# PHP : Hypertext Preprocessor

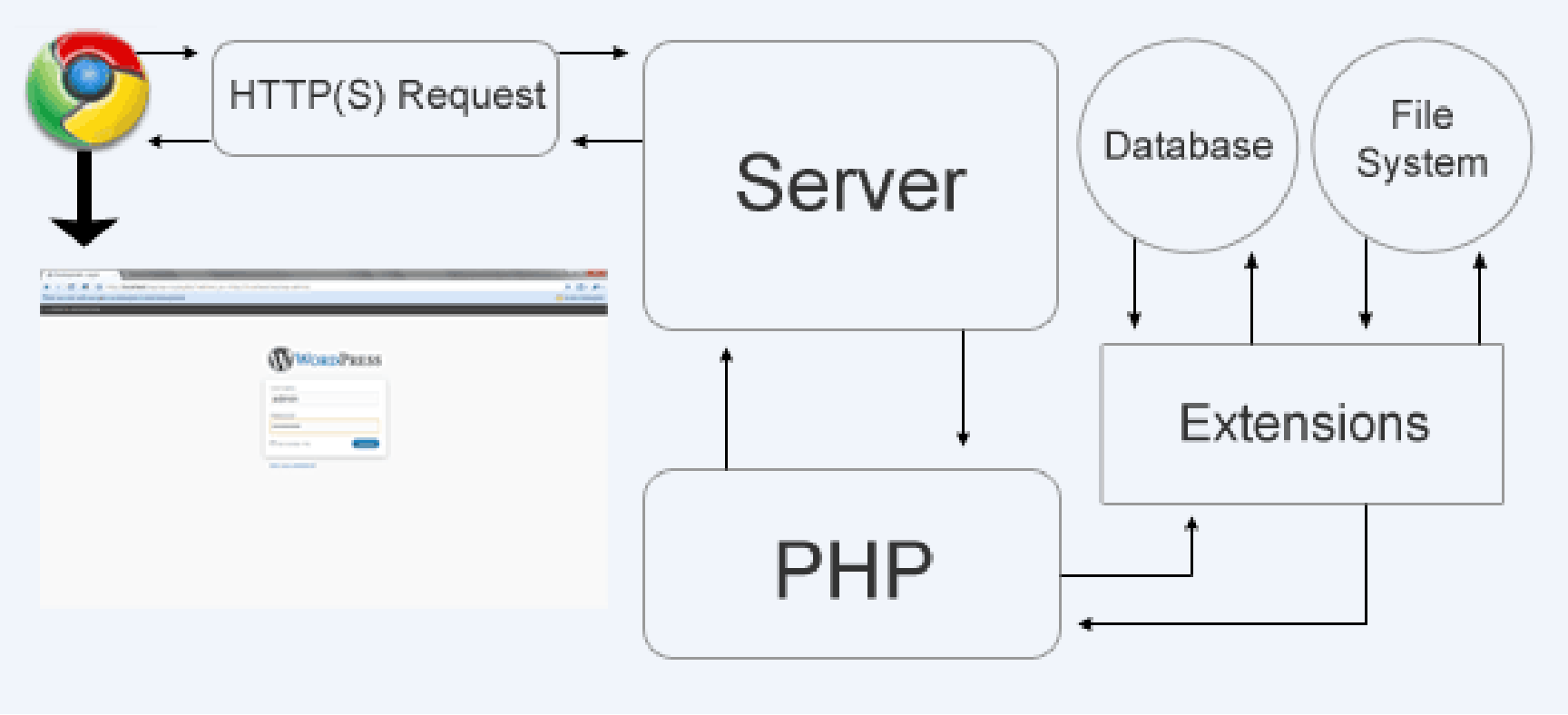
Seminário Linguagens de Programação

Allan Araújo
Eduardo Oliveira

# Introdução


O PHP é uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML.

O que distingue o PHP de algo como o Javascript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o cliente. O cliente recebe os resultados da execução desse script, mas não sabe qual era o código fonte.


