)
var_dump((bool) array());       // bool(false)
var_dump((bool) "false");       // bool(true)
```


Conversão para Inteiro

boolean FALSE será retornado como 0 (zero), e TRUE como 1 (um);

float Quando convertendo de números de ponto flutuante para inteiros, o número será truncado (ignorado as casas decimais);

```
<?php
var_dump((int) false);    // int(0)
var_dump((int) true);     // int(1)
var_dump((int) 3.1415);   // int(3)
var_dump((int) -10.5);    // int(-10)
```

Conversão para Float

Basicamente uma conversão para inteiro, mas o retorno do tipo *float*.

```
<?php
var_dump((int) false);    // int(0)
var_dump((float) false); // float(0)
var_dump((int) true);     // int(1)
var_dump((float) true);   // float(1)
```

Conversão de String para Int/Float

Quando uma string é avaliada como um valor numérico, o valor resultante e o tipo é determinado como segue.

A string será avaliada como um ponto flutuante se conter qualquer um dos caracteres ., e, ou E. Em outros casos, ela será avaliada como um inteiro.

O valor é obtido da porção inicial da *string*. Se a *string* começa com dados numéricos válidos, esse será o valor utilizado. Em outro caso, o valor será 0 (zero). Dados numéricos válidos são: um sinal opcional, seguido por um ou mais dígitos (opcionalmente contendo um ponto decimal), seguido de um expoente, também opcional. O expoente é um e ou E seguido de um ou mais dígitos.

Conversão de String para Int/Float

Se: *string* começa com dados numéricos, esse será o valor utilizado
Senão: 0.

```
<?php
var_dump((int) "10.5");           // int(10)
var_dump((float) "10.5");         // float(10.5)
var_dump((int) "-1.3e3");          // int(-1)
var_dump((float) "-1.3e3");        // float(-1300)
var_dump((int) "bob3");           // int(0)
var_dump((float) "bob3");          // float(0)
var_dump((int) "10.2 Little Piggies"); // int(10)
var_dump((float) "10.2 Little Piggies"); // float(10.2)
```

Leia mais

Manual PHP Conversão de strings para números

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores**
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores**
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operador de concatenação

A concatenação é feita pelo operador ponto (.).

```
<?php  
  
$a = 'ABC';  
$b = 'DEF';  
  
// ABCDEF  
echo $a . $b;
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operadores aritméticos

Realizam calculos aritméticos.

- + Soma;
- Subtração/Inversão de sinal;
- * Multiplicação;
- / Divisão;
- % Módulo (Resto);
- ** Exponencial;

Operadores aritméticos

Exemplos de uso

```
<?php
```

```
$a = 6;
```

```
$b = 2;
```

```
$c = $a + $b; // 8
```

```
$c = $a - $b; // 4
```

```
$c = -$a; // -6
```

```
$c = $a * $b; // 12
```

```
$c = $a / $b; // 3
```

```
$c = $a % $b; // 0
```

```
$c = $a % 4; // 2
```

```
$c = $a ** $b; // 36
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operadores de atribuição

Operadores responsáveis em atribuir valores para as variáveis. Alguns fazem calculos aritméticos.

- = Atribuição simples;
- += Soma e Atribuição;
- = Subtração e Atribuição;
- *= Multiplicação e Atribuição;
- /= Divisão e Atribuição;
- %= Módulo e Atribuição;

Operadores de atribuição

Exemplos de uso

```
<?php
```

```
$a = 6;
```

```
$b = 2;
```

```
$a += $b; // $a = $a + $b
```

```
$a -= $b; // $a = $a - $b;
```

```
$a *= $b; // $a = $a * $b;
```

```
$a /= $b; // $a = $a / $b;
```

```
$a %= $b; // $a = $a % $b;
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operadores de comparação

Operadores de comparação comparam dois valores e sempre retornam um valor booleano.

