

Agrupamentos de Escola de Mondim de Basto

PHP



Trabalho elaborado por: Rute Torres

Número:14

Ano/Turma: 11ºD

Professora: Anabela Leite

Disciplina: Redes de comunicação

Módulo: 05-Desenvolvimento de páginas Web dinâmicas

Índice

INTRODUÇÃO.....	4
CARACTERÍSTICAS DO PHP.....	5
O QUE É PRECISO PARA INSTALAR O PHP?.....	6
Instalação em Windows.....	6
SÍNTEXE BÁSICA EM PHP.....	7
Comentário em PHP.....	8
DECLARAÇÃO DE VARIÁVEIS.....	9
Tipos de dados.....	10
a) String:.....	10
b) Integer (Inteiros).....	11
c) Booleanos.....	12
d) Array.....	13
e) Objetos.....	14
f) NULL.....	14
g) Resources.....	14
INTERPRETAÇÃO DE STRINGS.....	15
Funções das Strings.....	16
CONSTANTES.....	20
OPERADORES.....	21
I. Operadores aritméticos.....	21
II. Operadores Bit-a-bit.....	21
III. Operadores de comparação.....	21
IV. Operadores de Incremento/Decremento.....	22
V. Operadores Lógicos.....	22
VI. Operadores de Array.....	22
ESTRUTURAS DE CONTROLO.....	23
A. IF.....	23
B. ELSE.....	23
C. ELSEIF.....	24
CONCLUSÃO.....	25
WEBGRAFIA.....	26

Índice de Ilustrações

Ilustração 1- Síntaxe básica em PHP	7
Ilustração 2- Exemplo comentário em PHP	8
Ilustração 3-Comentário em linha.....	8
Ilustração 4-Comentário na linha.....	8
Ilustração 5- Exemplo de string com aspas simples.....	10
Ilustração 6-Literais inteiras	11
Ilustração 7- Um array simples.....	13

INTRODUÇÃO

No âmbito da disciplina de Redes de Comunicação vou falar sobre a criação de páginas Web dinâmicas, mais concretamente PHP.

PHP é uma linguagem de programação que permite a criação de conteúdos dinâmicos e, interage com bases de dados. É usada principalmente para o desenvolvimento de aplicações web.

Toda a história desta linguagem começou em 1994, quando Ramus Lerdorf lançou a primeira versão deste projeto Open Source que, ao longo do tempo, começou a ganhar cada vez mais adeptos. Espero que, com esta introdução e futuros exemplos/tutoriais, o leitor também se torne adepto.

CARACTERÍSTICAS DO PHP

A linguagem **PHP**, surgiu com a designação- **Personal Home Pages** (Ramus Lerdorf-1995);

Atualmente PHP surge com o significado de **Hypertext Preprocessor**;

PHP é uma linguagem de programação de scripts para serem interpretados no lado dos servidores;

O **PHP** é “incorporado” em documentos HTML;

O **PHP** permite a construção de páginas web com conteúdos dinâmicos;

O facto do **PHP** ser uma das linguagens do lado do servidor mais utilizadas na web deve-se essencialmente a:

- Ser uma tecnologia distribuída gratuitamente;
- Ter uma excelente capacidade de articulação com:
 - Servidores Web- Apache;
 - Sistemas de gestão de base de dados- MYSQL.

O QUE É PRECISO PARA INSTALAR O PHP?

Instalação em Windows

O PHP 5.5+ requer ao menos Windows 2008/Vista, ou 2008r2, 2012, 2012r2, 2016 ou 7, 8, 8.1, 10. 32-Bit ou 64-bit (aka X86 or X64; PHP não roda no Windows RT/WOA/ARM).

O PHP requer o Visual C runtime (CRT). Muitas aplicações também tem esse requisito, de forma que ele já deve estar instalado.

PHP 5.4, 5.5, e 5.6 requerem VC CRT 11 (Visual Studio 2012). Veja:
» <https://www.microsoft.com/en-us/download/details.aspx?id=30679>

PHP 7.0+ requer VC CRT 14 (Visual Studio 2015). Ver:
» <https://www.microsoft.com/en-us/download/details.aspx?id=48145>

Você PRECISA instalar o CRT x86 para compilar o PHP em x86 builds CRT x64 para compilações PHP x64.

Se o CRT já está instalado, o instalador irá avisar que não fez nenhuma alteração.