HTTP(S) Request
Server
Database
File System
Extensions
PHP

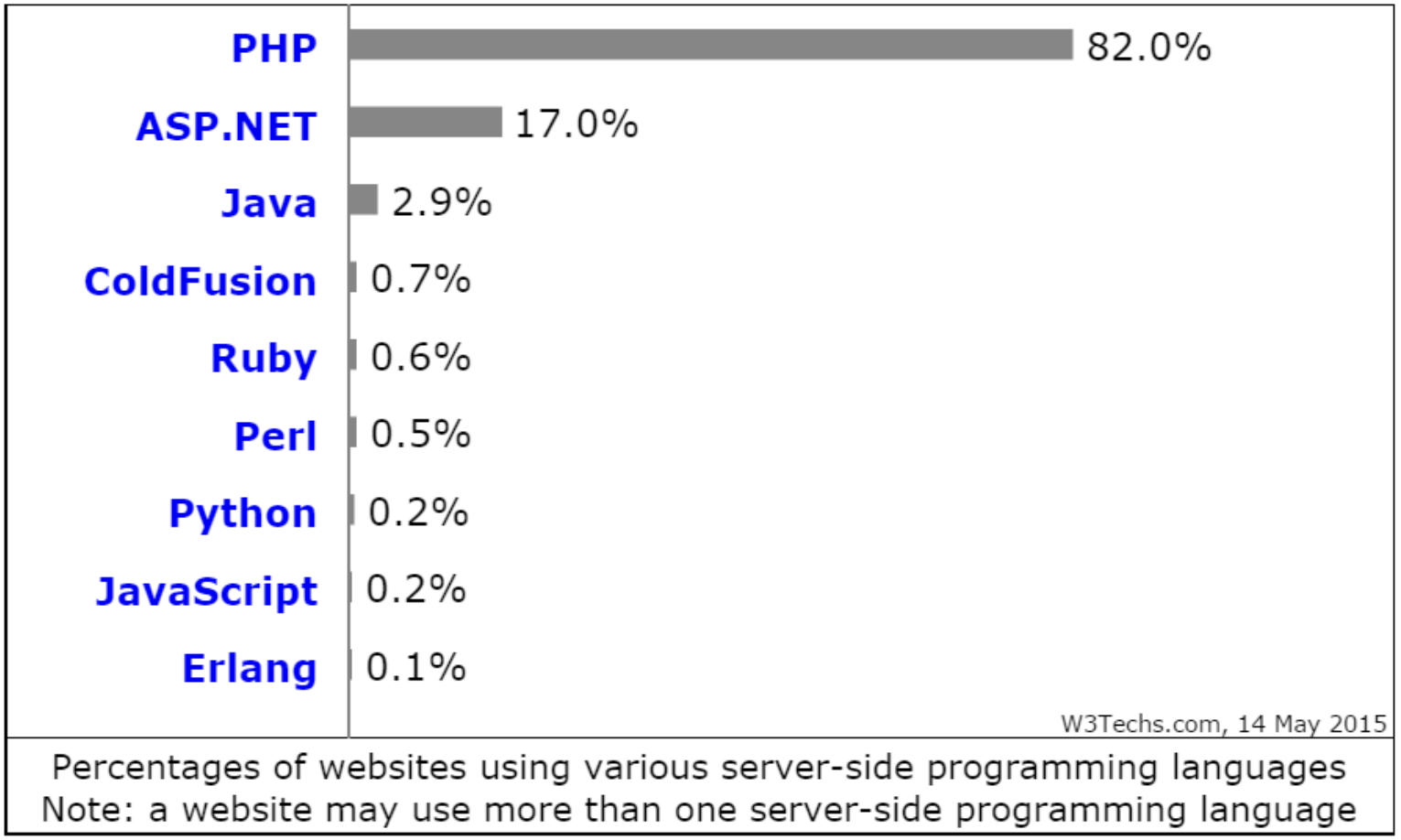
# Breve Histórico

- 1994: Rasmus Ledorf começa o desenvolvimento de um conjunto de ferramentas para aplicações web nomeado de *Personal Home Page Tools*.
- 1995: PHP 1.0 – Código fonte liberado, rebatizado de "*Personal Home Page Contruction Kit*";
- 1996: PHP 2.0 – Evolução para linguagem de script. Acrescido suporte a banco de dados, *cookies*, envio condicional de blocos HTML, etc.

# Breve Histórico

- 1998: PHP 3.0 - *PHP: Hypertext Preprocessor*. Novos recursos de extensibilidade e de orientação à objetos.
- 2000: PHP 4.0 - Motor 'Zend Engine', melhoria no desempenho, mudanças na arquitetura visando segurança, suporte a mais servidores.
- 2004: PHP 5.0 - Core Zend Engine 2.0 e Melhorias orientação a objetos.
- Obs: PHP 5.6.8 é a versão estável atual.

# Quem usa PHP?



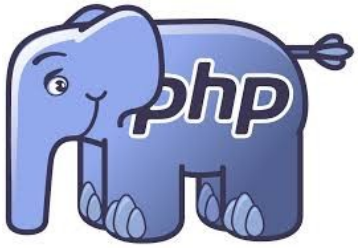

Referência: http://w3techs.com/technologies/overview/programming_language/all

# Quem usa PHP?

- Facebook.com
- Yahoo.com
- Flickr.com
- Wikipedia.org
- Twitter.com
- WordPress.com
- Joomla.org
- Drupal.org
- Moodle.org
- Baidu.com

# LAMP

# Mini Tutorial



# Sintaxe Básica - Exemplo ola.php

```
<html>

   <head>
         <title>PHP Teste</title>
   </head>

   <body>

        <?php
             echo "<p>Olá Mundo</p>";
        ?>

   </body>

</html>
```

# Sintaxe Básica - Exemplo myAge.php

```
<?php

function myAge($birthYear)
{
  $yearsOld = date('Y') - $birthYear;
  return $yearsOld . ' year' . ($yearsOld != 1? 's':'');
}

echo 'I am currently ' . myAge(1981) . ' old.';

?>
```

Exemplo de saída: 'I am currently 34 years old.'

# Sintaxe Básica

Marcadores:

```
<?php [um script em php] ?>
```

Separação de instruções:

```
<?php
    echo 'Isto é um teste';
?>

<?php echo 'Isto é um teste' ?>
```

Obs.: Espaços em branco e quebras de linha não interferem na interpretação da linguagem.

# Sintaxe Básica - Comentários

Comentário de uma linha estilo C:

```
//comentário
```

Comentário de múltiplas linhas:

```
/*múltiplas linhas*/
```

Comentário estilo *shell*:

```
# comentário
```

# Sintaxe Básica - Case Sensitivity

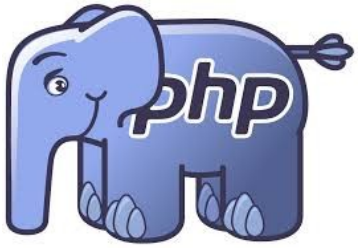
Classes, funções e palavras-chave como, echo, while, class, etc, são case-insensitive:

```
echo("Olá mundo");
ECHO("Olá mundo");
EcHo("Olá mundo");
```

Contudo, variáveis são case-sensitive:

$name, $NAME e $NaME representam variáveis diferentes.

