



Programação para Internet

Módulo 6

Introdução à Websites Dinâmicos com PHP

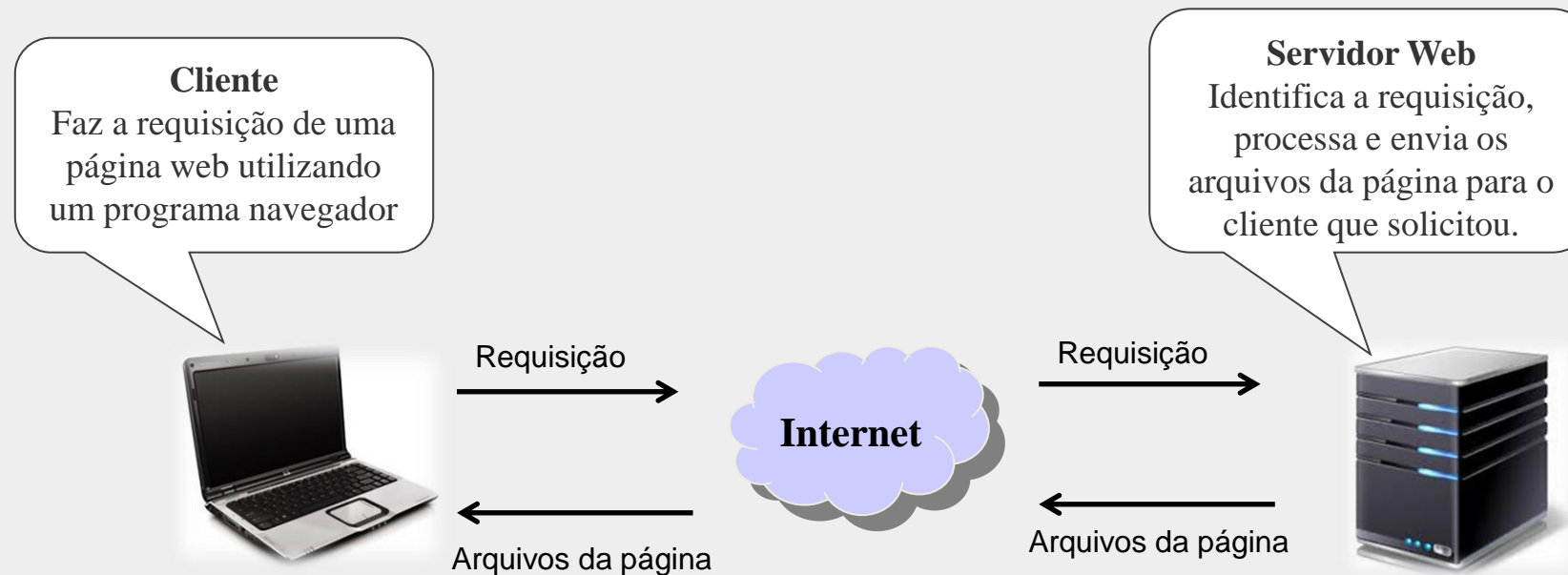
Prof. Dr. Daniel A. Furtado - FACOM/UFU

Conteúdo protegido por direito autoral, nos termos da Lei nº 9 610/98

A cópia, reprodução ou apropriação deste material, total ou parcialmente, é proibida pelo autor

Servidor Web

- Um **servidor web** (ou servidor HTTP) é um software no servidor capaz de receber as requisições HTTP do navegador de internet (solicitando arquivos como HTML, CSS, imagens etc.), e enviar de volta as respostas com os dados solicitados;
- O termo servidor web também é utilizado para referenciar o computador em si onde o servidor HTTP está instalado e em execução.



Websites Estáticos x Dinâmicos

■ Websites Estáticos

- Conteúdo sempre o mesmo
- HTML, CSS, JavaScript, imagens etc.
- Sem acesso a banco de dados
- Servidor web estático (ex. Apache HTTP)
- Exemplos: todos os websites criados anteriormente