O instalador CRT suporta as opções de linha de comando /quiet e /norestart, de forma que você pode automatizar o passo de instalação.

DLLs do CRT VC11 podem ser copiadas da sua máquina local para uma máquina remota (uma instalação "Copy Deployment") em vez de rodar o instalador na máquina remota (como o caso de uma máquina onde tenha acesso restrito. Veja: <http://www.sitepoint.com/install-php53-windows/>

O CRT VC14 não suporta uma instalação installation "Copy Deployment". CRT VC14 tem muitas DLLs a mais (a maioria com nomes iniciando com api-*). Se você conseguir achar todas e copiá-las isso pode funcionar (tente uma ferramenta como Resource Hacker para obter uma lista de DLLs a copiar).

SÍN Taxe BÁSICA EM PHP

Quando o PHP interpreta um arquivo ele procura pelas tags de abertura e fechamento, `<?php` e `?>`, que dizem ao PHP para iniciar ou parar a interpretação do código entre elas. A interpretação desta maneira, permite o PHP ser incluído em vários tipos de documentos, pois tudo que está fora destas tags é ignorado pelo interpretador do PHP.

PHP inclui uma tag curta echo `<?=>` que é uma forma abreviada mais verbosa para `<?php echo`.

O PHP também permite a tag curta `<?>` (cujo uso é desencorajado pois essa opção está disponível somente quando habilitada na diretiva [short open tag](#) no arquivo de configuração `php.ini`, ou quando o PHP tiver sido compilado com a opção `--enable-short-tags`).

Se um arquivo for código PHP puro, é preferível omitir a tag de fechamento no final do arquivo. Prevenindo a existência de espaços ou linhas em branco após a tag, que podem causar efeitos indesejáveis, por que o PHP iniciará o buffer de saída quando não existir intenção do programador de enviar alguma saída neste ponto do script.

```
<?php  
echo "Hello world";  
  
// ... mais código  
  
echo "última instrução";  
  
// o script termina aqui, sem tag de fechamento PHP
```

Ilustração 1- Síntaxe básica em PHP

Comentário em PHP

O PHP suporta comentários no estilo 'C', 'C++' e do Unix shell (estilo Perl). Por exemplo:

```
<?php
    echo 'Isto é um teste'; // Estilo de comentário de uma linha em  c++
    /* Este é um comentário de múltiplas linhas
       ainda outra linha de comentário */
    echo 'Isto é ainda outro teste';
    echo 'Um teste final'; # Este é um comentário de uma linha no estilo shell
?>
```

Ilustração 2- Exemplo comentário em PHP

Os comentários de estilo "uma linha" apenas comentam até o final da linha ou do bloco PHP de código corrente, o que chegar primeiro. Isto significa que o código HTML após // ... ?> ou # ... ?> SERÁ impresso: ?> interrompe o modo PHP e retorna para o modo HTML, e // ou # não podem influenciar isto. Se a diretiva de configuração asp_tags estiver habilitada, se comportará da mesma maneira que // %> e # %>. Entretanto, a tag </script> não interrompe o modo PHP em um comentário de uma linha.

```
<h1>Isto é um <?php # echo 'simples';?> exemplo.</h1>
<p>O cabeçalho acima irá dizer 'Isto é um exemplo'.</p>
```

Ilustração 3-Comentário em linha

Comentários no estilo 'C' terminam ao primeiro */ encontrado. Tenha certeza de não aninhar comentários no estilo 'C'. É fácil fazer este equívoco se estiver tentando comentar grandes blocos de código.

```
<?php
/*
    echo 'Isto é um teste'; /* Este comentário irá causar um problema */
*/
?>
```

Ilustração 4-Comentário na linha

DECLARAÇÃO DE VARIÁVEIS

As vezes, é conveniente possuir variáveis com nomes variáveis. Isto é, o nome de uma variável que pode ser definido e utilizado dinamicamente. Uma variável normal é definida com uma instrução como:

```
<?php  
$a = 'hello';  
?>
```

Uma variável variável obtém o valor de uma variável e a trata como o nome de uma variável. No exemplo acima, hello pode ser utilizada como o nome de uma variável utilizando dois sinais de cifrão:

```
<?php  
$$a = "world";  
?>
```

Neste ponto, duas variáveis foram definidas e armazenadas na árvore de símbolos do PHP: \$a que contém "hello" e \$hello que contém "world". Assim, esta instrução:

```
<?php  
echo "$a ${$a}";  
?>
```

produz a mesma saída que:

```
<?php  
echo "$a $hello";  
?>
```

assim sendo, as duas produzem: hello world.