# Tipos de dados



# Tipos de Dados Primitivos

**Escalares**:
boolean: true, false (case-insensitive);
integer: 123, -123, 0123, 0x1A;
float(double): 1.234, 1.2e3, 7E-10;
string: 'teste', “teste”;

**Compostos**:
array;
object;

**Especiais**:
resource;
null (case-insensitive);

# Array - Coleção

```
<?php
 $cores = array('vermelho','azul','verde','amarelo');

 foreach ($cores as $cor) {
    echo "Você gosta de $cor?\n";
 }
?>
```

**Saída:**

```
Você gosta de vermelho?
Você gosta de azul?
Você gosta de verde?
Você gosta de amarelo?
```

# Arrays Associativos

Definição:

```
array(  chave =>  valor , ... )
// chave pode ser tanto string ou um integer
// valor pode ser qualquer coisa
```

Exemplo:

```
<?php
  $arr = array("foo" => "bar", 12 => true);

  echo $arr["foo"]; // bar
  echo $arr[12];    // 1
?>
```

# Arrays Multidimensionais

```
<?php

$m = array(
      "Fulano" => array("rg" => "00.000.000-1",
"cpf" => "000.000.000-01"),
      "Ciclano" => array("rg" => "10.100.100-X",
"cpf" => "100.100.100-01"),
      "Beltrano" => array("rg" => "11.111.111-1",
"cpf" => "111.111.111-01")
    );

?>
```

# Tipos de Dados - Resource

Variável especial que mantém referência de recurso externo, como manipuladores especiais para arquivos abertos ou conexões com bancos de dados.

```
<?php

 $mysql_access = mysql_connect($server, $user, $pw);

 if( is_resource( $mysql_access))
  echo ”A variável $mysql_access é do tipo
resource.”;

?>
```

# Inferência

O tipo de uma variável geralmente não é definido pelo programador: isto é decidido em tempo de execução pelo PHP, dependendo do contexto na qual a variável é usada

```
<?php
  $foo = "0";  //$foo é string (ASCII 48)
  $foo += 2;   //$foo é agora um inteiro (2)
  $foo = $foo + 1.3;  //$foo é agora um float (3.3)

  $foo = 5 + "10 pequenos porcos";
  //$foo é inteiro (15)

  $foo = 5 + "10 minúsculos porcos";
  //$foo é inteiro (15)
?>
```

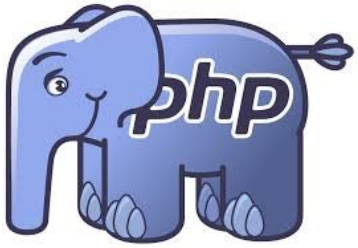
# Conversão de Tipos (Type Casting)

```
<?php
  $foo = 10;               // $foo é um inteiro
  $bar = (boolean) $foo; // $bar é um booleano
?>

<?php
 $foo = 10;            // $foo é um inteiro
 $str = "$foo";        // $str é uma string
 $fst = (string) $foo; // $fst tambem é uma string

 // Isto imprimira "eles são o mesmo"
 if ($fst === $str) {
    echo "eles são o mesmo";
 }
?>
```

# Inferência – gettype() e is_<tipo>()

```
<?php
   $a_bool = TRUE;   // um booleano
   $a_str  = "foo";  // uma string
   $a_str2 = 'foo';  // uma string
   $an_int = 12;     // um inteiro

   echo gettype($a_bool); // mostra:  boolean
   echo gettype($a_str);  // mostra:  string

   if (is_int($an_int)) { //Se ele é um inteiro, incrementa-o com
  quatro
        $an_int += 4;
   }
   if (is_string($a_bool)) {  // Se $bool é uma string, mostre-a
        echo "String: $a_bool";
   }
?>
```

# Variáveis e constantes



# Variáveis e Constantes

Constantes e variáveis são case-sensitive por padrão. Um nome válido de variável ou constante começa com uma letra ou sublinhado, seguido por qualquer número de letras, números ou sublinhados.

Variáveis são declaradas usando cifrão ($):

```
$_cpf; $curso1; $nome_completo;
```

Por convenção, identificadores de constantes são sempre em maiúsculas:

```
define(“PI”,”3.1415”);// global
const MAX_VALUE = 10.0;  // local
```

# Variáveis – Atribuição por referência

Uso do e-comercial (&).

```
<?php

$foo = 'Bob'; //Atribui o valor 'Bob' a variável $foo
$bar = &$foo; // Referecia $foo através de $bar.
$bar = "My name is $bar"; // Altera $bar...

echo $bar;
echo $foo;  //$foo é alterada também.

?>
```

# Escopo de variável

O escopo de uma variável é o contexto onde ela foi definida. Usualmente, definimos a maior parte das variáveis com escopo local. Se criamos uma nova função, então temos um novo escopo;

Para termos acesso à essa variável em outros escopos, a definimos como **global** $var;

# Escopo local e global

```
$a = 3;
function f1() {
   $a += 2;
}

f1();
echo $a;
```

Resultado: 3

```
$a = 3;
function f2() {
   global $a;
   $a += 2;
}
f2();
echo $a;
```

Resultado: 5

# Variáveis Superglobais

Várias variáveis pré-definidas no PHP são "superglobais", que significa que elas estão disponíveis em todos escopos pelo script. Não há necessidade de fazer global $variavel para acessá-las dentro de funções ou métodos;

```
Ex: $GLOBALS, $SERVER, $FILE;
```