■ Websites Dinâmicos

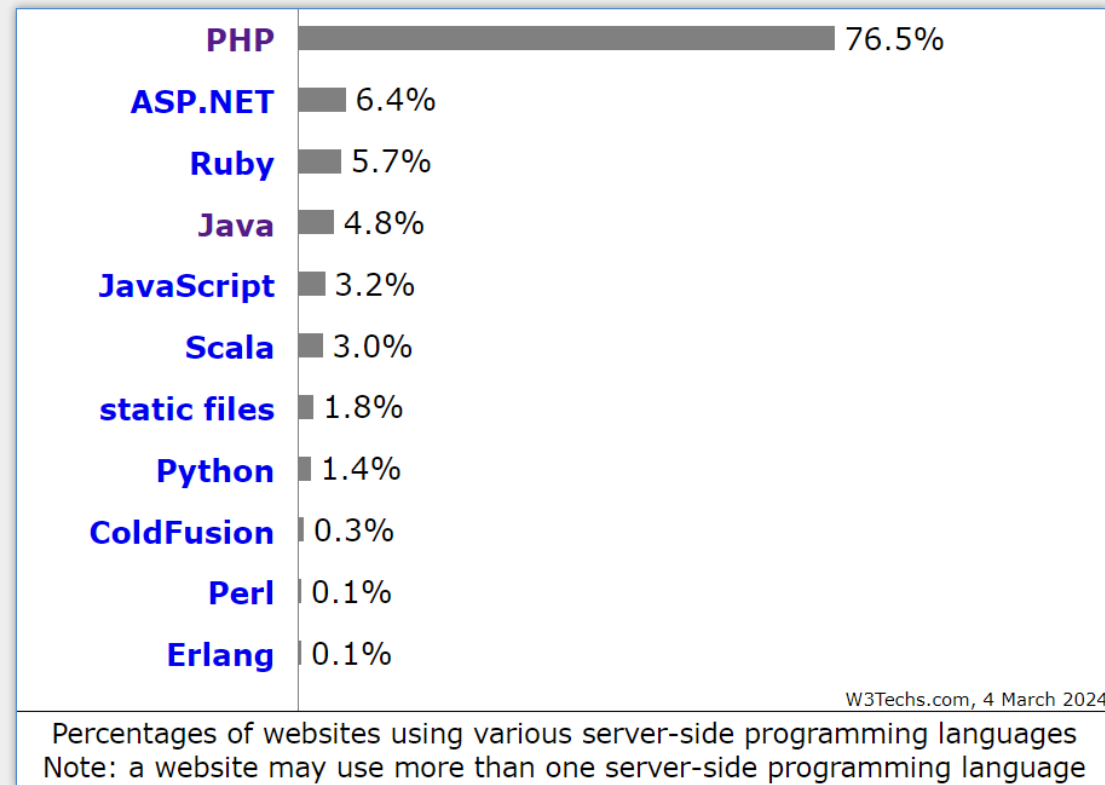
- Conteúdo produzido dinamicamente
- Envolve programação **server-side**: PHP, Python, Java etc.
- Realizam processamentos adicionais no servidor como:
 - Processamento de formulários, acesso à bancos de dados, gerenciamento de sessões, controle de login etc.
- Servidor web dinâmico (ex. Apache HTTP + PHP)

Websites Dinâmicos com PHP

- PHP é um acrônimo recursivo para **PHP: Hypertext Preprocessor**
- É uma linguagem de script *server-side* que permite o desenvolvimento de websites dinâmicos
- Scripts em PHP são normalmente executados no servidor, em conjunto com o servidor HTTP
- PHP é *open source* e há vários serviços de hospedagem que oferecem suporte gratuitamente
- Muito popular e de fácil aprendizado
 - Permite introduzir com facilidade conceitos fundamentais relacionados à programação *server-side* que vão além da linguagem em si como o de página dinâmica, requisições HTTP assíncronas, técnica Ajax, aspectos de segurança, serviços web, comunicação com banco de dados etc.

Linguagens Server-Side

Segundo o W3Techs, PHP é utilizada em 76% dos websites que se tem conhecimento da linguagem *server-side*



Fonte: W3Techs (Março/2024)

Exemplos de websites populares que utilizam PHP

- facebook.com
- microsoft.com
- wikipedia.org
- pinterest.com
- yahoo.com
- mozilla.org
- wordpress.com

O que preciso para começar a trabalhar com PHP?