<code>==</code> Igual;	<code>></code> Maior;
<code>===</code> Idêntico;	<code><</code> Menor;
<code>!=</code> Diferente;	<code>>=</code> Maior ou Igual;
<code><></code> Diferente;	<code><=</code> Menor ou Igual;
<code>!==</code> Não idêntico;	

Operadores de comparação

Descrição

Os operadores de comparação retornam true se:

<code>==</code>	<code>\$a</code>	<code>==</code>	<code>\$b</code>	<code>\$a</code> é igual a <code>\$b</code>
<code>!=</code>	<code>\$a</code>	<code>!=</code>	<code>\$b</code>	<code>\$a</code> é igual a <code>\$b</code>
<code>></code>	<code>\$a</code>	<code>></code>	<code>\$b</code>	<code>\$a</code> é maior que <code>\$b</code>
<code><</code>	<code>\$a</code>	<code><</code>	<code>\$b</code>	<code>\$a</code> é menor que <code>\$b</code>
<code>>=</code>	<code>\$a</code>	<code>>=</code>	<code>\$b</code>	<code>\$a</code> é maior ou igual que <code>\$b</code>
<code><=</code>	<code>\$a</code>	<code><=</code>	<code>\$b</code>	<code>\$a</code> é menor ou igual que <code>\$b</code>
<code>===</code>	<code>\$a</code>	<code>===</code>	<code>\$b</code>	<code>\$a</code> é igual a <code>\$b</code> e eles são do mesmo tipo
<code>!==</code>	<code>\$a</code>	<code>!==</code>	<code>\$b</code>	<code>\$a</code> não é igual a <code>\$b</code> ou eles não são do mesmo tipo

Operadores de comparação

Exemplos de uso

```
<?php
$a = 6;
$b = 2;

$c = ($a == $b); // false
$c = ($a != $b); // true
$c = ($a > $b); // true
$c = ($a < $b); // false
$c = ($a >= $b); // true
$c = ($a <= $b); // false
```

Operadores de comparação

Exemplos de uso

```
<?php
$c = (10 == '10'); // true
$c = (10 != '10'); // false

$c = (10 === '10'); // false

$c = (10 !== 1); // true
$c = (10 !== '10'); // true
$c = (10 !== 10); // false
$c = ('10' !== '10'); // false
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operadores de Incremento/Decremento

Incrementam (+1) e decrementam (-1) o valor de números.

`$a++` Retorna `$a` e então faz o incremento;

`$a--` Retorna `$a` e então faz o decremento;

`++$a` Incrementa `$a` e então retorna o novo valor;

`--$a` Decrementa `$a` e então retorna o novo valor.

Operadores de incremento/decremento

Exemplos de uso

```
<?php
```

```
// $a = $a + 1
```

```
$a++;
```

```
++$a;
```

```
// $a = $a - 1
```

```
$a--;
```

```
--$a;
```

```
// echo $a; $a = $a + 1;
```

```
echo $a++;
```

```
// $a = $a + 1; echo $a;
```

```
echo ++$a;
```

```
// echo $a; $a = $a - 1;
```

```
echo $a--;
```

```
// $a = $a - 1; echo $a;
```

```
echo --$a;
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
 - Concatenação
 - Aritméticos
 - Atribuição
 - Comparação
 - Incremento/Decremento
 - Lógicos
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Operadores Lógicos

Realizam operações lógicas e retornam um valor booleano.

`and` E;

`&&` E;

`or` OU;

`||` OU;

`!` Negação;

`xor` OU Exclusivo (xor);

Operadores Lógicos

Descrição

Os operadores lógicos retornam:

<code>and</code>	<code>\$a</code>	<code>and</code>	<code>\$b</code>	true se <code>\$a</code> E <code>\$b</code> forem true;
<code>or</code>	<code>\$a</code>	<code>or</code>	<code>\$b</code>	true se <code>\$a</code> OU <code>\$b</code> forem true;
<code>xor</code>	<code>\$a</code>	<code>xor</code>	<code>\$b</code>	true se apenas <code>\$a</code> OU <code>\$b</code> for true;
<code>!</code>	<code>!\$a</code>			Inverte o valor de <code>\$a</code>