# Operadores

- Aritméticos;
- de Atribuição;
- Bit-a-bit;
- de Comparação;
- de Controle de erro;
- de Execução;
- de Incremento/decremento;
- Lógicos;
- de String;
- de Arrays;
- de Tipo;

# Operadores

Bit-a-Bit:

```
$valor = 1;
$valor= $valor << 1; // shift left 1 bit

echo  $var1 = 3 & $var2 = 10;
//3 AND 10 = 2

echo  $var1 = 3 ^ $var2 = 10;
//3 XOR 10 = 9
```

# Operadores

Comparação:

`$a === $b:`
Retorna TRUE caso $a seja idêntico a $b e forem do mesmo tipo;

`$a ?? $b ?? $c`
Retorna o primeiro operando (da direita para esquerda) diferente de NULL. Caso contrário, retorna NULL;

# Controle de Erro

O PHP suporta um operador de controle de erro: o sinal 'arroba' (@) – *silence operator*. Quando ele precede uma expressão em PHP, qualquer mensagem de erro que possa ser gerada por aquela expressão será ignorada;

```
$vogais = array("a","e","i","o","u");
echo $vetor[$chave];
//Notice: variavel $chave indefinida

$vogais = array("a","e","i","o","u");
echo @$vetor[$chave];
//Parte do codigo ignorada
```

# Estruturas de Controle

- if;
- switch;
- while;
- for;
- foreach;
- die and return;
- goto;

# Estruturas de Controle - if

Forma casual:

```
if ($usuario_valido){
   echo "Bem-Vindo!";
   $saudados = 1;
}
else {
   echo "Accesso Negado!";
   exit;
}
```

# Estruturas de Controle - if

Formas alternativas:

```
<?php
   echo $ativado ? "yes": ”no";
?>

if ($usuario_valido):
    echo “Bem-Vindo!";
    $saudados = 1;
else: echo “Acesso Negado!";
      exit;
endif;
```

# Estruturas de Controle - switch

Forma casual:

```
switch($id) {
    case 1:
    // faz algo
    break;
    case 2:
    // faz algo
    // (...)
    default
    // faz algo
}
```

Forma alternativa:

```
switch($id):
    case 1:
    // faz algo
    break;
    case 2:
    // faz algo
    // (...)
    default
    // faz algo
endswitch;
```

# Estruturas de Controle - while

Forma casual:

```
$total = 0;
$i = 1;

while ($i <= 10) {
    $total += $i;
    $i++;
}
```

Forma alternativa:

```
$total = 0;
$i = 1;

while ($i <= 10):
    $total += $i;
    $i++;
endwhile;
```

# Estruturas de Controle - for

Forma casual:

```
$total = 0;
for ($i= 1; $i <= 10; $i++) {
    $total += $i;
}
```

Forma alternativa:

```
for ($i = 1; $i <= 10; $i++):
    $total += $i;
endfor;
```

# Estruturas de Controle - foreach

Nos permite iterar sobre elementos de um array;

```
$vetor = array(1,2,3,4);
foreach ($array as $elem) {
    $elem = $elem * 2;
} //$vetor = array(2,4,6,8)

unset(elem);
```

Após o **foreach,** faz-se **unset**($elem) para destruir a referência criada para essa variável;

# Estruturas de Controle - foreach

Forma alternativa:

```
$vetor = array(1,2,3,4);
foreach ($array as $elemento):
   $elem = $elem * 2;
endforeach;

unset($elem);
```

# Estruturas de Controle - die(exit)

```
$db = mysql_connect("localhost", $USERNAME,
$PASSWORD);
if (!$db) {
   die("Could not connect to database");
}
```

Outra forma:

```
$db = mysql_connect("localhost", $USERNAME,
$PASSWORD)
or die("Could not connect to database");
```

# Estruturas de Controle - return

- Funções podem retornar algum valor;
- Palavra reservada **return;**

```
function multa($valor) {
   return $valor * 1,2;
}
```

# Estruturas de Controle – goto

```
for ($i = 0; $i < $count; $i++) {
// encontra algum erro
   if ($erro) {
      goto tratador;
   }
}
tratador:
//trata o erro;
```

# Include  e Require

PHP fornece duas construções para carregar scripts de outros módulos: include e require;

Principal diferença:

**Include:** produz uma advertência, mas não para a execução do script;

```
<?php include “arquivo1.html"; ?>
```

**Require:**  gera erro fatal, fim da execução do script;

```
<?php require "codelib.php"; ?>
```

# Include  e Require

**Problema:**
Ocorrer include/require mais de uma vez do mesmo arquivo, gerando erro, redefinição de funções ou múltiplas cópias do arquivo;

**Solução:**
include_once e require_once;

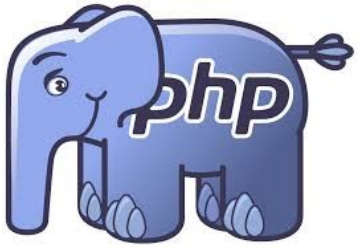
Possuem o mesmo comportamento de include e require, porém ignoram tentativas de carregamento de um mesmo arquivo;

Curiosidade!