Para poder utilizar variáveis variáveis com arrays, você precisa resolver um problema de ambiguidade. Isso é, se você escrever \$\$a[1] o interpretador precisa saber que se deseja utilizar \$a[1] como uma variável ou que se deseja usar \$\$a como uma variável e [1] como o índice dessa variável. A sintaxe para resolver essa ambiguidade é \${\$a[1]} para o primeiro caso e \${\$a}[1], para o segundo.

Propriedades de classes também podem ser acessadas utilizando-se nomes de propriedades variáveis. O nome de propriedade variável será resolvido dentro do escopo em que a chamada foi feita. Por exemplo, se houver uma expressão como \$foo->\$bar, o escopo local será examinado procurando por \$bar e seu valor será utilizado como o nome da propriedade \$foo. Isso também funciona se \$bar for um array.

Tipos de dados

a) String:

Uma string literal pode ser especificada de quatro formas diferentes.

- aspas simples;
- aspas duplas ;
- sintaxe heredoc;
- sintaxe nowdoc (a partir do PHP 5.3.0).

Aspas simples ¶

A maneira mais simples de se especificar uma string é delimitá-la entre aspas simples (o caractere ').

Para especificar um apóstrofo, escape-o com uma contrabarra (\). Para especificar uma contrabarra literal, dobre-a (\\). Todas as outras ocorrências da contrabarra serão tratadas como uma contrabarra literal: isso significa que outras sequências de escape que se esteja acostumado a utilizar, como \r ou \n, serão literalmente impressas em vez de ter qualquer significado especial.

```
<?php
echo 'isto é uma string comum';

echo 'Você pode incluir novas linhas em strings,
dessa maneira que estará
tudo bem';

// Imprime: Arnold disse uma vez: "I'll be back"
echo 'Arnold disse uma vez: "I\'ll be back"';

// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C:\\*.*?';

// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C:\*.*?';

// Imprime: Isto não será substituído: \n uma nova linha
echo 'Isto não será substituído: \n uma nova linha';

// Imprime: Variáveis $também não $expandem
echo 'Variáveis $também não $expandem';
?>
```

Ilustração 5- Exemplo de string com aspas simples

b) Integer (Inteiros)

Inteiros podem ser especificados em notação decimal (base 10), hexadecimal (base 16), octal (base 8) ou binária (base 2). O operador de negação pode ser usado para indicar um inteiro negativo.

Inteiros binários literais estão disponíveis a partir do PHP 5.4.0.

Para usar a notação octal, preceda o número com um 0 (zero). Para utilizar a notação hexadecimal, preceda o número com 0x. Para utilizar a notação binária, preceda o número com 0b.

```
<?php
$a = 1234; // número decimal
$a = 0123; // número octal (equivalente a 83 em decimal)
$a = 0x1A; // número hexadecimal (equivalente a 26 em decimal)
$a = 0b11111111; // número binário (equivalente ao 255 decimal)
?>
```

Ilustração 6-Literais inteiras

Formalmente, as estruturas para inteiros literais são:

decimal : [1-9][0-9]*
| 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal : 0[0-7]+

binary : 0[bB][01]+

integer : decimal
| hexadecimal
| octal
| binary

O tamanho de um inteiro depende da plataforma, sendo um número aproximado a 2 bilhões o valor mais comum (número de 32 bits com sinal). Plataformas 64-bit possuem comumente o valor máximo de aproximadamente 9E18, exceto no Windows em versões anteriores ao PHP 7, onde são sempre 32-bit. O PHP não suporta inteiros sem sinal. O tamanho do inteiro pode ser determinado pela constante PHP_INT_SIZE, e seu o valor máximo com a constante PHP_INT_MAX, a partir do PHP 5.0.5, e o valor mínimo utilizando a constante PHP_INT_MIN a partir do PHP 7.0.0.

c) Booleanos

Para especificar um booleano literal, use as palavras-chave **TRUE** ou **FALSE**. Ambas são case-insensitive.

```
<?php
$foo = True; // atribui o valor True para $foo
?>
```

