



Desenvolvimento Web II

Linguagem PHP

Conceitos Iniciais / Tipos de Dados

https://secure.php.net/manual/pt_BR/

Gil Eduardo de Andrade





Conceitos Iniciais

Introdução

- PHP é uma linguagem de script embutida no HTML (linguagens de script são linguagens que podem ser embutidas em outros programas ou em outras linguagens);
- A sintaxe do PHP é baseada, em grande parte, nas linguagens C, Java e Perl – incluindo algumas características específicas do próprio PHP;





Conceitos Iniciais

Objetivos Gerais

- O objetivo da linguagem PHP é possibilitar que os desenvolvedores web codifiquem páginas dinâmicas e de forma rápida;
- A sigla PHP significa: Hypertext Preprocessor ou Pré-processador de hipertexto;





Conceitos Iniciais

Funcionamento

- A infra-estrutura da Internet é baseada no modelo cliente x servidor, onde clientes estão conectados a servidores que possuem e fornecem cópias de documentos;
- O PHP é uma tecnologia “*server-side*” (do lado do servidor), onde um servidor faz sua interpretação (do seu código) e retorna como resultado dados que serão exibidos pelo navegador (*browser*);





Conceitos Iniciais

Funcionamento

- Portanto, todos os processos, rotinas e funções são processadas no lado servidor, ou seja, o usuário recebe apenas o resultado desse processamento no seu navegador;
- Para que o PHP funcione corretamente é necessário o servidor Apache, responsável por interpretar o código PHP e dar suporte a Sistemas Gerenciadores de Banco de dados (ex: MySQL, Oracle);





Conceitos Iniciais

Configuração /Instalação

- Para configurar um computador como servidor com suporte a linguagem PHP (e também ao SGBD MySQL) é preciso instalar o pacote AMP, composto por Apache, MySQL e PHP.
- No Windows, esse pacote é chamado de WAMP, no Linux é denominado LAMP e no Mac é conhecido como MAMP;



Conceitos Iniciais

Estrutura Básica (codificação)

A sequência de caracteres “<?php” indica que um novo trecho de código em linguagem PHP está sendo iniciado. Em outras palavras, tal sequência é utilizada para especificar onde inicia-se o conjunto de instruções que deve ser interpretado pelo servidor Apache.

```
1 <?php
2 echo "Hello";
3 print "World!";
4 ?>
```

Trecho de código em linguagem PHP que será interpretado. As rotinas “echo” e “print” são utilizadas para apresentar dados no navegador;

A sequência de caracteres “?>” é utilizada para especificar onde terminar o trecho de código em linguagem PHP.



Conceitos Iniciais

Estrutura Básica (interpretação / execução)



HelloWorld!

Resultado da execução.





Tipos de Dados

Variáveis / Sintaxe

- Na linguagem PHP, assim como outras linguagens interpretadas (ex.: Python,), não há a necessidade de declarar as variáveis (como acontece nas linguagens C e Java);
- Sendo assim, a linguagem PHP é denominada fracamente tipada, já que ao não declararmos uma variável não definimos o seu tipo;





Tipos de Dados

Variáveis / Sintaxe

- Por essa característica (não declaração das variáveis), a linguagem PHP faz uso da chamada “*tipagem dinâmica*”;
- Na *tipagem dinâmica* a escolha do tipo da variável ocorre de forma dinâmica no momento que o código está sendo interpretado/executado;



Tipos de Dados

Variáveis / Sintaxe (codificação)

```
1 <?php
2
3 $curso = "Graduação em Análise e Desenvolvimento de Sistemas";
4 $disciplina = "Desenvolvimento Web II";
5 $turma = 2015;
6
7 echo "[CURSO]: $curso<br>";
8 echo "[DISCIPLINA]: $disciplina<br>";
9 echo "[TURMA]: $turma";
10 ?>
```

O nome das variáveis sempre iniciam pela caractere reservado \$.

Variáveis não precisam ser declaradas, sendo criadas no momento que são utilizadas, elas podem receber valores de qualquer tipo (no exemplo, tipos "String" e "inteiro").

Obs.: repare que o comando "echo" permite apresentar textos estáticos juntamente com o conteúdo de variáveis e "tags" HTML.



Tipos de Dados

Variáveis / Sintaxe (interpretação/execução)



[CURSO]: Graduação em Análise e Desenvolvimento de Sistemas
[DISCIPLINA]: Desenvolvimento Web II
[TURMA]: 2015

Resultado da execução.





Tipos de Dados

Variáveis / Sintaxe (codificação)

```
1 <?php
2
3     $var = "Gil Eduardo";    // String
4     $tipo = gettype($var);
5     echo "$var ($tipo)<br>";
6     $var = 12;              // Inteiro
7     $tipo = gettype($var);
8     echo "$var ($tipo)<br>";
9     $var = 3.1415;          // Float/Double
10    $tipo = gettype($var);
11    echo "$var ($tipo)<br>";
12    $var = true;             // Booleano
13    $tipo = gettype($var);
14    echo "$var ($tipo)<br>";
15 ?>
```

Observe que uma variável (*\$var*) pode receber dados de vários tipos (*string*, *inteiro*, *double*). Sendo assim, para reconhecermos o tipo de uma variável PHP podemos utilizar a função *gettype()*.





Tipos de Dados

Variáveis / Sintaxe (interpretação/execução)



Gil Eduardo (string)

12 (integer)

3.1415 (double)

1 (boolean)

Resultado da execução.



Funções – Tipos de Dados

Funções / Sintaxe (codificação)

```
1 <?php
2
3 if(is_null($var)) {
4     echo "Variável \ $var é nula/vazia!<br>";
5 }
6 $var = "Gil Eduardo";
7 if(is_string($var)) {
8     echo "Variável \ $var é uma string!<br>";
9 }
10 $var = 12;
11 if(is_integer($var)) {
12     echo "Variável \ $var é um inteiro!<br>";
13 }
14 $var = false;
15 if(is_bool($var)) {
16     echo "Variável \ $var é booleana!";
17 }
18 // is_float(), is_array(), is_object()
19 ?>
```

Observe que para mostra o caractere '\$' (ou outro reservado) utilizamos '\ ' antes dele.