Operadores Lógicos

Exemplos de uso

```
<?php
```

```
$t = true;
```

```
$f = false;
```

```
$c = ($t and $f); // false
```

```
$c = ($t or $f); // true
```

```
$c = ($t xor $f); // true
```

```
$c = !$t; // false
```

```
$c = !$f; // true
```

Leia mais

- PHP: Arithmetic Operators
- PHP: Assignment Operators
- PHP: Comparison Operators
- PHP: Incrementing/Decrementing Operators
- PHP: Logical Operators
- PHP: Operator Precedence
- PHP 5 Operators

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Conditionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Estruturas de controle

Estruturas de controle

"(...) estruturas de controle refere-se à ordem em que instruções, expressões e chamadas de função são executadas ou avaliadas em programas de computador (...)" (Wikipédia)

Estruturas de controle

Suportadas pelo PHP

- if/elseif/else
- while
- do-while
- for
- foreach
- break
- continue
- switch
- include/require

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Condicionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Estrutura if

- O if permite a execução condicional de um bloco de instruções;
- A expressão será sempre avaliada como um *boolean*;
- Se a expressão for avaliada como true, o bloco de instruções será executado.

```
<?php  
  
$a = 1;  
$b = 10;  
  
// Se A > B  
if ($a > $b) {  
    // entao escreve uma mensagem  
    echo "A > B";  
}
```

Estrutura if

```
<?php

$c = (5 > 6);
if ($c) {
    echo "C = true";
}

// avaliacao do resultado
// de uma atribuicao
if ($d = (15 > 10)) {
    echo "D = true";
}
```


Estrutura if/else

else

O else estende a instrução if para executar outras instruções caso a expressão do if retornar false.

```
<?php  
  
$numero = 17;  
  
if ($numero >= 18) {  
    echo "Numero >= 18";  
} else {  
    echo "Numero < 18";  
}
```

Estrutura if/elseif/else

elseif

Extende um if para executar instruções diferentes no caso do if retornar false. Entretanto, diferente do else, ele executará uma expressão alternativa somente se a condição do elseif retornar true.

```
<?php

$numero = 17;

if ($numero >= 18) {
    echo "Numero >= 18";
} elseif ($numero >= 16) {
    echo "Numero >= 16";
} else {
    echo "Numero < 16";
}
```

Estrutura if/elseif x if

Comparação de resultados

```
<?php
```

```
$numero = 19;
```

```
if ($numero >= 18) {  
    echo "Numero >= 18";  
}  
elseif ($numero >= 17) {  
    echo "Numero >= 16";  
}  
elseif ($numero >= 16) {  
    echo "Numero >= 16";  
}  
else {  
    echo "Numero < 16";  
}
```

```
<?php
```

```
$numero = 19;
```

```
if ($numero >= 18) {  
    echo "Numero >= 18";  
}  
if ($numero >= 17) {  
    echo "Numero >= 16";  
}  
if ($numero >= 16) {  
    echo "Numero >= 16";  
}  
else {  
    echo "Numero < 16";  
}
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Condicionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Iteração

Iteração

1. Ato de iterar; repetição;
2. Iteração é o processo chamado na programação de repetição de uma ou mais ações.

Estrutura While

“Repetição de um trecho de código. Geralmente utilizado quando **não** se sabe de antemão a quantidade de iterações.”

- Loop mais simples do PHP;
- Executa o bloco de instruções enquanto `$expr` retornar `true`;

```
while ($expr) {  
    // instrucoes  
}
```

Estrutura While

Exemplo de uso

```
<?php

// 2468
$i = 2;
while ($i < 10) {
    echo $i;
    $i+=2;
}
```

Estrutura Do-While

“Repetição de um trecho de código. Geralmente utilizado quando sabe-se de antemão que o mínimo de iterações será 1.”

- Semelhante ao while, exceto que `$expr` será avaliado no final de cada iteração;
- Garante no mínimo 1 iteração;

```
do {  
    // instrucoes  
} while ($expr);
```


Estrutura Do-While

Exemplo de uso