Tipicamente, o resultado de um operador que retorne um valor booleano, é passado para uma estrutura de controle.

```
<?php
// == É um operador que testa
// igualdade e retorna um booleano.
if ($action == "mostrar-versao") {
    echo "A versão é 1.23";
}

// isto não é necessário ...
if ($exibir_separadores == TRUE) {
    echo "<hr>\n";
}

// ... porque você pode simplesmente escrever isso:
if ($exibir_separadores) {
    echo "<hr>\n";
}
?>
```

d) Array

Um array pode ser criado com o construtor de linguagem `array()`. Ele leva qualquer quantidade de pares separados por vírgula chave => valor como argumentos.

```
array(  
    chave => valor,  
    chave2 => valor2,  
    chave3 => valor3,  
    ...)
```

A vírgula após o último elemento do array é opcional e pode ser omitida. Isso geralmente é feito para arrays em uma única linha, por exemplo, `array(1, 2)` é preferível em vez de `array(1, 2, )`. Para arrays escritos em várias linhas, por outro lado, a vírgula a direita do último elemento é usada frequentemente, para permitir a adição de forma simples de novos elementos ao final.

A partir do PHP 5.4 você também pode utilizar a sintaxe contraída de array, que troca `array()` por `[]`.

```
<?php  
$array = array(  
    "foo" => "bar",  
    "bar" => "foo",  
);  
  
// a partir do PHP 5.4  
$array = [  
    "foo" => "bar",  
    "bar" => "foo",  
];  
?>
```

Ilustração 7- Um array simples

A chave pode ser um inteiro ou uma string. O valor pode ser de qualquer tipo.



e) Objetos

Para criar um novo objeto, utilize a instrução `new` para instanciar uma classe:

```
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

Para uma discussão completa, veja o capítulo **Classes e Objetos**.

f) NULL

Há apenas um valor do tipo null, a constante case-insensitive `NULL`.

```
<?php
$var = NULL;
?>
```

Veja também as funções `is_null()` e `unset()`.

g) Resources

Como as variáveis `resource` mantêm manipuladores especiais para arquivos abertos, conexões de bancos de dados, canvas de imagens e coisas do tipo, converter para `resource` não faz sentido algum.

INTERPRETAÇÃO DE STRINGS

Quando uma string é especificada dentro de aspas duplas ou heredoc, as variáveis são interpretadas dentro delas.

Há dois tipos de sintaxe: uma simples e uma complexa. A sintaxe simples é a mais comum e conveniente. Provê uma maneira de interpretar uma variável, o valor de um array ou uma propriedade de objeto em uma string com o mínimo de esforço.

A sintaxe complexa pode ser reconhecida pelas chaves envolvendo a expressão.

Sintaxe simples

Se um sinal de cifrão (\$) for encontrado, o interpretador tentará obter tantos identificadores quantos forem possíveis para formar um nome de variável válido. Envolve o nome da variável com chaves para especificar explicitamente o fim do nome.

```
<?php
$juice = "apple";

echo "He drank some $juice juice.".PHP_EOL;
// Invalid. "s" is a valid character for a variable name, but the variable is $juice.
echo "He drank some juice made of $juices.";
// Valid. Explicitly specify the end of the variable name by enclosing it in braces:
echo "He drank some juice made of ${juice}s.";
?>
```

O exemplo acima irá imprimir:

```
He drank some apple juice.
He drank some juice made of .
He drank some juice made of apples.
```

Funções das Strings

- **addslashes** — String entre aspas com barras no estilo C
- **addslashes** — Adiciona barras invertidas a uma string
- **bin2hex** — Converte um dado binário em representação hexadecimal
- **chop** — Sinónimo de rtrim
- **chr** — Retorna um carácter específico
- **chunk_split** — Divide uma string em pequenos pedaços
- **convert_cyr_string** — Converte de um conjunto caracteres cirílico para outro
- **convert_uuencode** — Decodifica uma string codificada com uuencode
- **convert_uuencode** — Codifica com uuencode uma string
- **count_chars** — Retorna informações sobre os caracteres usados numa string
- **crc32** — Calcula polinómio crc32 de uma string
- **crypt** — Encriptação unidirecional de string (hashing)
- **echo** — Exibe uma ou mais strings
- **explode** — Divide uma string em strings
- **fprintf** — Escreve uma string formatada para um stream
- **get_html_translation_table** — Retorna a tabela de tradução usada por

htmlspecialchars e htmlentities

- **hebrew** — Converte texto lógico Hebraico para texto visual
- **hebrevc** — Converte um texto lógico Hebraico para um texto visual com

conversão newline

- **hex2bin** — Decodes a hexadecimally encoded binary string
- **html_entity_decode** — Converte todas as entidades HTML para os seus

caracteres

- **htmlentities** — Converte todos os caracteres aplicáveis em entidades html.

- **htmlspecialchars_decode** — Converte especiais entidades HTML para

caracteres

- **htmlspecialchars** — Converte caracteres especiais para a realidade HTML

- **implode** — Junta elementos de uma matriz em uma string

- **join** — Sinónimo de implode

- **lcfirst** — Torna minúsculo o primeiro caractere de uma string

- **levenshtein** — Calcula a distância Levenshtein entre duas strings

- **localeconv** — Obtém a informação da formatação numérica

- **ltrim** — Retira espaços em branco (ou outros caracteres) do início da string

- **md5_file** — Calcula hash md5 de um dado arquivo



- **md5** — Calcula o "hash MD5" de uma string
- **metaphone** — Calcula a metaphone key de uma string
- **money_format** — Formata um número como uma string de moeda
- **nl_langinfo** — Retorna informação de linguagem e local
- **nl2br** — Insere quebras de linha HTML antes de todas newlines em uma string
- **number_format** — Formata um número com os milhares agrupados
- **ord** — Retorna o valor ASCII do caractere
- **parse_str** — Converte a string em variáveis
- **print** — Mostra uma string
- **printf** — Mostra uma string formatada
- **quoted_printable_decode** — Converte uma string quoted-printable para uma string de 8 bit
- **quoted_printable_encode** — Converte uma string de 8 bits em uma string quoted-printable
- **quotemeta** — Adiciona uma barra invertida antes dos meta caracteres
- **rtrim** — Retira espaço em branco (ou outros caracteres) do final da string
- **setlocale** — Define informações locais
- **sha1_file** — Calcula a hash sha1 de um arquivo
- **sha1** — Calcula a hash sha1 de uma string
- **similar_text** — Calcula a similaridade entre duas strings
- **soundex** — Calcula a chave soundex de uma string
- **sprintf** — Retorna a string formatada
- **sscanf** — Interpreta a entrada de uma string de acordo com um formato
- **str_getcsv** — Analisa uma string CSV e retorna os dados em um array
- **str_ireplace** — Versão que não diferencia maiúsculas e minúsculas de str_replace.
- **str_pad** — Preenche uma string para um certo tamanho com outra string
- **str_repeat** — Repete uma string
- **str_replace** — Substitui todas as ocorrências da string de procura com a string de substituição
- **str_rot13** — Executa a transformação rot13 em uma string
- **str_shuffle** — Mistura uma string aleatoriamente
- **str_split** — Converte uma string para um array
- **str_word_count** — Retorna informação sobre as palavras usadas em uma string
- **strcasecmp** — Comparação de strings sem diferenciar maiúsculas e minúsculas segura para binário



- **strchr** — Sinónimo de strstr
- **strcmp** — Comparação de string segura para binário
- **strcoll** — Comparação de string baseada no local
- **strcspn** — Encontra o tamanho do segmento inicial que não contenha a máscara
- **strip_tags** — Retira as tags HTML e PHP de uma string
- **stripslashes** — Desfaz o efeito de addslashes
- **stripos** — Encontra a primeira ocorrência de uma string sem diferenciar maiúsculas e minúsculas
- **stripslashes** — Desfaz o efeito de addslashes
- **stristr** — strstr sem diferenciar maiúsculas e minúsculas
- **strlen** — Retorna o tamanho de uma string
- **strnatcasecmp** — Comparação de strings sem diferenciar maiúsculas/minúsculas usando o algoritmo "natural order"
- **strnatcmp** — Comparação de strings usando o algoritmo "natural order"
- **strncasecmp** — Comparação de string caso-sensitivo de Binário seguro dos primeiros n caracteres
- **strncmp** — Comparação de string segura para binário para os primeiros n caracteres
- **strpbrk** — Procura na string por um dos caracteres de um conjunto
- **strpos** — Encontra a posição da primeira ocorrência de uma string
- **strrchr** — Encontra a última ocorrência de um caractere em uma string
- **strrev** — Reverte uma string
- **strrpos** — Encontra a posição da última ocorrência de uma string case-insensitive em uma string
- **strrpos** — Encontra a posição da última ocorrência de um caractere em uma string
- **strspn** — Encontra o comprimento do segmento inicial combinando com a máscara
- **strstr** — Encontra a primeira ocorrência de uma string
- **strtok** — Tokeniza uma string
- **strtolower** — Converte uma string para minúsculas
- **strtoupper** — Converte uma string para maiúsculas
- **strtr** — Traduz certos caracteres
- **substr_compare** — A comparação binária entre duas strings de um offset até o limite do comprimento de caracteres
- **substr_count** — Conta o número de ocorrências de uma substring

- **substr_replace** — Substitui o texto dentro de uma parte de uma string
- **substr** — Retorna uma parte de uma string
- **trim** — Retira espaço no início e final de uma string
- **ucfirst** — Converte para maiúscula o primeiro caractere de uma string
- **ucwords** — Converte para maiúsculas o primeiro caractere de cada palavra
- **vfprintf** — Escreve uma string formatada para um stream
- **vprintf** — Mostra uma string formatada
- **vsprintf** — Retorna uma string formatada
- **wordwrap** — Quebra uma string em um dado número de caracteres



CONSTANTES

Uma constante é um identificador (nome) para um valor único. Como o nome sugere, esse valor não pode mudar durante a execução do script (exceto as constantes mágicas, que não são constantes de verdade). As constantes são case-sensitive por padrão. Por convenção, identificadores de constantes são sempre em maiúsculas.

O nome de uma constante tem as mesmas regras de qualquer rótulo do PHP. Um nome válido de constante começa com uma letra ou sublinhado, seguido por qualquer número de letras, números ou sublinhados. Como uma expressão regular, pode ser representada por: `^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$`

É permitido utilizar `define()` em constantes com, nomes reservados ou mesmo inválidos, cujo valor pode (apenas) ser recuperado com `constant()`. No entanto, isso não é recomendado.