Assim com a função **gettype()** permite obter o tipo de uma variável (\$var), existem funções que permite testar se a variável é de um tipo específico ou seu conteúdo é nulo: **is_null()**, **is_string()**, **is_integer()**, etc.



Funções – Tipos de Dados

Funções / Sintaxe (interpretação/execução)



Variável \$var é nula/vazia!
Variável \$var é uma string!
Variável \$var é um inteiro!
Variável \$var é booleana!

Resultado da execução.





Tipos de Dados - Arrays

Descrição

- Os arrays, em PHP, são mapas ordenados de chaves e valores, ou seja, é possível atribuir a um elemento do array uma chave e um valor;
- Tal característica permite que os arrays, em PHP, representem listas, tabelas *hash*, pilhas, filas, coleções, etc.;





Tipos de Dados - Arrays

Descrição

- Os arrays, em PHP, também permitem que o elemento de um array '*x*' seja um outro array '*y*';
- Essa característica possibilita que estruturas (arrays) multidimensionais sejam facilmente criadas – uma árvore, por exemplo, pode ser facilmente criada em PHP pelo uso de arrays;





Tipos de Dados - Arrays

Definição de Arrays

- Existem duas forma de definir um array em PHP:
 - 1) de maneira explícita através do construtor *array()*.
 - *Ex.: Array([chave] => valor, ...)* ;
 - 2) de maneira implícita ou direta.
 - *Ex.: \$array_exemplo[chave] = valor;*



Tipos de Dados - Arrays

Definição Explícita de Arrays (codificação)

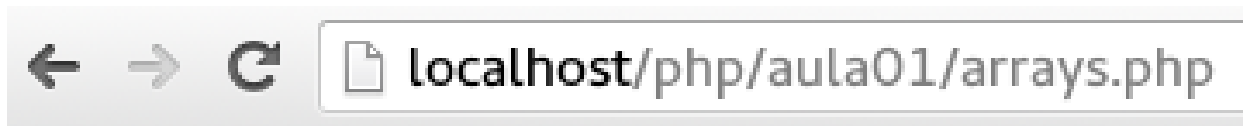
```
1 <?php
2 // Array: Definição Explícita (sem chave)
3 $var = array(1, 2, 3, 4);
4
5 echo "[for]: ";
6 for($a=0; $a<count($var); $a++) {
7     echo "$var[$a] ";
8 }
9
10 echo "<br>[foreach]: ";
11 foreach ($var as $dado) {
12     echo "$dado ";
13 }
14 ?>
```

Quando utilizamos o construtor **array()** durante a definição de um novo **array** dizemos que essa definição é explícita. Perceba que não são utilizadas chaves nesta construção, mas apenas valores, quando isso acontece são atribuídos índices numéricos crescentes, partindo do '0', para os dados inseridos no **array**.



Tipos de Dados - Arrays

Definição Explícita de Arrays (interpretação/execução)



[for]: 1 2 3 4

[foreach]: 1 2 3 4

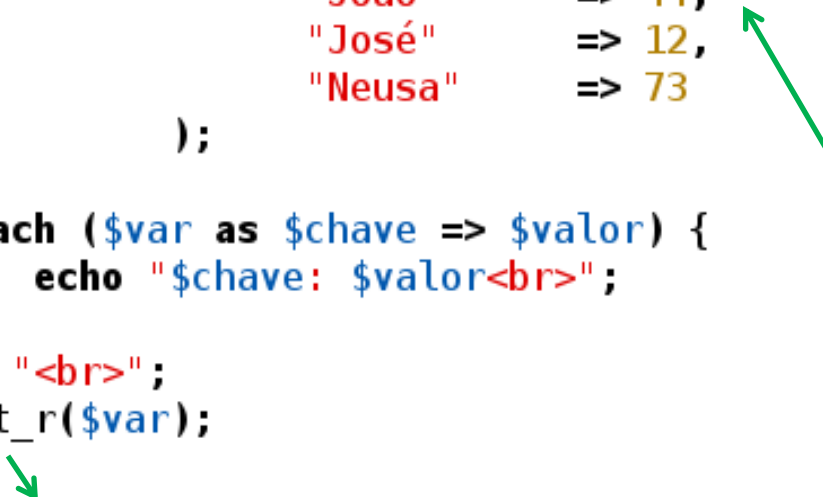
Resultado da execução.



Tipos de Dados - Arrays

Definição Explícita de Arrays (codificação)

```
1 <?php
2 // Array: Definição Explícita (com chave)
3 $var = array(      "Maria"    => 25,
4                   "João"     => 44,
5                   "José"     => 12,
6                   "Neusa"    => 73
7                   );
8
9 foreach ($var as $chave => $valor) {
10     echo "$chave: $valor<br>";
11 }
12 echo "<br>";
13 print_r($var);
14 ?>
```



Neste exemplo de definição explícita foram utilizadas chaves durante a construção do *array*. Sendo assim, os valores inteiros (25, 44, 12, 73) estão vinculados as suas respectivas chaves (Maria, João, José, Neusa).

Observe que função “*print_r*” permite apresentar, de maneira compreensível, todo o array de uma só vez.



Tipos de Dados - Arrays

Definição Explícita de Arrays (interpretação/execução)



Maria: 25

João: 44

José: 12

Neusa: 73

Array ([Maria] => 25 [João] => 44 [José] => 12 [Neusa] => 73)

Resultado da execução.