```
<?php  
  
// 2468  
$i = 2;  
do {  
    echo $i;  
    $i+=2;  
} while ($i < 10);
```

Estrutura For

Executam um bloco de código por um determinado número de vezes. Geralmente utilizado quando sabe-se de antemão a quantidade de iterações.

- `$expr1` é sempre executado no início do laço;
- antes de cada iteração, `$expr2` é avaliado: se true, executa o bloco de instruções, senão a execução do laço termina;
- após cada iteração, `$expr3` é executada;

```
for ($expr1; $expr2; $expr3) {  
    // instrucoes  
}
```

Estrutura For

Exemplo de uso

```
<?php  
  
// 2468  
for ($i = 2; $i < 10; $i+=2) {  
    echo $i;  
}
```

Estrutura while vs for

Comparação de estrutura

```
while ($expr) {  
    // instrucoes  
}
```

```
for ($expr1; $expr2; $expr3) {  
    // instrucoes  
}
```

Estrutura while x for

Comparação em implementação

```
<?php

// 2468
$i = 2;
while ($i < 10) {
    echo $i;
    $i+=2;
}
```

```
<?php

// 2468
for ($i = 2; $i < 10; $i+=2) {
    echo $i;
}
```

Estrutura Foreach

Interface para iterações sobre *arrays* e objetos.

- O laço itera sobre `$valores`.
Em cada iteração, o valor do elemento corrente é atribuído à variável `$v`, e o ponteiro interno avança para o próximo elemento.
- Na segunda forma, também há a atribuição da chave do elemento corrente para a variável `$k`.

```
<?php

// Forma 1
foreach ($valores as $v) {

}

// Forma 2
foreach ($valores as $k => $v) {

}
```

Estrutura Foreach

Exemplo de uso da forma 1

```
<?php

$a = array(0,2,4);

/*
Valor = 0
Valor = 2
Valor = 4
*/
foreach ($a as $v) {
    echo "Valor = $v";
}
```

Estrutura Foreach

Exemplo de uso da forma 2

```
<?php

$a = array(0,2,4);

/*
Chave = 0; Valor = 0
Chave = 1; Valor = 2
Chave = 2; Valor = 4
*/
foreach ($a as $k => $v) {
    echo "Chave = $k; Valor = $v";
}
```


Estrutura Foreach

Exemplo de uso

```
<?php

$cliente = array(
    'nome' => 'Curry India',
    'idade' => 67
);

/*
Valor = Curry India
Valor = 67
*/
foreach ($cliente as $v) {
    echo "Valor = $v";
}
```

Estrutura Foreach

Exemplo de uso

```
<?php
```

```
$uf = array(
    'PR' => array('nome' => 'Parana', 'capital' => 'Curitiba'),
    'AM' => array('nome' => 'Amazonas', 'capital' => 'Manaus'),
);

/*
Capital PR = Curitiba
Capital AM = Manaus
*/
foreach ($uf as $k => $v) {
    echo "Capital $k = " . $v['capital'];
}
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Condicionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Estrutura Break

A instrução `break` para a execução das seguintes estruturas de controle:

- `while`
- `do while`
- `for`
- `foreach`
- `switch`

```
<?php

// 01234
for ($i = $i; $i < 10; $i++) {
    if ($i == 5) {
        break;
    }

    echo $i;
}
```

Estrutura Continue

Continue

Utilizado em estruturas de *loop* para pular o resto da iteração atual do *loop* e iniciar a próxima iteração.

```
<?php

// 13579
for ($i = 0; $i < 10; $i++) {
    if (($i % 2) == 0) {
        continue;
    }

    echo $i;
}
```

Estrutura Return

Continue

Faz com que a função termine sua execução imediatamente passando o controle de volta para o ponto de onde ela foi chamada. Além disso retorna seu argumento como valor da chamada dessa função.

```
<?php  
  
function somar($a, $b) {  
    return $a + $b;  
}
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Condicionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Estrutura Switch

Útil quando se deseja fazer comparações de igualdade.