PHP
>>
PHP
>>
php


elePHPant

# Funções



# Funções

- Nome de funções são case-insensitive;
- Palavra reservada **function**;
- Possuem escopo global em PHP, ou seja, podem ser chamadas fora de uma função, mesmo que tenham sido definidas dentro dela;
- O PHP não suporta sobrecarga de funções, e também não é possível cancelar ou alterar a definição de funções previamente declaradas;

# Passagem de parâmetros

Por cópia:

```
function duplica($valor) {
   $valor= $valor << 1;
}
$a = 3;
DUPLICA($a);
echo $a;
```

Saída: 3

# Passagem de parâmetros

Por referência:

```
function duplica1(&$valor) {
   $valor= $valor << 1;
}
$b = 3;
duplica1($b);
echo $b;
```

Saída: 6

# Passagem de parâmetros

Default:

```
function imprimeDefault($a = “Nada”) {
   echo $a;
}

imprimeAlgo(); //imprime “Nada”

imprimeAlgo(“Ola mundo!”);
//imprime “Ola mundo!”
```

Quando a função tem mais de um parâmetro, o valor default deve ser declarado por último;

# Classes e Objetos

- PHP também suporta programação OO;
- Definição de Classes: palavra reservada **class**;
- Criação de objeto: palavra reservada **new**;
- Acesso aos métodos e propriedades dos objetos: **“->”;**
- Suporta *namespaces*, construtores e destrutores de objetos;

# Exemplo: Classe Pessoa

```
class Pessoa {
//propriedades
   protected $nome;
   protected $idade;

//métodos
   public function __construct($nome,
   $idade){
      $this->nome = $nome;
      $this->idade = $idade;
   }

   public function __destruct(){}
}
```

# Exemplo: Classe Pessoa

```
   public function imprime(){
      echo "Nome: " . $this->nome.
“<br> Idade: ".$this->idade.“<br>";
   }
}

$p = new Pessoa(“Maria", 23);
$p->imprime();
```

Saída:

```
Nome: Maria
Idade: 23
```

# Manipulação de Arquivos

**fopen()**: Abre o arquivo para que possa ser manipulado;

**gets()**: Pega uma linha do arquivo (até 1024 bytes);

**feof()**: Usada durante a leitura para verificar se chegou ao final;

**file_get_contents**(): Pega todo o conteúdo do arquivo aberto como uma string;

**ftruncate**(): Reduz o tamanho do arquivo. Usado para apagar seu conteúdo;

**fwrite**(): Escreve no arquivo;

**unlink**(): apaga o arquivo indicado;

# Manipulação de Arquivos

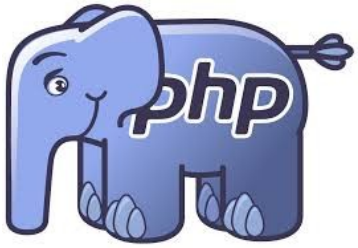
```
<?php
$arquivo = fopen('meuarquivo.txt','r');
 if ($arquivo == false)
   die('O arquivo não existe.');

while(!feof($arquivo)) { echo
fgets($arquivo). '<br />'; }
fclose($arquivo);
?>
```

# Manipulação de Arquivos

Modos de acesso:

“w”: Apenas escrita;
“r”: Apenas leitura;
“w+” ou “r+”: Leitura e escrita;
“a”: Apenas escrita, com ponteiro no fim do arquivo;
“a+”: Leitura e escrita, com ponteiro no fim do arquivo;

# Aspectos de Linguagem de Programação



# Paradigmas da Linguagem

- PHP suporta os seguintes paradigmas: estruturado, funcional, orientado a objeto e reflexivo;

- Mais utilizado adotando paradigmas estruturado e orientado a objetos;

# Propriedades desejáveis em LP’s

Facilidade de aprendizado:

- Para aqueles que já programam, aprender PHP se torna mais fácil, pois a sintaxe é bastante semelhante à C e Java, linguagens já conhecidas;
- Por não ser fortemente tipada, usuário pode ter problemas em identificar variáveis e seus tipos;
- Contexto web: muitos recursos;

# Propriedades desejáveis em LP’s

Confiabilidade:

Prós: Coletor de lixo, exceções;
Contras: Checagem de tipos; permite manipulação de memória;

```
<?php
   $var1 = 100;
   $var2 = &$var1;
   $var2 = “PHP”;
   echo $var1."<br>".$var2;
 ?>
```

Valor de $var1 = PHP;

# Propriedades desejáveis em LP’s

Portabilidade:

- No contexto da Web, PHP é uma linguagem portável entre diferentes servidores HTTP, entre diferentes sistemas operacionais, e entre diferentes arquiteturas de hardware;

- Entretanto, não são todos os recursos existentes na linguagem que são portáveis;

# Propriedades desejáveis em LP’s

Reusabilidade:

- Reaproveitamento de rotinas - acesso ao banco de dados, funções para envio de email, acesso a APIs de outros serviços;
- Programação OO: reuso de classes, funções;
- Uso de frameworks;

# Propriedades desejáveis em LP’s

Eficiência:

- Linguagem ideal para implementação de soluções para web;
- Velocidade e a robustez;
- Estrutura e orientação a objetos;
- Portabilidade, garantida pela independência de plataforma;
- Tipagem fraca e a sintaxe similar a C/C++;

# Coletor de Lixo

- Contagem de referências e *Copy-on-write;*
- Atua em diversos momentos. Quando especificado pelo usuário, em fim de função ou no final de um script;
- Variáveis fora de escopo são liberadas. No fim de um script, naturalmente todos os recursos do mesmo são liberados;
- O usuário pode utilizar funções de liberação de memória como unset() e mysql_free_result(). No final de uma função, o interpretador limpa os recursos implicitamente;

# Polimorfismo

Ad-hoc:

- Coerção (Conversão implícita);
- Sobecarga;

Universal:

- Paramétrico;
- Inclusão;

# Ad-hoc - Sobrecarga

- Não suporta sobrecarga de funções, porém suporta sobrescrita (subclasse e superclasse);
- Suporta sobrecarga de operadores:
- Ex: Operador de soma, quando utilizado em arrays, funcionam como união dos elementos;
- O operador + acrescenta os elementos do array da direita no array da esquerda;