Tipos de Dados - Arrays

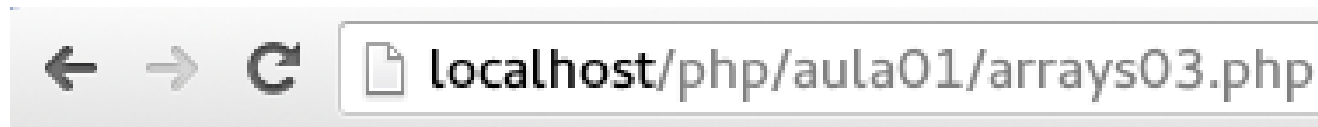
Definição Explícita de Arrays (codificação)

```
1 <?php
2     // Array: Definição Explícita (com chave)
3     $var = array(         "000.000.000-00" => "Maria",
4                           "000.000.000-01" => "João"
5     );
6
7     foreach ($var as $chave => $valor) {
8         echo "($chave: $valor) ";
9     }
10
11    $var = array(         12      => "Maria",
12                        21      => "João"
13    );
14
15    echo "<br>";
16    foreach ($var as $chave => $valor) {
17        echo "($chave: $valor) ";
18    }
19 ?>
```




Tipos de Dados - Arrays

Definição Explícita de Arrays (interpretação/execução)



```
(000.000.000-00: Maria) (000.000.000-01: João)  
(12: Maria) (21: João)
```

Resultado da execução.



Tipos de Dados - Arrays

Definição Direta de Arrays (codificação)

```
1 <?php
2 // Array: Definição Direta (sem chave)
3 $var[0] = "Desenvolvimento";
4 $var[1] = "Web";
5 $var[2] = "II";
6
7 echo "[for]: ";
8 for($a=0; $a<count($var); $a++) {
9     echo "$var[$a] ";
10 }
11
12 echo "<br>[foreach]: ";
13 foreach ($var as $dado) {
14     echo "$dado ";
15 }
16 ?>
```

Quando não utilizamos o construtor **array()**, durante a criação de um novo **array**, dizemos que essa definição é direta. Nela a sintaxe torna-se muito próxima da utilizada quando codificamos uma aplicação na linguagem C.



Tipos de Dados - Arrays

Definição Direta de Arrays (interpretação/execução)



[for]: Desenvolvimento Web II

[foreach]: Desenvolvimento Web II

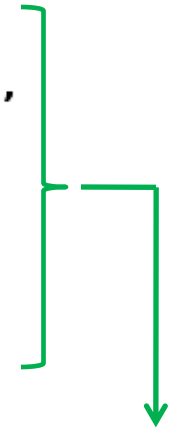
Resultado da execução.



Tipos de Dados - Arrays

Arrays Multidimensionais (codificação)

```
1 <?php
2 // Array Multidimensional: Definição Explícita
3 $arr = array(      "Maria" => array(
4                     "endereco" => "Rua Chile 1046",
5                     "bairro"   => "Rebouças",
6                     ),
7
8                     "João" => array(
9                     "endereco" => "Rua Iapó 234",
10                    "bairro"   => "Prado Velho",
11                    )
12                );
13
14 foreach($arr as $chave => $aux) {
15
16     echo strtoupper($chave).": <br>";
17     foreach($aux as $chave => $valor) {
18         echo "    - $valor<br>";
19     }
20     echo "<br>";
21 }
22 ?>
```



Os valores vinculados as chaves (Maria, João) são outros arrays contendo novas chaves (endereco, bairro) e novos valores vinculados a elas.



Tipos de Dados - Arrays

Definição Direta de Arrays (interpretação/execução)



MARIA:

- Rua Chile 1046
- Rebouças

JOÃO:

- Rua Iapó 234
- Prado Velho

Resultado da execução.





Tipos de Dados - Arrays

Funções Arrays (https://secure.php.net/manual/pt_BR/ref.array.php)

Os códigos-fonte que exemplificam essas funções estão disponíveis em: “Mais exemplos sobre o conteúdo”.

- ***array array_key(arr)***: retorna todas as chaves do array ***arr***;
- ***array array_values(arr)***: retorna todos os valores do array ***arr***;
- ***String array_search(val, arr)***: busca pelo valor ***val*** no array ***arr*** e retorna a respectiva chave;





Tipos de Dados - Arrays

Funções Arrays

- ***bool array_key_exists(key, arr):*** verifica se uma chave ou índice *key* existe para um array *arr*;
- ***bool in_array(val, arr):*** verifica se um valor *val* existe em um array *arr*;
- ***bool isset(var):*** verifica se a variável *var* foi inicializada;





Tipos de Dados - Arrays

Funções Arrays

- *void unset(var)*: destrói a variável *var*;
- *bool empty(var)*: verifica se *var* está vazia;
- *int array_push(arr, ele[])*: adiciona um ou mais elementos *ele[]* no final do array *arr*;
- *String array_pop(arr)*: extrai um elemento do final do array *arr*;





Tipos de Dados - Arrays

Funções Arrays



Indica que o retorno pode ser um determinado conjunto de tipos, porém não todos .

- *mixed* ***array_shift(arr)***: remove o primeiro elemento do array ***arr***;
- *String* ***array_unshift(arr)***: adiciona um ou mais elementos no início do array ***arr***;
- *int* ***count(var)***: Conta o número de elementos da variável ***var***, ou propriedades do objeto ***var***;





Tipos de Dados - Arrays

Funções Arrays

- ***array explode(del, str)***: retorna uma matriz de strings, dividindo *str* de acordo com *del*;
- ***String implode(str, arr)***: retorna uma string contendo os elementos do array *arr* concatenados pela string *str*;
- ***array array_combine(key, val)***: Cria um array usando o array *key* para chaves e o array *val* para os valores;





Tipos de Dados - Arrays

Funções Arrays

- *array array_diff(arr1, arr2):* encontra a diferença entre os arrays *arr1* e *arr2* (elementos que existem em *arr1* e não existem em *arr2*);
- *array array_intersect (arr1, arr2):* encontra a intersecção entre os arrays *arr1* e *arr2* (elementos que existem tanto em *arr1* quanto em *arr2*);





Desenvolvimento Web II

CONSTANTES
VARIÁVEIS SUPERGLOBAIS
ESCOPO DE VARIÁVEIS





Tipos de Dados - Constantes

Definição

- As constantes, em PHP, podem ser vistas como identificadores para um determinado valor;
- Após ter sido definida, uma constante não pode ser alterada nem removida;
- Para definir uma constante utilizamos o comando ***define()***;



Tipos de Dados - Constantes

Definição de Constantes (codificação/execução)

Dica: por convenção o nome de uma constante contém somente letras maiúsculas.

```
1 <?php
2
3     define("PI", 3.1415);
4     echo PI;
5 ?>
```



localhost/php/aula01/constantes.php

3.1415

Resultado da execução.