- É possível utilizar um serviço de hospedagem com suporte a PHP
 - Há serviços pagos e gratuitos
 - Início em poucos minutos
- Outra opção é instalar e configurar um servidor web com PHP localmente
 - Inicia-se pela instalação de um servidor HTTP (ex. Apache HTTP)
 - Em seguida deve-se instalar e configurar o PHP (php.net/downloads)
 - Tutorial: <https://furtado.prof.ufu.br/site/teaching/PPI/apache-php-mysql-tutorial-instalacao.pdf>
- Outra possibilidade é utilizar pacotes de instalação que incluem o PHP
 - Ex.: WAMP (**W**indows, **A**pache, **M**ySQL and **P**HP)

Hello World com PHP

Página Dinâmica do tipo Hello World com PHP

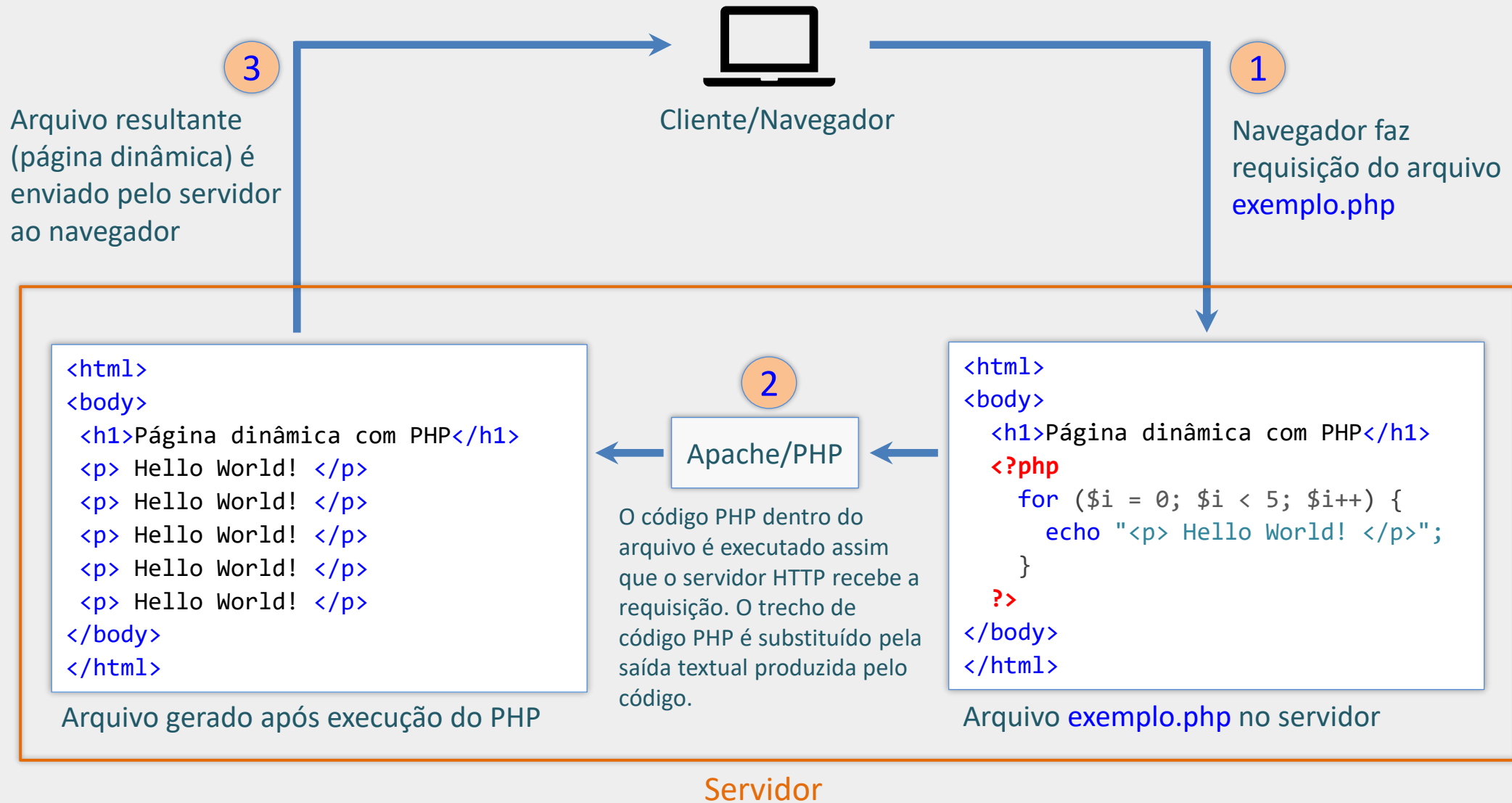
```
<html>
<body>
  <h1>Página dinâmica com PHP</h1>
  <?php
    for ($i = 0; $i < 10; $i++) {
      echo "<p>Hello World!</p>";
    }
  ?>
</body>
</html>
```

Exercício: colocar os exemplos disponibilizados no link a seguir online e testá-los começando pelo exemplo **hello**.