```
<?php
```

```
// Nomes de constantes válidos
```

```
define("FOO", "alguma coisa");
```

```
define("FOO2", "alguma outra coisa");
```

```
define("FOO_BAR", "alguma coisa mais");
```

```
// Nomes de constantes inválidas
```

```
define("2FOO", "alguma coisa");
```

```
// Isto é válido, mas deve ser evitado:
```

```
// O PHP pode vir a fornecer uma constante mágica
```

```
// que danificará seu script
```

```
define("__FOO__", "alguma coisa");
```

```
?>
```

Como as superglobals, o escopo de uma constante é global. Você pode acessar constantes de qualquer lugar em seu script sem se preocupar com o escopo. Para mais informações sobre o escopo no PHP, leia a seção do manual escopo de variáveis.

OPERADORES

I. Operadores aritméticos

Exemplo	Nome	Resultado
$+ \$a$	Identidade	Conversão de $\$a$ para <code>int</code> ou <code>float</code> conforme apropriado.
$- \$a$	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 $\$a$ e $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido por $\$b$.
$\$a ** \b	Exponencial	Resultado de $\$a$ elevado a $\$b$. Introduzido no PHP 5.6.

II. Operadores Bit-a-bit

Exemplo	Nome	Resultado
$\$a \& \b	E (AND)	Os bits que estão ativos tanto em $\$a$ quanto em $\$b$ são ativados.
$\$a \b	OU (OR inclusivo)	Os bits que estão ativos em $\$a$ ou em $\$b$ são ativados.
$\$a \wedge \b	XOR (OR exclusivo)	Os bits que estão ativos em $\$a$ ou em $\$b$, mas não em ambos, são ativados.
$\sim \$a$	NÃO (NOT)	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")

III. Operadores de 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.
$\$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 $\$b$.
$\$a <=> \b	Spaceship (nave espacial)	Um <code>integer</code> menor que, igual a ou maior que zero quando $\$a$ é, respectivamente, menor que, igual a ou maior que $\$b$. Disponível a partir do PHP 7.

IV. Operadores de Incremento/Decremento

Exemplo	Nome	Efeito
<code>++\$a</code>	Pré-incremento	Incrementa \$a em um, e então retorna \$a.
<code>\$a++</code>	Pós-incremento	Retorna \$a, e então incrementa \$a em um.
<code>--\$a</code>	Pré-decremento	Decrementa \$a em um, e então retorna \$a.
<code>\$a--</code>	Pós-decremento	Retorna \$a, e então decrementa \$a em um.

V. Operadores Lógicos

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

VI. Operadores de Array

Exemplo	Nome	Resultado
<code>\$a + \$b</code>	União	União de \$a e \$b.
<code>\$a == \$b</code>	Igualdade	TRUE se \$a e \$b tem os mesmos pares de chave/valor.
<code>\$a === \$b</code>	Identidade	TRUE se \$a e \$b tem os mesmos pares de chave/valor na mesma ordem e do mesmo tipo.
<code>\$a != \$b</code>	Desigualdade	TRUE se \$a não é igual a \$b.
<code>\$a <> \$b</code>	Desigualdade	TRUE se \$a não é igual a \$b.
<code>\$a !== \$b</code>	Não identidade	TRUE se \$a não é idêntico a \$b.

ESTRUTURAS DE CONTROLO

A. IF

O construtor if é um dos recursos mais importantes em muitas linguagens, inclusive no PHP. Permite a execução condicional de fragmentos de código. O PHP apresenta uma estrutura if semelhante a do C:

```
if (expr)
    statement
```

Como descrito na seção sobre expressões, expressões são avaliadas por seus valores Booleanos. Se uma expressão for avaliada como TRUE, o PHP executará a declaração, e se avaliá-la como FALSE - ignorá-la. Mais informações sobre quais valores são avaliados como FALSE pode ser encontrada na seção 'Conversão para booleano'.

O exemplo a seguir exibirá a is bigger than b se \$a for maior que \$b:

```
<?php
if ($a > $b)
    echo "a is bigger than b";
?>
```

B. ELSE

Muitas vezes deseja-se executar uma instrução se uma certa condição for válida, e uma instrução diferente se a mesma condição não for válida. Para isso que o else serve. O else estende a instrução if para executar outras caso a expressão no if retornar FALSE. Por exemplo, o código a seguir exibirá a is greater than b se \$a for maior que \$b, e a is NOT greater than b caso contrário:

```
<?php
if ($a > $b) {
    echo "a is greater than b";
} else {
    echo "a is NOT greater than b";
}
?>
```

A instrução else só é executada se a expressão de avaliação do if for avaliada como FALSE, e se tiver qualquer expressão elseif - somente se também retornarem FALSE.

C. ELSEIF

Elseif, como o nome sugere, é uma combinação do if e else. Como o else, estende um if para executar instruções diferentes no caso da expressão if original ser avaliada como FALSE. Entretanto, diferentemente do else, executará uma expressão alternativa somente se a expressão condicional do elseif for avaliada como TRUE. Por exemplo, o código a seguir exibirá a is bigger than b, a equal to b ou a is smaller than b:

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
?>
```

Pode haver vários elseifs dentro da mesma declaração if. A primeira expressão elseif (se houver) que retornar TRUE será executada. No PHP, pode-se escrever 'else if' (em duas palavras), e o comportamento será idêntico ao do 'elseif' (em uma única palavra). O significado sintático é um pouco diferente (se você está familiarizado com C, o comportamento é o mesmo), mas, no fundo, ambos terão exatamente o mesmo comportamento.

O elseif só é executado se o if precedente ou qualquer elseif for avaliado como FALSE, e o elseif corrente for avaliado como TRUE.

```
<?php

/* Incorrect Method: */
if ($a > $b):
    echo $a." is greater than ".$b;
else if ($a == $b): // Will not compile.
    echo "The above line causes a parse error.";
endif;

/* Correct Method: */
if ($a > $b):
    echo $a." is greater than ".$b;
elseif ($a == $b): // Note the combination of the words.
    echo $a." equals ".$b;
else:
    echo $a." is neither greater than or equal to ".$b;
endif;

?>
```

CONCLUSÃO

Em suma, a linguagem PHP é uma linguagem de domínio específico, ou seja, estende-se a um campo de atuação que é a WEB, embora tenha variáveis como o OHP-GTK.

O objetivo principal da linguagem PHP é implementar soluções WEB rápidas, simples e eficientes.



WEBGRAFIA

[HTTP://PT.WIKIPEDIA.ORG/WIKI](http://pt.wikipedia.org/wiki)

[HTTP://WWW.APOSTILANDO.COM](http://www.apostilando.com)

[HTTP://WWW.W3SCHOOLS.COM/PHP/](http://www.w3schools.com/php/)