Tipos de Dados - Constantes

Função *define()*

- Na linguagem PHP, o identificador da função ***define()*** (nome da constante) é *case sensitive*, ou seja, diferencia letra maiúsculas de minúsculas;
- Entretanto é possível alterar isso colocando o valor ***true*** no terceiro parâmetro da função;



Tipos de Dados - Constantes

Definição de Constantes (codificação)

```
1 <?php
2
3     define("PI", 3.1415, true);
4     echo PI;
5 ?>
```



3.1415

Resultado da execução.



Tipos de Dados - Constantes

Constantes Pré-definidas

- A linguagem PHP disponibiliza um conjunto de constantes já pré-definidas muito úteis;
- Entre tais constantes destacam-se:
 - **__FILE__**: contém o nome do arquivo (script) que está sendo executado;
 - **__DIR__**: contém o diretório do script que está sendo executado;





Tipos de Dados - Constantes

Constantes Pré-definidas

- `__LINE__`: contém o número da linha onde está a constante;
- `__FUNCTION__`: contém o nome da função que está sendo executada;
- `__CLASS__`: contém o nome da classe;
- `__METHOD__`: contém o nome do método da classe;
- `__NAMESPACE__`: contém o nome do *namespace* atual;





Tipos de Dados - Constantes

Constantes Pré-definidas (codificação)

```
1 <?php
2
3     function funcConsts() {
4         echo "ARQUIVO: " . __FILE__ . "<br>";
5         echo "DIRETÓRIO: " . __DIR__ . "<br>";
6         echo "LINHA: " . __LINE__ . "<br>";
7         echo "FUNÇÃO: " . __FUNCTION__ . "<br>";
8     }
9
10    funcConsts( );
11 ?>
```

Observação: o conceito de funções (functions) em PHP será abordado em detalhes na Aula 03.



Tipos de Dados - Constantes

Constantes Pré-definidas (interpretação/execução)



ARQUIVO: /var/www/html/php/aula01/constantes_pre.php

DIRETÓRIO: /var/www/html/php/aula01

LINHA: 6

FUNÇÃO: funcConsts

Resultado da execução.





Tipos de Dados - Constantes

Constantes Pré-definidas (codificação)

```
1 <?php
2     class veiculo {
3
4         private $marca;
5
6         function __construct() {
7             echo "CLASSE: " . __CLASS__ . "<br>";
8         }
9         function setMarca($marca) {
10             $this->marca = $marca;
11             echo "MÉTODO: " . __METHOD__ . "<br>";
12         }
13     }
14
15     $obj = new veiculo();
16     $obj->setMarca("Wolksvagem");
17 ?>
```

Observação: o conceito sobre classe e objeto em PHP será abordado em detalhes na Aula 04.



Tipos de Dados – Constantes

Constantes Pré-definidas (interpretação/execução)



CLASSE: veiculo

MÉTODO: veiculo::setMarca

Resultado da execução.





Tipos de Dados – Variáveis

Variáveis Pré-definidas

- A linguagem PHP disponibiliza também um conjunto de variáveis já pré-definidas acessíveis por qualquer script;
- Tais variáveis dependem do ambiente e módulo PHP que estão carregados, elas podem ser obtidas pela função ***get_defined_vars();***





Tipos de Dados – Variáveis

Variáveis Pré-definidas (codificação/execução)

```
1 <?php
2     $vars_pre = get_defined_vars();
3     print_r($vars_pre);
4 ?>
```



← → ↻  localhost/php/aula01/variaveis_pre.php

Array ([_GET] => Array () [_POST] => Array () [_COOKIE] => Array () [_FILES] => Array ())

Resultado da execução.





Tipos de Dados – Variáveis

Variáveis *Superglobais*

- A linguagem PHP disponibiliza ainda um conjunto de variáveis *superglobais* já pré-definidas;
- Essas variáveis tem por objetivo facilitar o acesso a dados enviados pelo servidor web (como, por exemplo, campos de um formulário);





Tipos de Dados – Variáveis

Variáveis Superglobais

- As seguintes variáveis superglobais estão disponíveis no PHP:
 - ***\$GLOBALS***: retorna um *array* para todas as variáveis que estão atualmente disponíveis no escopo global;
 - ***\$_SERVER***: *array* contendo informações sobre o servidor web e o ambiente de execução;
 - ***\$_GET***: *array* contendo todas as variáveis enviadas via método GET (mais detalhes na próxima aula);





Tipos de Dados – Variáveis

Variáveis Superglobais

- ***\$_POST***: *array* contendo todas as variáveis enviadas via método POST (mais detalhes na próxima aula);
- ***\$_COOKIE***: *array* contendo todas as variáveis especiais que são gravadas na máquina do usuário e recuperadas pelo navegador (mais detalhes na Aula 03);
- ***\$_FILES***: *array* contendo informações sobre arquivos enviados do computador do cliente para o servidor web – *upload* (mais detalhes nas aulas de importação de dados);





Tipos de Dados – Variáveis

Variáveis Superglobais

- ***\$_ENV***: *array* contendo as variáveis de ambiente disponíveis no momento;
- ***\$_REQUEST***: *array* contendo o todas as variáveis do *\$_GET*, *\$_POST* e *\$_COOKIE* (mais detalhes na Aula 03);
- ***\$_SESSION***: *array* contendo registradas na seção corrente (mais detalhes na Aula 03);



Tipos de Dados – Variáveis

Superglobais - \$GLOBALS (codificação/execução)

```
1 <?php
2     function exemplo_globals() {
3
4         $var = "variável local a função";
5
6         echo "\$var NO ESCOPO LOCAL: $var<br>";
7         echo "\$var NO ESCOPO GLOBAL: " . $GLOBALS['var'] . "<br>";
8     }
9
10    $var = "variável global ao script";
11    exemplo_globals();
12 ?>
```