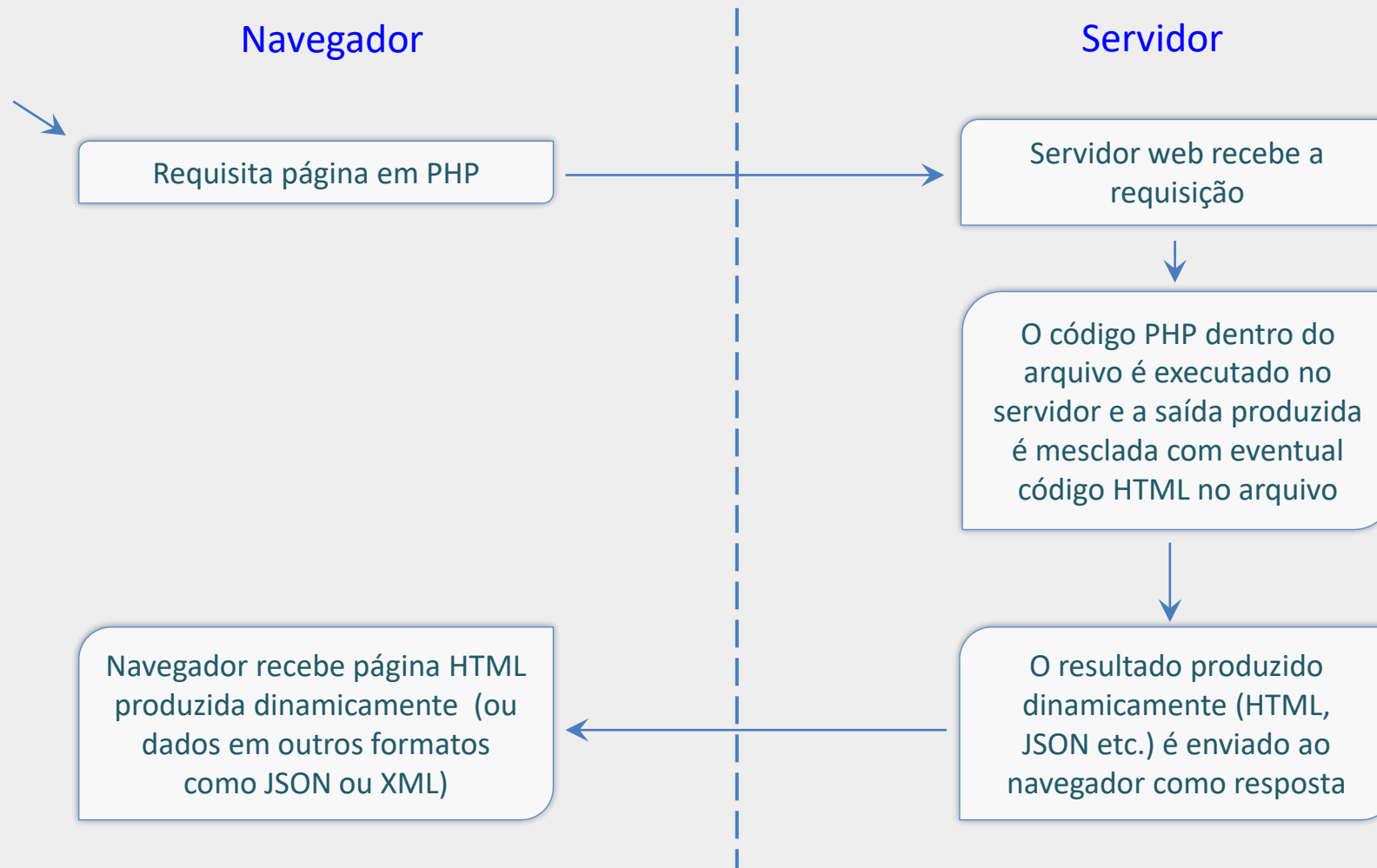
<https://furtado.prof.ufu.br/site/teaching/PPI/Exemplos-PHP.zip>