```
<?php  
  
switch($expr) {  
    case coincidencia1:  
        // instrucoes 1  
    case coincidencia2:  
        // instrucoes 2  
        break;  
    default:  
        // instrucoes 3  
}
```

- O valor de **\$expr** é testado, na ordem, contra os valores das constantes especificadas nos comandos case;
- Quando uma **coincidência** for encontrada, a seqüência de comando associada àquele case será executada até que o comando **break** ou o **fim do comando switch** seja alcançado;
- O comando default é executado se nenhuma *coincidência* for detectada;
- O default é opcional.

Estrutura Switch

Exemplo de uso

```
<?php

$n = 2;

switch($n) {
    case 2:
    case 4:
        echo 'par';
        break;
    case 1:
    case 3:
        echo 'impar';
        break;
    default:
        echo 'desconhecido';
}
```

```
<?php

$n = 10;

switch($n) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        echo '<=5';
        break;
    default:
        echo '>5';
}
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
 - Condicionais
 - Laços
 - Break/Continue/Return
 - Switch
 - Inclusão de arquivos
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)

Estrutura Include

A instrução `include` inclui e avalia um arquivo específico.

```
<?php  
  
include 'arquivo.php';
```

Estrutura Include

A instrução `require` inclui e avalia um arquivo específico.

```
<?php  
  
require 'arquivo.php';
```

Aviso

Diferentemente do `include`, o `require` lança um **erro fatal** caso o arquivo a incluir não for encontrado.

Leia mais

- PHP: Control Structures
- Estruturas de controle no PHP
- A instrução return do PHP e o HTML
- If, else e elseif as estruturas de controle no PHP
- PHP Control Structures

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySQL (mysqli)

Definição

"Funções são formas de modularizar uma ou mais linhas de código de maneira que possam ser executadas em diferentes momentos do script/aplicação quando necessário."

"A função poderia ser definida como um conjunto de instruções que permitem processar as variáveis para obter um resultado."

"As funções são métodos de economizar tempo e trabalho para ações que irão se repetir."

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Assinatura de funções

Assinatura de função

Cada função pode ser representada por uma assinatura (protótipo de função) para referência rápida.

```
int soma (int $a, int $b)
```

Retorna a soma entre \$a e \$b.

Assinatura de funções

retorno nome (parametros)

Descrição.

retorno Tipo de retorno da função. Pode ser um tipo do PHP ou `mixed`, indicando que a função pode retornar mais de um tipo;

nome Nome da função;

parametros Lista de parâmetros divididos por vírgula. Necessário informar o tipo de cada parâmetro.

descrição Descrição/Detalhes da função.

Assinatura de funções

int strlen (string \$string)

Retorna o tamanho de uma *string*.

string implode (string \$glue , array \$pieces)

Junta elementos de uma matriz em uma *string*.

string trim (string \$str)

Retira espaço no início e final de uma *string*.

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Chamada de funções

Toda e qualquer função deve ser chamada pelo nome identificador seguido de parenteses.

- Os parâmetros devem ser informados dentro dos parenteses;
- A quantidade de parâmetros deve ser igual ao definido na função;
- Se não houver parâmetros, deixe os parentes vazios.

Chamada de funções

Exemplo

string strtoupper (\$string)

Converte uma *string* para maiúscula.

```
<?php  
  
// UNIPAR  
$a = strtoupper('unipar');  
  
$b = strtoupper('php');  
echo $b; // PHP
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Criação de funções

"Toda função deve iniciar com `function` e possuir um nome identificador, podendo ter ou não parâmetros que devem ser definidos dentro de parênteses, já seu bloco de código é delimitado por chaves."

```
<?php  
  
function nome( parametros ) {  
    // instrucoes  
}
```

Aviso

A nomenclatura de uma função deve seguir as mesmas regras de nomenclatura de uma variável.

Criação de funções

Função sem parâmetros

Função que escreve números entre 1 e 10.

```
<?php

function numero1e10( ) {
    for ($i = 1; $i <= 10; $i++) {
        echo $i;
    }
}

// 12345678910
numero1e10();
```

Criação de funções

Função com 1 parâmetro

Função que escreve números entre 1 e \$max.