← → ↻  localhost/php/aula01/superglobal_globals.php

\$var NO ESCOPO LOCAL: variável local a função

\$var NO ESCOPO GLOBAL: variável global ao script



Escopo de Variáveis

Definição

- O escopo é o contexto onde uma variável foi definida, ou seja, no qual é possível acessá-la;
- De maneira geral o escopo resume-se ao contexto do script em execução, onde as variáveis estão disponíveis em qualquer parte dele, inclusive em arquivos carregados pelos mesmo;





Escopo de Variáveis

Definição

- Contudo esse escopo não abrange funções e classes, ou seja, uma variável definida dentro de um script não abrange o contexto de uma função ou classe;
- Para que possa abranger também esses dois contextos é preciso defini-la com o uso da palavra reserva ***global***;



Escopo de Variáveis

Escopo do script apenas (codificação)

```
1 <?php
2     $val = 12; // Escopo do script
3
4     function quadrado() {
5
6         $val = 6; // Escopo da função
7         $val = $val * $val;
8         echo "\$val (na função \" . __FUNCTION__ . "()\") = $val";
9     }
10
11     echo "\$val = $val<br>";
12     quadrado();
13     echo "<br>\$val = $val";
14 ?>
```

A variável **\$val** é definida no escopo do script, não fazendo parte do contexto da função **quadrado()**;

A variável **\$val** (que não é a mesma do escopo do script) é definida no escopo da função **quadrado()**, não fazendo parte do escopo de todo o script;



Escopo de Variáveis

Escopo do script apenas (interpretação/execução)

Resultado da execução.



```
$val = 12  
$val (na função "quadrado()") = 36  
$val = 12
```

A variável **\$val** (definida no escopo do script) não tem seu valor alterado mesmo após a execução da função `quadrado()`. Isso acontece porque a variável **\$val** definida no escopo da função não é a mesma definida no escopo do script.



Escopo de Variáveis

Escopo Global (codificação)

A variável *\$val* é definida no escopo *global*, ou seja, ela vale para o contexto de todo o script. Sendo assim, alterando seu valor dentro da função altera para todo o script;

```
1 <?php
2     $val = 12; // Escopo do script
3
4     function quadrado() {
5
6         global $val; // Escopo global
7         $val = 6;
8         $val = $val * $val;
9         echo "\$val (na função \"\". __FUNCTION__ . \"()\") = $val";
10    }
11
12    function metade() {
13        global $val;
14        $val = $val / 2;
15        echo "\$val (na função \"\". __FUNCTION__ . \"()\") = $val";
16    }
17
```

Escopo de Variáveis

Escopo Global (codificação)

Utilizando a superglobal ***\$GLOBALS*** também é possível acessar a variável ***\$val*** no contexto global do script, ou seja, alterando seu valor via ***\$GLOBALS*** ele é válido para todo o script;

```
18 function soma10() {  
19     $GLOBALS['val'] = $GLOBALS['val'] + 10;  
20     echo "\$val (na função \"\". __FUNCTION__ .\"()\") = \".$GLOBALS['val']\";  
21 }  
22  
23 echo "\$val = $val<br>\";  
24 quadrado();  
25 echo "<br>\$val = $val<br>\";  
26 metade();  
27 echo "<br>\$val = $val<br>\";  
28 soma10();  
29 echo "<br>\$val = $val\";  
30 ?>
```

Após a execução de cada uma das três funções o valor da variável ***\$val*** é mostrado novamente para mostra que seu conteúdo foi modificado pela definição global utilizada nas funções;



Escopo de Variáveis

Escopo Global (interpretação/execução)



```
$val = 12  
$val (na função "quadrado()") = 36  
$val = 36  
$val (na função "metade()") = 18  
$val = 18  
$val (na função "soma10()") = 28  
$val = 28
```

Resultado da execução.





Desenvolvimento Web II

FORMULÁRIOS – PHP
POST e GET





Formulário HTML – PHP

Definição

- Formulários HTML são interfaces criadas (lado cliente) para que os usuários possam inserir informações;
- Essas informações podem ser posteriormente tratadas por algum script no lado do servidor;
- No nosso caso, o script em questão, é o PHP.





Formulário HTML – PHP

Exemplo de formulário HTML

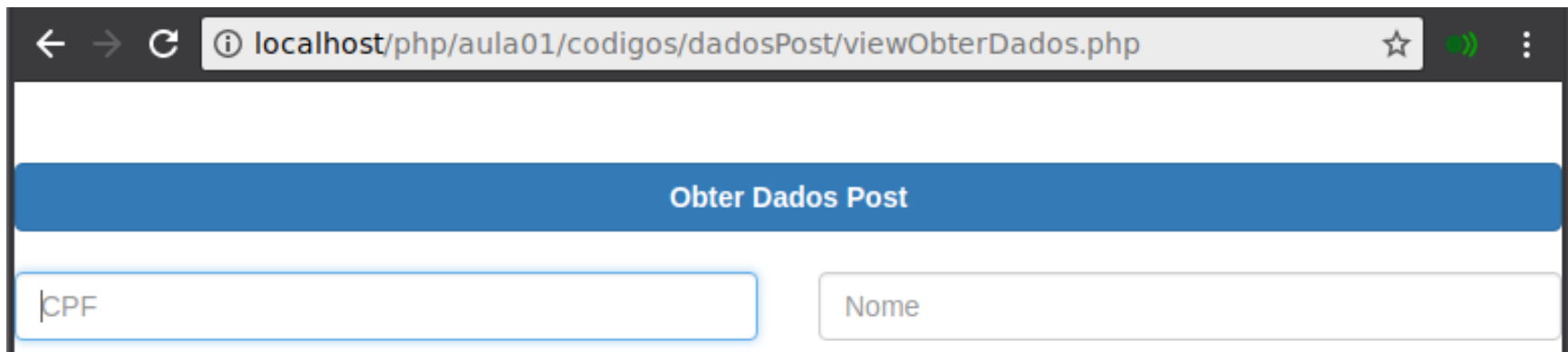
```
<form class="form" method="post" action="viewObterDados.php">
<input TYPE="hidden" NAME="form_submit" VALUE="OK">
  <br><br>
  <button type="submit" class="btn btn-primary btn-block">
    <b>Obter Dados Post</b>
  </button>
  <br>
  <div class='row'>
    <div class='col-sm-6'>
      <label class="sr-only">CPF</label>
      <input type="text" class="form-control" name="cpf">
    </div>
    <div class='col-sm-6'>
      <label class="sr-only">nome</label>
      <input type="text" class="form-control" name="nome">
    </div>
  </div>
</form>
```

O exemplo engloba os arquivos ***viewObterDados.php*** e ***obterDadosPost.php***.