- O arquivo com código PHP deve ter a extensão .php
- O código PHP deve ser inserido dentro de `<?php ?>`
- O arquivo .php não precisa conter apenas código PHP (pode conter HTML, CSS etc.)
- PHP é um pré-processador de hipertexto: os trechos de código PHP do arquivo são executados no servidor e a saída produzida é mesclada com o conteúdo restante do arquivo

Funcionamento Básico do PHP – Hello World



Funcionamento Básico do PHP



Arquivo PHP

- Em PHP, utiliza-se bastante o construtor `echo` (apresentado no código do slide anterior), para produzir uma saída textual
- Caso o arquivo tenha múltiplos trechos de código PHP, eles serão executados na ordem em que aparecem no arquivo
 - As eventuais saídas produzidas serão mescladas, respectivamente, com o restante do conteúdo do arquivo
- Um arquivo PHP pode conter apenas código PHP
 - Por exemplo, é possível criar um script PHP que receba uma requisição HTTP com parâmetros, acesse um banco de dados e retorne um conteúdo no formato JSON

Recursos Básicos da Linguagem

Observações Gerais

- Declarações terminam com o ponto-e-vírgula
- Comentários de linha: `// comentário`
- Comentários de bloco: `/* comentário */`
- Sensível a letras maiúsculas e minúsculas
 - Nomes de variáveis e constantes
 - Chaves de arrays associativos
- Não sensível a maiúsculas e minúsculas
 - Palavras reservadas da linguagem
 - Nomes de classes, métodos e funções
- Os operadores aritméticos, relacionais e lógicos são similares ao de linguagens tradicionais como Java, JavaScript e C++
 - Porém o operador para concatenar strings é o ponto (`"string a" . "string b"`)

Estruturas Condicionais e de Repetição

```
if (expressão) {  
    // operações se verdadeiro  
}  
else {  
    // operações se falso  
}
```

```
switch (expressao) {  
    case condicao1:  
        // operações  
        break;  
  
    case condicaoN:  
        // operações  
        break;  
  
    ...  
    default:  
        // operações  
}
```

```
for ($i = 0; $i < 10; $i++)  
{  
    // operações  
}
```

```
foreach ($array as $elem)  
{  
    // operações  
}
```

```
while (expressao)  
{  
    // operações  
}
```

```
do {  
    // operações  
} while (expressao)
```

Variáveis e Constantes

- Começam com o símbolo **\$**
- São declaradas na primeira atribuição
- O tipo é definido automaticamente
- Constantes podem ser definidas com o construtor **define**

```
$x = 5;           // variavel do tipo integer
$y = -6;         // variavel do tipo integer
$z = 3.14;       // variavel do tipo float
$str = "bla bla"; // variavel do tipo string
$cond = true;    // variável booleana
$pares = [2, 4, 6]; // array
define('DB_NAME', 'mysqlTeste');
define('DB_PSWD', '123456');
```


Strings e o construtor echo

- Podem ser definidas com **aspas duplas**
 - Conteúdo da string é avaliado
 - Pode conter nomes de variáveis
- Podem ser definidas com **aspas simples**
 - String não avaliada
 - Apenas texto

```
$idade = 15;
$mes = 10;
$dist = 30;

echo "A idade eh $idade"; // a saída será: A idade eh 15
echo 'A idade eh $idade'; // a saída será: A idade eh $idade

// Para imprimir uma variável sem espaço/caracter especial depois
// so seu nome, coloque o nome da variável entre chaves
echo "Distância de {$dist}km"; // Distância de 30km
```

String Heredoc

- PHP permite criar strings especiais com a sintaxe **Heredoc**
 - Coloca-se um **texto delimitador** no início da string, precedido por `<<<`
 - Repete-se o texto delimitador no final da string, sem `<<<`, seguido de `;`
 - Permite criar strings longas, de múltiplas linhas, sem aspas
 - O conteúdo da string é avaliado
- Não pode haver espaços ou outros caracteres depois do texto delimitador

```
$str = <<<MINHA_STRING_HEREDOC
```

```
Texto com múltiplas linhas.
```

```
Aqui dentro posso utilizar aspas simples '
```

```
e também aspas duplas "
```

```
assim como nomes de variáveis como $exemplo
```

```
MINHA_STRING_HEREDOC;
```

String Heredoc - Exemplo

Exemplo

```
$maxSal = 10;
$str = <<<SQL
    SELECT *
    FROM Funcionario
    WHERE salario > $maxSal and
           codDepart = 'Vendas'
SQL;
echo $str;
```