# ad-hoc - Sobrecarga

```
<?php
$a = array("1" => "melancia", "2" => "banana");
$b = array("3" =>"abacaxi", "4" => "framboesa",
"5" => "morango");

$c = $a + $b; // Uniao de $a e $b

foreach($c as $elem)
   echo $elem."<br>";

echo '<br>';

$c = $b + $a; // União de $b e $a
foreach($c as $elem)
   echo $elem."<br>";
?>
```

# Polimorfismo – Universal

Inclusão:

- Por suportar paradigma orientado a objeto, permite recursos de herança;

Ex: Classe Pessoa definida;

# Polimorfismo – Herança

```
Class Professor extends Pessoa {
   protected $departamento;

   public function __construct($nome,
$idade, $depto) {
      parent::__construct($nome,$idade);
      departamento = depto;
   }

   public function __destruction(){}
```

# Polimorfismo – Herança

```
   public function imprime() {
      parent::imprime();

      echo “Departamento: ”.$this->
departamento.”<br>”;
   }
} //Fim classe Professor

$pf = new Professor(‘Jose’, 32, ‘MAT’);
$pf->departamento = ‘DI’; // altera
$pf->imprime();
```

# Exceções

- O PHP 5 possui um modelo de exceções similar ao de outras linguagens de programação. Uma exceção pode ser lançada (*throw)* e capturada (catch);

- Código envolvido por bloco **try** para facilitar a captura de exceções potenciais. Cada bloco **try** precisa ter ao menos um bloco **catch** ou **finally** correspondente;

- O objeto lançado precisa ser uma instância da classe **Exception** ou uma subclasse de Exception;

# Exceções

```
<?php
function inverte($x) {
    if ($x == 0) {
        throw new Exception('Divisão por 0.');
    }
    return 1/$x;
}

try {
    echo inverte(5) . "\n";
} catch (Exception $e) {
    echo 'Exceção capturada: ',
   $e->getMessage(), "\n";
} finally {
    echo "Primeiro finaly.\n";
}
```

# Exceções

```
try {
    echo inverte(0) . "\n";
} catch (Exception $e) {
    echo 'Exceção capturada: ',  $e-
>getMessage(), "\n";
} finally {
    echo "Segundo finally.\n";
}

// Execução continua
echo "Olá mundo\n";
?>
```

# Exceções

Saída:

0.2
Primeiro finally.
Exceção capturada: Divisão por zero.
Segundo finally.
Olá mundo

# Concorrência

PHP não é uma linguagem multithread, mas é possível simular paralelismo utilizando multiprocesso;

Quando um processo pai cria um processo filho ambos os processos são executados concorrentemente, possível somente em sistemas Linux;

É arriscado utilizar funções de multiprocessos em um servidor web, pois resultados inesperados podem ocorrer;

# Avaliação da Linguagem

# Alguns Critérios Gerais

- Aplicabilidade;
- Confiabilidade;
- Facilidade de aprendizado;
- Eficiência;
- Portabilidade;
- Suporte ao método de projeto;
- Evolulbilidade;
- Reusabilidade;
- Integração com outros softwares;
- Custo;

# Alguns Critérios Específicos

- Escopo;
- Expressões e comandos;
- Tipos primitivos e compostos;
- Gerenciamento de memória;
- Persistência de dados;
- Passagem de parâmetros;
- Encapsulamento e proteção;
- Sistema de tipos;
- Verificação de tipos;
- Polimorfismo;
- Exceções;
- Concorrência;

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| Eficiência | Sim | Parcial | Sim |
| Portabilidade | Não | Sim | Sim |
| Método projeto | Estruturado | OO | Estruturado e OO |
| Evolutibilidade | Não | Sim | Parcial |
| Reusabilidade | Parcial | Sim | Sim |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| **Aplicabilidade** | **Sim** | **Parcial** | **Parcial** |
| Confiabilidade | | | |
| Aprendizado | | | |
| Eficiência | | | |
| Portabilidade | | | |
| Método projeto | | | OO |
| Evolutibilidade | Não | Sim | Parcial |
| Reusabilidade | Parcial | Sim | Sim |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

C é uma linguagem de propósito geral. Java, contudo, não oferece recursos para controlar diretamente o hardware, obrigando o programador a usar métodos nativos. PHP é de proposito geral, porém, é oficialmente voltado para aplicações web.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| **Confiabilidade** | **Não** | **Sim** | **Parcial** |
| Aprendizado | | | |
| Eficiência | | | |
| Portabilidade | | | |
| Método projeto | | | |
| Evolutibilidade | | | |
| Reusabilidade | Parcial | Sim | Sim |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

C possui inúmeras características estimuladoras de erros em programação (desvio incondicional irrestrito, aritmética de ponteiros etc). Java centraliza certas operações para evitar problemas (verificação de índices de vetor, coleta de lixo etc). PHP possui coletor de lixo. Possui algumas incompatibilidades entre versões; inferência de tipos

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| **Aprendizado** | **Não** | **Não** | **Não** |
| Eficiência | | | |
| Portabilidade | | | |
| Método projeto | | | |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

Nenhuma das LPs atende ao critério. C exige uso massivo de ponteiros, que não é um conceito trivial. Java apresenta muitos conceitos, nem sempre ortogonais. PHP apresenta muitos recursos para web, banco de dados, paradigmas estruturado e orientado a objeto.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| **Eficiência** | **Sim** | **Parcial** | **Sim** |
| Portabilidade | | | |
| Método projeto | | | |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| Interação | | | |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