Formulário HTML – PHP

Exemplo de formulário HTML (interpretação/execução)



A screenshot of a web browser window. The address bar shows the URL `localhost/php/aula01/codigos/dadosPost/viewObterDados.php`. The page has a blue header bar with the text "Obter Dados Post". Below the header, there are two input fields: one labeled "CPF" and another labeled "Nome".

Resultado da execução

O código apresentado no slide anterior encontra-se disponível na seção “dicas de aula” – localização: “dadosPost/viewObterDados.php”.

Os detalhes da implementação são explicados durante a aula.



Formulário HTML – PHP

Exemplo de formulário HTML

```
<form class="form" method="post" action="viewObterDadosArray.php">
<input TYPE="hidden" NAME="form_submit" VALUE="OK">
  <br><br>
  <button type="submit" class="btn btn-primary btn-block">
    <b>Obter Dados Post</b>
  </button>
  <br>
  <div class='row'>
    <div class='col-sm-4'>
      <label class="sr-only">CPF</label>
      <input type="text" class="form-control" name="cpf" max
    </div>
    <div class='col-sm-4'>
      <label class="sr-only">nome</label>
      <input type="text" class="form-control" name="nome" ma
    </div>
    <div class='col-sm-4'>
      <label class="sr-only">telefone</label>
      <input type="text" class="form-control" name="telefone
    </div>
  </div>
</form>
```

O exemplo engloba os arquivos ***viewObterDadosArray.php*** e ***obterDadosMontarArray.php***.



Formulário HTML – PHP

Exemplo de formulário HTML (interpretação/execução)



A screenshot of a web browser window. The address bar shows the URL `localhost/php/aula01/codigos/dadosPost/viewObterDadosArray.php`. The page has a blue header bar with the text "Obter Dados Post". Below the header, there are three input fields: "CPF", "Nome", and "Telefone". The "CPF" field is currently active, showing a cursor.

Resultado da execução

O código apresentado no slide anterior encontra-se disponível na seção “dicas de aula” – localização: “dadosPost/viewObterDadosArray.php”.

Os detalhes da implementação são explicados durante a aula.





Formulário HTML – PHP

Exemplo de formulário HTML

```
<form class="form" method="post" action="viewRoute.php">
<input TYPE="hidden" NAME="form_submit" VALUE="OK">
  <br><br>
  <div class='row'>
    <div class='col-sm-4'>
      <button type="submit" name="acao" value="cadastar/0">
        <b>Cadastar</b>
      </button>
    </div>
    <div class='col-sm-4'>
      <button type="submit" name="acao" value="alterar/1">
        <b>Alterar</b>
      </button>
    </div>
    <div class='col-sm-4'>
      <button type="submit" name="acao" value="remover/2">
        <b>Remover</b>
      </button>
    </div>
  </div>
</form>
```

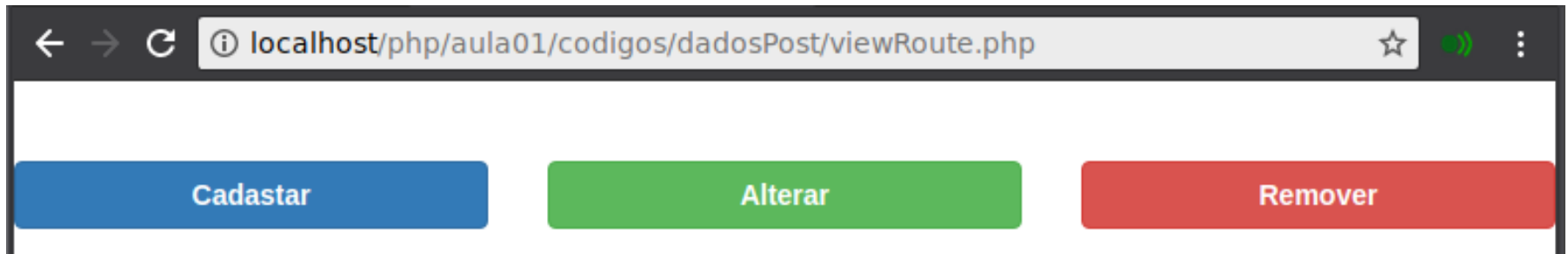
O exemplo engloba os arquivos ***viewRoute.php*** e ***route.php***.





Formulário HTML – PHP

Exemplo de formulário HTML (interpretação/execução)



Resultado da execução

O código apresentado no slide anterior encontra-se disponível na seção “dicas de aula” – localização: “dadosPost/viewRoute.php”.

Os detalhes da implementação são explicados durante a aula.



HTML Dinâmico (Table) – PHP

Exemplo de HTML Dinâmico

HTML

```
<table class='table table-striped'>
  <thead>
    <tr>
      <th>CPF</th>
      <th>NOME</th>
      <th>ENDEREÇO</th>
      <th>TELEFONE</th>
      <th>AÇÕES</th>
    </tr>
  </thead>
  <tbody>
    <?php loadTabela(); ?>
  </tbody>
</table>
```