```
<?php

function numero1eMax( $max ) {
    for ($i = 1; $i <= $max; $i++) {
        echo $i;
    }
}

// 12345
numero1eMax(5);
```

Criação de funções

Função com mais de 1 parâmetro

Função que escreve números entre \$min e \$max.

```
<?php

function numeroMinMax( $min, $max ) {
    for ($i = $min; $i <= $max; $i++) {
        echo $i;
    }
}

// 56789
numeroMinMax(5, 9);
```

Criação de funções

Instrução return

"Valores podem ser retornados utilizando a instrução opcional `return`. Qualquer tipo pode ser retornado, incluindo *arrays* e objetos."

```
<?php

function media($a, $b) {
    return ($a + $b) / 2;
}

echo media (1, 3); // 2
```

Aviso

Quando uma instrução `return` é executado, a execução da função é imediatamente interrompida e o valor retornado.

Criação de funções

Parâmetros opcionais

"Uma função pode definir valores padrão para argumentos, como a seguir:"

```
<?php
function cafeteira ($tipo = "cappuccino") {
    return "Fazendo uma xícara de café $tipo";
}

// Fazendo uma xicara de café cappuccino
echo cafeteira();

// Fazendo uma xicara de café
echo cafeteira(null);

// Fazendo uma xicara de café expresso
echo cafeteira("expresso");
```

Criação de funções

Parâmetros opcionais: forma errada

```
<?php

function iogurtera ($tipo = 'azedada', $sabor) {
    return 'Fazendo uma taça de $sabor $tipo';
}

echo iogurtera ('framboesa');
```

Criação de funções

Parâmetros opcionais: forma errada

Aviso

Um parâmetro opcional nunca deve ser declarado antes de qualquer parâmetro obrigatório.

```
<?php
// errado (nem executa)
function iogurtera ($tipo = 'azedo', $sabor) {
    return 'Fazendo uma taça de $sabor $tipo';
}
echo iogurtera ('morango');

// certo
function iogurtera ($sabor, $tipo = 'azedo') {
    return 'Fazendo uma taça de $sabor $tipo';
}
echo iogurtera ('morango'); // Fazendo uma taça de morango azedo
echo iogurtera ('morango', 'doce'); // Fazendo uma taça de morango doce
```

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Manipulação de strings I

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

Divide uma *string* em *strings*. Retorna uma matriz de *strings*.

```
string implode ( string $glue , array $pieces )
```

Junta elementos de uma matriz em uma *string*. Retorna uma *string* contendo os elementos da matriz.

```
string nl2br ( string $string )
```

Insere quebras de linha HTML antes de todas *newlines* em uma string

Manipulação de strings II

```
string number_format ( float $number [, int $decimals ] )
```

Formata um número com os milhares agrupados.

```
string number_format ( float $number , int $decimals , string  
$dec_point , string $thousands_sep )
```

Formata um número com os milhares agrupados.

```
int strlen ( string $string )
```

Retorna o tamanho da *string*.

Manipulação de strings III

```
int strpos ( string $haystack , string $needle [, int $offset ] )
```

Encontra a posição da primeira ocorrência de uma *string*.

```
string strtolower ( string $str )
```

Converte uma *string* para caixa baixa.

```
string strtoupper ( string $str )
```

Converte uma *string* para caixa alta.

```
string strtr ( string $str , array $replace_pairs )
```

Traduz certos caracteres.

Manipulação de strings IV

```
string substr ( string $string , int $start [, int $length ] )
```

Retorna uma parte de uma *string*.

```
string ucfirst ( string $str )
```

Converte para maiúscula o primeiro caractere de uma *string*.

```
string ucwords ( string $str )
```

Converte para maiúscula o primeiro caractere de cada palavra.

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Manipulação de arrays I

```
array array_diff ( array $array1 , array $array2 [, array $ ... ] )
```

Retorna as diferenças entre *arrays*.

```
array array_intersect ( array $array1 , array $array2 [, array $ ... ] )
```

Calcula a interseção entre *arrays*.