O critério toma como base programadores muito experientes. Neste caso, C permite um controle mais fino e é portanto mais eficiente. Java assume o controle de diversos aspectos, adicionando mecanismos de verificação, coleta de lixo etc, Diminuindo a eficiência. PHP permite o programador decidir se Coletor de Lixo fica ativado ou não. É considerada a linguagem mais eficiente para implementação de soluções web.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| Eficiência | Sim | Parcial | Sim |
| **Portabilidade** | **Não** | **Sim** | **Sim** |
| Método projeto | | | |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| Interação | | | |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

Embora C seja padronizadas pela ANSI, é comum compiladores diferentes terem características diferentes. Java e PHP, por outro lado, tem a portabilidade como uma característica fundamental da LP.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| Eficiência | Sim | Parcial | Sim |
| Portabilidade | Não | Sim | Sim |
| **Método projeto** | **Estruturado** | **OO** | **Estruturado e OO** |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| Interação | | | |
| Custo | Ferramenta | Ferramenta | Ferramenta |

Esse critério depende da escolha do paradigma adotado no projeto. Contudo, cada vez mais o paradigma OO vem sendo adotado em projetos de sistemas de informação.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | | | |
| Confiabilidade | | | |
| Aprendizado | | | |
| Eficiência | | | |
| Portabilidade | | | |
| Método projeto | | | |
| **Evolutibilidade** | **Não** | **Sim** | **Parcial** |
| Reusabilidade | Parcial | Sim | Sim |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

C possui características que permitem código ilegível e difícil de manter. Java só admite programação OO e ainda oferece estímulo para construção de código bem documentado (ex.: JavaDoc). PHP também possui características que atrapalham legibilidade, porém melhora esse aspecto utilizando a orientação a objetos (estímulo ao encapsulamento e abstração).

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | | | |
| Eficiência | | | |
| Portabilidade | | | |
| Método projeto | | | |
| Evolutibilidade | | | |
| **Reusabilidade** | **Parcial** | **Sim** | **Sim** |
| Interação | Sim | Parcial | Sim |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

C oferece apenas reuso de funções, tipos e variáveis distribuídas em bibliotecas. Java oferece o conceito de classes e possuem mecanismo de pacotes. PHP possui conceito de classes. O polimorfismo universal também auxilia na criação de código reusável e frameworks.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| Eficiência | Sim | Parcial | Sim |
| Portabilidade | Não | Sim | Sim |
| Método projeto | | | |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| **Interação** | **Sim** | **Parcial** | **Sim** |
| Custo | Depende da Ferramenta | Depende da Ferramenta | Depende da Ferramenta |

C pode invocar código compilado por qualquer LP. Java tem que recorrer ao mecanismo JNI (Java Native Interface), que integra com C/C++ apenas. PHP interage com banco de dados.

# Comparando C, Java e PHP

| Critérios Gerais | C | Java | PHP |
|---|---|---|---|
| Aplicabilidade | Sim | Parcial | Parcial |
| Confiabilidade | Não | Sim | Parcial |
| Aprendizado | Não | Não | Não |
| Eficiência | Sim | Parcial | Sim |
| Portabilidade | Não | Sim | Sim |
| Método projeto | Estruturado | OO | Estruturado e |
| Evolutibilidade | | | |
| Reusabilidade | | | |
| Interação | | | |
| **Custo** | **Depende da Ferramenta** | **Depende da Ferramenta** | **Depende da Ferramenta** |

C é de domínio público. Java pertence à Oracle, porém é liberada gratuitamente. Existem inúmeras ferramentas
gratuitas e pagas. PHP, assim como C, é de domínio público.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| Expressões e Comandos | Sim | Sim | Sim |
| Tipos Primitivos e Compostos | Sim | Sim | Sim |
| Gerenciamento de Memória | Programador | Sistema | Programador/ Sistema |
| Persistência de Dados | Biblioteca de Funções | JDBC,Bibliotec a de classes, serialização | Biblioteca de classes, serialização |
| Passagem de Parâmetros | Lista variável e por valor | Lista variável, valor e cópia referência | Lista variável, valor |

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| **Escopo** | **Sim** | **Sim** | **Sim** |
| Expressões e Comandos | | | |
| Tipos Primitivos e Compostos | | | |
| Gerenciamento de Memória | Programador | Sistema | Programador/ Sistema |
| Persistência de Dados | Biblioteca de Funções | JDBC,Bibliotec a de classes, serialização | Biblioteca de classes, serialização |
| Passagem de Parâmetros | Lista variável e por valor | Lista variável, valor e cópia referência | Lista variável, valor |

As três LPs requerem a definição explícita de entidades, associando-as a um escopo de visibilidade.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| **Expressões e Comandos** | **Sim** | **Sim** | **Sim** |
| Tipos Primitivos e Compostos | | | |
| Gerenciamento de Memória | Programador | Sistema | Programador/ Sistema |
| Persistência de Dados | Biblioteca de Funções | JDBC,Biblioteca de classes, serialização | Biblioteca de classes, serialização |
| Passagem de Parâmetros | Lista variável e por valor | Lista variável, valor e cópia referência | Lista variável, valor |

Todas oferecem uma ampla variedade de expressões e comandos.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| Expressões e Comandos | Sim | Sim | Sim |
| **Tipos Primitivos e Compostos** | **Sim** | **Sim** | **Sim** |
| Gerenciamento de Memória | | | |
| Persistência de Dados | | | |
| Passagem de Parâmetros | Lista variável e por valor | Lista variável, valor e cópia referência | Lista variável, valor |