PHP

```
foreach ($pessoas as $cpf => $dados) {

    if(!empty($dados)) {
        echo "<tr>";
        echo "<td>".$cpf."</td>";

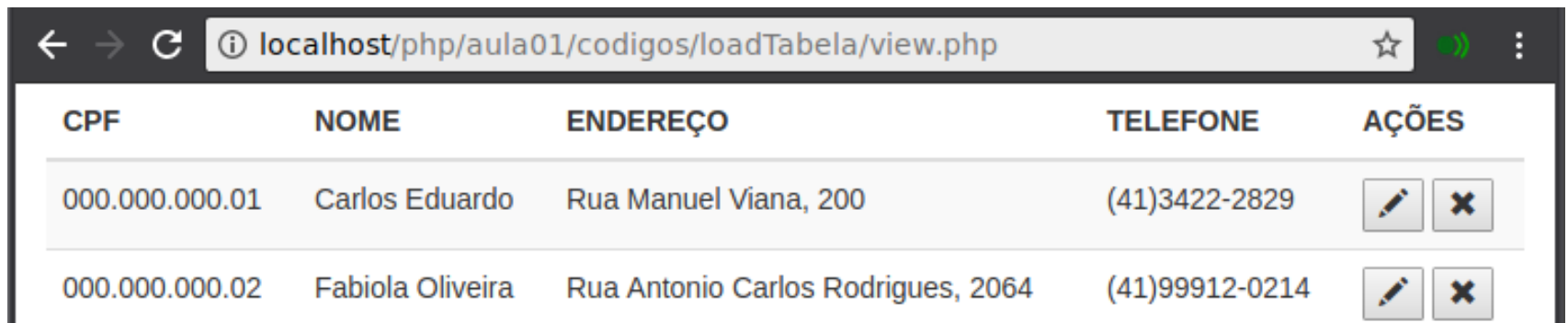
        foreach ($dados as $valor) {
            echo "<td>".$valor."</td>";
        }

        echo "<td>";
        echo "<button type='submit'";
        echo "<span class='glyph";
        echo "</button>";
    }
}
```





O exemplo engloba os arquivos ***view.php*** e ***loadTabelaArray.php***.

Formulário HTML – PHP

Exemplo de HTML Dinâmico (interpretação/execução)



The screenshot shows a web browser window with the address bar displaying 'localhost/php/aula01/codigos/loadTabela/view.php'. The browser interface includes back, forward, and refresh buttons. The main content area displays a table with five columns: CPF, NOME, ENDEREÇO, TELEFONE, and AÇÕES. There are two data rows. Each row in the 'AÇÕES' column contains two icons: a pencil for editing and an 'X' for deleting.

CPF	NOME	ENDEREÇO	TELEFONE	AÇÕES
000.000.000.01	Carlos Eduardo	Rua Manuel Viana, 200	(41)3422-2829	 
000.000.000.02	Fabiola Oliveira	Rua Antonio Carlos Rodrigues, 2064	(41)99912-0214	 

Resultado da execução

Os códigos apresentados no slide anterior encontram-se disponíveis na seção “dicas de aula” – localização: “loadTabela/view.php” e “loadTabela/loadTabelaArray.php” .

Os detalhes da implementação são explicados durante a aula.



Desenvolvimento Web II

MANIPULANDO
ARQUIVO TEXTO





ARQUIVO TEXTO – PHP

Definição

- Um arquivo texto é um recurso computacional estruturado em linhas que permite o armazenamento de dados;
- Ele é utilizado, normalmente, para armazenar um ou mais caracteres, contendo também caracteres especiais como **EOF** (final de arquivo).





ARQUIVO TEXTO – PHP

Exemplo Leitura Arquivo

```
<?php

    include_once('lerArquivo.php');
    ler();

?>
```

```
000.000.000-01
Gil Eduardo de Andrade#Rua das Oliveiras, 2012#(41)3423-5678
000.000.000-02
Denise Camargo Correa#Av. José Lobo, 905#(41)3424-0033
000.000.000-03
Carlos Eduardo Ferla#Rua Xavier da Silva, 134#(41)3425-6713
```

O exemplo engloba os arquivos ***view.php***
e ***lerArquivo.php*** e ***pessoas.txt***.

```
function ler() {

    $dados = array();
    $fp = fopen('pessoas.txt', 'r');

    if ($fp) {

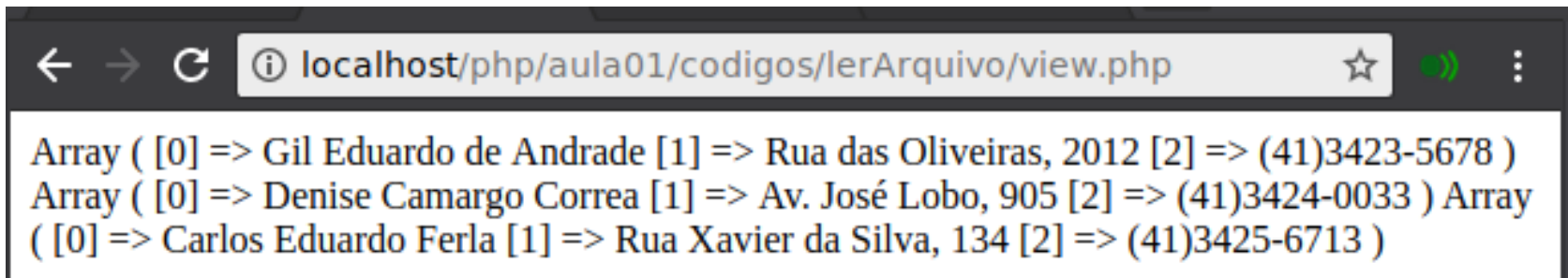
        while(!feof($fp)) {
            $cpf = fgets($fp);
            $linha = fgets($fp);
            if(!empty($linha)) {
                $dados = explode("#", $linha);
                print_r($dados);
            }
        }

        fclose($fp);
    }
}
```



Formulário HTML – PHP

Exemplo Leitura Arquivo (interpretação/execução)



```
Array ( [0] => Gil Eduardo de Andrade [1] => Rua das Oliveiras, 2012 [2] => (41)3423-5678 )  
Array ( [0] => Denise Camargo Correa [1] => Av. José Lobo, 905 [2] => (41)3424-0033 ) Array  
( [0] => Carlos Eduardo Ferla [1] => Rua Xavier da Silva, 134 [2] => (41)3425-6713 )
```

Resultado da execução

Os códigos apresentados no slide anterior encontram-se disponíveis na seção “dicas de aula” – localização: “lerArquivo/view.php” e “lerArquivo/lerArquivo.php” .