Saída produzida

```
SELECT *
FROM Funcionario
WHERE salario > 10 and
           codDepart = 'Vendas'
```

- O identificador de fechamento só pode ser indentado a partir do [PHP 7.3](#)
- O espaçamento utilizado na linha do delimitador de fechamento (antes de `SQL;` no exemplo acima) será removido de cada linha da string. Se não for possível remover, ocorrerá um erro (veja o próximo slide)

String Heredoc - Exemplo

```
$maxSal = 10;  
$str = <<<SQL  
    SELECT *  
        FROM Funcionario  
        WHERE salario > $maxSal and  
              codDepart = 'Vendas'  
SQL;
```

Erro de sintaxe. Primeira linha da string não começa com a quantidade mínima de espaços, conforme delimitador de fechamento.

Funções

Função simples

```
function max($a, $b)
{
    if ($a > $b)
        return $a;
    else
        return $b;
}
```

Parâmetros com tipos definidos

```
function max(float $a, float $b)
{
    if ($a > $b)
        return $a;
    else
        return $b;
}
```

Chamada da função

```
$maior = max(2, 5);
$a = 10;
```

Variáveis Globais

```
<?php

$x = 10;    // $x é global

function exemplo() {
    echo $x;    // erro
    global $x;
    echo $x;    // 10
}

?>
```

É necessário utilizar `global` para acessar variável global dentro de funções

Array Indexado

```
// Define array com números pares
$ pares1 = [2, 4, 6];

// Define array vazio e adiciona elementos no final
$ pares2 = [];
for ($i = 2; $i <= 10; $i+=2) {
    $ pares2[] = $i; // acrescenta elemento no final
};

// Percorre array com estrutura for simples
$n = count($pares2);
for ($i = 0; $i < $n; $i++) {
    echo $pares2[$i];
};

// Percorre array com estrutura foreach
foreach ($pares2 as $par) {
    echo $par;
};
```

Array Associativo

- Coleção de pares do tipo **chave => valor**
- Cada elemento é acessado por meio da **chave** (*key*)
- A chave pode ser **string** ou **inteiro**
- O valor pode ser de qualquer tipo

```
$alunos = [  
    "GSI010" => "Augusto",  
    "GSI011" => "Camila",  
    "GSI012" => "Pedro"  
];  
  
echo $alunos['GSI011']; // a saída será 'Camila'  
echo $alunos[0];      // ocorrerá um erro!
```


Arrays Super Globais

- Arrays **super globais** são arrays associativos especiais que podem ser acessados de qualquer lugar sem utilizar a palavra reservada **global**
- Veja alguns exemplos a seguir

Array	Descrição
<code>\$GLOBALS</code>	Permite acessar as variáveis globais do script
<code>\$_GET</code>	Permite acessar os campos de formulários submetidos pelo método GET
<code>\$_POST</code>	Permite acessar os campos de formulários submetidos pelo método POST
<code>\$_FILES</code>	Permite acessar arquivos enviados anexados a formulários
<code>\$_COOKIE</code>	Permite resgatar os cookies enviados ao servidor na requisição HTTP
<code>\$_SESSION</code>	Permite acessar e criar variáveis de sessão (para controle de login, por exemplo)
<code>\$_SERVER</code>	Permite resgatar informações sobre o servidor, sobre as requisições HTTP etc.

Exemplos de Chaves do Array Super Global `$_SERVER`

Array	Descrição
<code>\$_SERVER['PHP_SELF']</code>	Retorna o nome do arquivo PHP em execução
<code>\$_SERVER['REQUEST_METHOD']</code>	Retorna o método da requisição HTTP utilizado para acessar a página
<code>\$_SERVER['HTTP_USER_AGENT']</code>	Contém informações sobre o cliente HTTP (navegador) que fez a requisição
<code>\$_SERVER['REMOTE_ADDR']</code>	Retorna o endereço IP do usuário que está acessando a página

Classes e Objetos

```
class Circulo
{
    // Declaração de atributos.
    // O modificador de acesso padrão
    // é public (caso seja omitido)
    private $raio;
    private $area;

    // Construtor
    function __construct($raio)
    {
        $this->raio = $raio;
        $this->area = 3.14*$raio*$raio;
    }

    // Declaração de método
    public function mostraArea()
    {
        echo $this->area;
    }
}
```

```
<?php
$circ1 = new Circulo(5);
$circ1->mostraArea();
?>
```

Processamento de Formulários

Recebendo Formulários - Método POST

Formulário HTML - Método POST

```
<form action="processaForm.php" method="POST">  
  Nome: <input type="text" name="nome" >  
  Email: <input type="email" name="email">  
</form>
```



```
<?php  
  $nome = $_POST["nome"];  
  $email = $_POST["email"];  
  echo "Sr. $nome, seu e-mail é $email";  
?>
```

Arquivo processaForm.php

Atenção: Utilize o atributo **name** para fornecer um nome de identificação para o campo.