Todas oferecem ampla variedade de tipos primitivos (mas C não oferece booleano) e compostos (mas nenhuma oferece conjunto potência). PHP não oferece tipo enumerado.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| Expressões e Comandos | Sim | Sim | Sim |
| Tipos Primitivos e Compostos | Sim | Sim | Sim |
| **Gerenciamento de Memória** | **Programador** | **Sistema** | **Programador / Sistema** |
| Persistência de Dados | | | |
| Passagem de Parâmetros | por valor | valor e cópia referência | valor |

C deixa a cargo do programador. Java utiliza Coletor de Lixo. PHP o programador pode ou não usar/habilitar o Coletor de Lixo.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| Expressões e Comand | | | |
| Tipos Primitivos e Compostos | | | |
| Gerenciamento de Memória | | | |
| **Persistência de Dados** | **Biblioteca de Funções** | **JDBC,Bibliote ca de classes, serialização** | **Biblioteca de classes, serialização** |
| Passagem de Parâmetros | Lista variável e por valor | Lista variável, valor e cópia referência | Lista variável, valor |

C oferece funções de I/O, mas deixa persistência a cargo do programador. Não existe padrão para interface com BD. Java possui serialização e padronizou interface com BD no JDBC, além de ter operações de I/O. PHP possui serialização e interface com bancos MySQL e outros.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Escopo | Sim | Sim | Sim |
| Expressões e Comandos | Sim | Sim | Sim |
| Tipos Primitivos e Compostos | Sim | Sim | Sim |
| Gerenciamento de Memória | | | |
| Persistência de Dados | | | |
| **Passagem de Parâmetros** | **Lista variável e por valor** | **Lista variável, valor e cópia referência** | **Lista variável, valor** |

C usa apenas passagem por valor, obrigando o uso de ponteiros em diversas ocasiões. Java usa passagem por valor para tipos primitivos, cópia de referência para tipos não primitivos. PHP usa passagem por valor e por referência.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| Sistema de Tipos | Não | Sim | Não |
| Verificação de Tipos | Estática | Estática/ Dinâmica | Dinâmica |
| Polimorfismo | Coersão e Sobrecarga | Todos | Todos |
| Exceções | Não | Sim | Sim |
| Concorrência | Não(biblioteca de funções) | Sim | Parcial |

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| **Encapsulamento e Proteção** | **Parcial** | **Sim** | **Sim** |
| Sistema de Tipos | | | |
| Verificação de Tipos | | | |
| Polimorfismo | | | |
| Exceções | Não | Sim | Sim |
| Concorrência | Não(biblioteca de funções) | Sim | Parcial |

C oferece apenas encapsulamento de dados. Versões recentes permitem ocultamento com declaração (.h) e definição (.c). Java oferecem mecanismo de classes e pacotes. PHP oferece mecanismo de classes e visibilidade – public, protected e private.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| **Sistema de Tipos** | **Não** | **Sim** | **Não** |
| Verificação de Tipos | | | |
| Polimorfismo | | | |
| Exceções | | | |
| Concorrência | | | |

Em C, vários mecanismos(ex.: coerções e aritmética de ponteiros) permitem violação do sistema de tipos. Java possui um sistema de tipos bastante rigoroso. PHP usa inferência de tipos e violações só são detectadas em tempo de execução.

de funções)

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| Sistema de Tipos | Não | Sim | Nâo |
| **Verificação de TiIpos** | **Estática** | **Estática/ Dinâmica** | **Dinâmica** |
| Polimorfismo | | | |
| Exceções | | | |
| Concorrência | | | |

Todas as verificações de C são estáticas. Java faz algumas verificações dinâmicas (ex.: amarração tardia, verificação de índice de vetor). O interpretador do PHP faz verificação de tipos por meio de inferência, somente quando script é executado.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| Ssitema de Tipos | | | |
| Verificação de TIpos | | | |
| **Polimorfismo** | **Coersão e Sobrecarga** | **Todos** | **Todos** |
| Exceções | Não | Sim | Sim |
| Concorrência | Não(biblioteca de funções) | Sim | Parcial |

C não possui polimorfismo paramétrico ou de inclusão. Java possuem todos, porém Java não permite sobrescrita de operadores. PHP possui todos, porém não permite sobrecarga de funções.

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| Sistema de Tipos | | | |
| Verificação de TIpos | | | |
| Polimorfismo | | | |
| **Exceções** | **Não** | **Sim** | **Sim** |
| Concorrência | Não(biblioteca de funções) | Sim | Parcial |

C não oferece. Java oferece um sistema bastante rigoroso de tratamento de exceções. PHP oferece sistema de tratamento de exceções (Exception, blocos try, catch).

# Comparando C, Java e PHP

| Critérios Específicos | C | Java | PHP |
|---|---|---|---|
| Encapsulamento e Proteção | Parcial | Sim | Sim |
| Ssitema de Tipos | Não | Sim | Não |
| Verificação de TIpos | | | |
| Polimorfismo | | | |
| Exceções | | | |
| **Concorrência** | **Não(biblioteca de funções)** | **Sim** | **Parcial** |

Java oferece recursos nativos para exclusão mútua (*synchronized*) e oferece *threads* em sua API básica. PHP suporta operações com threads, porém não possui recursos para multithread (implementado pela biblioteca PCNTL).

# Referências

http://php.net/manual/pt_BR/

F. M. Varejão. Linguagens de Programação – Conceitos e Técnicas. Campus, 2004;

http://programabrasil.org

Tatroe, P. MacIntyre, R. Lerdorf. Programming PHP. O'Reilly Media, 2013;