Os detalhes da implementação são explicados durante a aula.



ARQUIVO TEXTO – PHP

Exemplo Leitura Arquivo

```
<?php  
    include_once('lerArquivoMontarArray.php');  
    lerMontarArray();  
?>
```

```
000.000.000-01  
Gil Eduardo de Andrade#Rua das Oliveiras, 2012#(41)3423-5678  
000.000.000-02  
Denise Camargo Correa#Av. José Lobo, 905#(41)3424-0033  
000.000.000-03  
Carlos Eduardo Ferla#Rua Xavier da Silva, 134#(41)3425-6713
```

```
function lerMontarArray() {  
  
    $pessoas = array();  
    $fp = fopen('pessoas.txt', 'r');  
  
    if ($fp) {  
  
        while(!feof($fp)) {  
            $arr = array();  
            $cpf = fgets($fp);  
            $dados = fgets($fp);  
            if(!empty($dados)) {  
                $arr = explode("#", $dados);  
                $pessoas[$cpf] = $arr;  
            }  
        }  
  
        fclose($fp);  
        print_r($pessoas);  
    }  
}
```

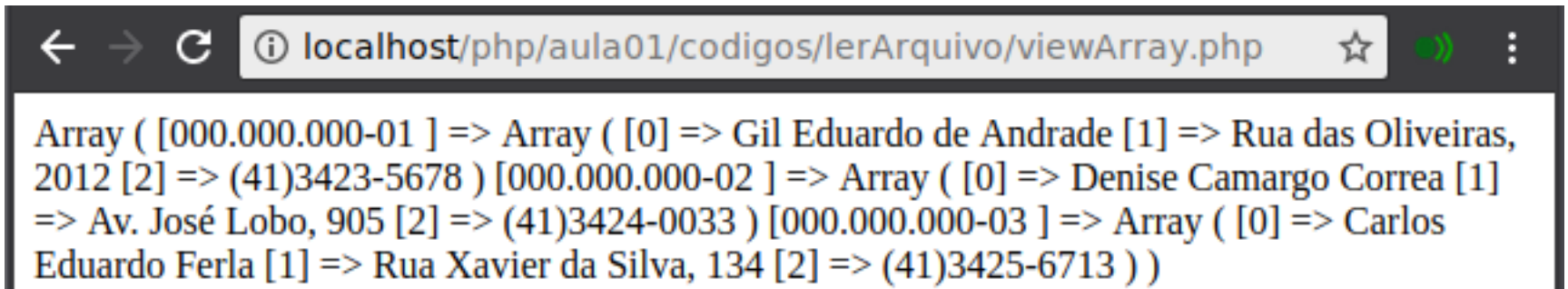
O exemplo engloba os arquivos ***viewArray.php*** e ***lerArquivoMontarArray.php*** e ***pessoas.txt***.





Formulário HTML – PHP

Exemplo Leitura Arquivo (interpretação/execução)



```
Array ( [000.000.000-01 ] => Array ( [0] => Gil Eduardo de Andrade [1] => Rua das Oliveiras, 2012 [2] => (41)3423-5678 ) [000.000.000-02 ] => Array ( [0] => Denise Camargo Correa [1] => Av. José Lobo, 905 [2] => (41)3424-0033 ) [000.000.000-03 ] => Array ( [0] => Carlos Eduardo Ferla [1] => Rua Xavier da Silva, 134 [2] => (41)3425-6713 ) )
```

Resultado da execução

Os códigos apresentados no slide anterior encontram-se disponíveis na seção “dicas de aula” – localização: “lerArquivo/viewArray.php” e “lerArquivo/lerArquivoMontarArray.php” .

Os detalhes da implementação são explicados durante a aula.



ARQUIVO TEXTO – PHP

Exemplo Escrita Arquivo

```
include_once('escreverArquivoArray.php');

$peessoas = array(
    "000.000.000.01" => array(
        "nome" => "Carlos Eduardo",
        "endereco" => "Rua Manuel Viana, 200",
        "telefone" => "(41)3422-2829",
    ),

    "000.000.000.02" => array(
        "nome" => "Fabiola Oliveira",
        "endereco" => "Rua Antonio Carlos Rodrigues, 2064",
        "telefone" => "(41)99912-0214",
    )
);

escreverArquivoArray($peessoas);
```

O exemplo engloba os arquivos ***view.php*** e ***escreverArquivoArray.php***.





ARQUIVO TEXTO – PHP

Exemplo Escrita Arquivo

```
function escreverArquivoArray($arr) {  
  
    $fp = fopen('pessoas.txt', 'a+');  
  
    if ($fp) {  
        foreach($arr as $cpf => $dados) {  
            if(!empty($dados)) {  
                $linha = $cpf." - ".$dados['nome']." - ".$dados['endereco'];  
                fputs($fp, "$linha\n");  
            }  
        }  
  
        fclose($fp);  
    }  
  
    echo "[OK] Dados escritos com Sucesso!";  
}
```

O exemplo engloba os arquivos ***view.php*** e ***escreverArquivoArray.php***.





Formulário HTML – PHP

Exemplo Leitura Arquivo (interpretação/execução)



```
[OK] Dados escritos com Sucesso!
```

```
pessoas.txt
1 000.000.000.01 - Carlos Eduardo - Rua Manuel Viana, 200 - (41)3422-2829
2 000.000.000.02 - Fabiola Oliveira - Rua Antonio Carlos Rodrigues, 2064 - (41)99912-0214
```

Resultado da execução

Os códigos apresentados no slide anterior encontram-se disponíveis na seção “dicas de aula” – localização: “escreverArquivo/view.php” e “escreverArquivo/escreverArquivoArray.php” .

Os detalhes da implementação são explicados durante a aula.



Conceitos Iniciais

Exemplos Utilizados no Documento (Conceitos Iniciais)

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_exdoc01.zip

Exemplos Utilizados no Documento (Formulário /Arquivo)

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_dica01.zip

Exercícios sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_pratica01.pdf