```
bool array_key_exists ( mixed $key , array $search )
```

Checa se uma chave ou índice existe em um *array*.

Manipulação de arrays II

```
array array_keys ( array $input [, mixed $search_value [,  
bool $strict ]] )
```

Retorna todas as chaves de um *array*.

```
array array_merge ( array $array1 [, array $array2 [, array $  
... ]] )
```

Funde um ou mais *arrays*.

```
mixed array_pop ( array &$amp;array )
```

Remove um elemento do final do *array* e o retorna.

Manipulação de arrays III

```
number array_product ( array $array )
```

Calcula o produto dos valores de um *array*.

```
int array_push ( array &$amp;array , mixed $var [, mixed $ ...  
] )
```

Adiciona um ou mais elementos no final de um *array*.

```
mixed array_rand ( array $input [, int $num_req ] )
```

Retorna um ou mais elementos aleatórios de um *array*.

```
mixed array_shift ( array &$amp;array )
```

Retira o primeiro elemento do *array* e o retorna.

Manipulação de arrays IV

```
number array_sum ( array $array )
```

Calcula a soma dos elementos.

```
int array_unshift ( array &$amp;array , mixed $var [, mixed $ ...  
] )
```

Adiciona um ou mais elementos no início de um *array*.

```
int array_unshift ( array &$amp;array , mixed $var [, mixed $ ...  
] )
```

Adiciona um ou mais elementos no início de um *array*.

Manipulação de arrays V

```
bool arsort ( array &$array [, int $sort_flags ] )
```

Ordena um *array* em ordem decrescente mantendo a associação entre índices e valores.

```
bool asort ( array &$array [, int $sort_flags ] )
```

Ordena um *array* mantendo a associação entre índices e valores.

```
int count ( mixed $var [, int $mode ] )
```

Conta o número de elementos de uma variável, ou propriedades de um objeto.

Manipulação de arrays VI

```
bool in_array ( mixed $needle , array $haystack [, bool $strict ] )
```

Checa se um valor existe em um array

Outline - Subseção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle

5 Funções

- Assinatura de funções

- Chamando funções

- Criação de funções

- Manipulação de Strings

- Manipulação de Arrays

- Manipulação de Data

6 Envio de dados via GET e POST

7 Conexão com Banco de Dados MySql (mysqli)

Timestamp

"*Timestamp* é uma representação de tempo e é formado por um **número inteiro**. O *timestamp* de uma data é o número inteiro que representa a **quantidade de segundos que se passaram desde o dia 1 de janeiro de 1970.**"

0 01/01/1970 00:00:00 GMT

1 01/01/1970 00:00:01 GMT

60 01/01/1970 00:01:00 GMT

3600 01/01/1970 01:00:00 GMT

946684800 01/01/2000 00:00:00hrs

Leia mais

Epoch Unix Time Stamp Converter

Manipulação de data/hora I

```
string date ( string $format [, int $timestamp ] )
```

Retorna a data/hora atual de acordo com a *string* \$format. Se \$timestamp foi fornecido, então utiliza este valor em vez de data/hora atual.

```
int mktime ([ int $hora [, int $minuto [, int $second [, int  
$mes [, int $dia [, int $ano [, int $is_dst ]]]]] ] )
```

Obtém um timestamp Unix para uma data.

```
int strtotime ( string $time [, int $now ] )
```

Analisa uma *string* (em inglês) de data/hora e retorna o *timestamp* correspondente.

Leia mais

- PHP: String Functions
- PHP: Array Functions
- PHP: Date/Time Functions
- PHP: date function
- Aritmética de Datas com PHP
- Timestamp em PHP

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST**
- 7 Conexão com Banco de Dados MySql (mysqli)

Outline - Seção

- 1 Conceitos básicos
- 2 Variáveis e constantes
- 3 Operadores
- 4 Estruturas de controle
- 5 Funções
- 6 Envio de dados via GET e POST
- 7 Conexão com Banco de Dados MySql (mysqli)**

For Further Reading I



A. Author.

Handbook of Everything.

Some Press, 1990.



S. Someone.

On this and that.

Journal of This and That, 2(1):50–100, 2000.