Posteriormente, utilize essa identificação para recuperar o dado no PHP.

Um **erro** comum é utilizar o atributo **id** para esse propósito.

Recebendo Formulários - Método GET

Formulário HTML - Método GET

```
<form action="processaForm.php" method="GET">  
  Nome: <input type="text" name="nome">  
  Email: <input type="email" name="email">  
</form>
```



```
<?php  
  $nome = $_GET["nome"];  
  $email = $_GET["email"];  
  echo "Sr. $nome, seu e-mail é $email";  
?>
```

Arquivo `processaForm.php`

Recebendo Parâmetros pela URL

Exemplo de links com passagem de parâmetros pela URL

```
<a href="detalhes.php?codProd=1"> Notebook </a>  
<a href="detalhes.php?codProd=2"> Celular </a>
```



```
<?php  
    $codigoProduto = $_GET["codProd"];  
    // busca pelo produto  
?>
```

Exemplo de resgate do parâmetro em arquivo PHP

Formulários com Campos Vazios

```
<?php
    $nome = isset($_POST["nome"]) ? $_POST["nome"] : "";
    $email = isset($_POST["email"]) ? $_POST["email"] : "";
    echo "Sr. $nome, seu e-mail é $email";
?>
```

Utilizando a função **isset** em conjunto com o operador **?**

Para prevenir eventuais erros causados por campos vazios, pode-se utilizar a função **isset** do PHP para verificar se a variável está definida.

```
<?php
    $nome = $_POST["nome"] ?? "";
    $email = $_POST["email"] ?? "";
    echo "Sr. $nome, seu e-mail é $email";
?>
```

Forma equivalente utilizando apenas o operador **??**

Uma alternativa mais prática é utilizar o operador **??**, que permite resgatar o valor da variável apenas caso ela esteja definida.

Exemplo Simples de Validação dos Dados no Servidor

```
<?php
    $nome = $_POST["nome"] ?? "";

    // Não deixe de validar os campos
    if (trim($nome) == "")
        $errorMsg = "O nome é obrigatório";
?>
```

A validação dos campos no lado servidor é recomendada mesmo que eles já tenham sido validados no lado cliente. A função **trim** retorna uma nova string após remover eventuais espaços no início e no fim da string passada como parâmetro (assim como tabulações e quebras de linhas)

Cuidado com Ataques XSS!

```
<html>
<body>
  <h1>Site Vulnerável à Ataques XSS</h1>
  <?php
    $nome = $_GET["nome"];
    $email = $_GET["email"];
    echo <<<HTML
    <table>
      <tr>
        <td> $nome </td>
        <td> $email </td>
      </tr>
    </table>
    HTML;
  ?>
</body>
</html>
```

Cross-Site Scripting (XSS)

é um tipo de ataque malicioso onde um código potencialmente prejudicial é injetado utilizando a URL ou campos de formulário.

Este exemplo está vulnerável a ataques XSS, uma vez que cria uma página dinâmica inserindo diretamente na tabela HTML os dados oriundos de campos de formulário, produzidos pelo usuário, sem qualquer sanitização.

Cuidado com Ataques XSS!

```
<html>
<body>
  <h1>Prevenindo XSS</h1>
  <?php
    $nome = $_GET["nome"];
    $email = $_GET["email"];
    ...
    $nome = htmlspecialchars($nome);
    $email = htmlspecialchars($email);
    echo <<<HTML
    <table>
      <tr>
        <td> $nome </td>
        <td> $email </td>
      </tr>
    </table>
    HTML;
  ?>
  ...
```

htmlspecialchars

Pode ser utilizada para "sanitizar" um conteúdo produzido pelo usuário, **antes de inserir** esse conteúdo na página HTML

htmlspecialchars converte, por padrão:

- < em <
- > em >
- & em &
- " em "

Mas Atenção: não é uma boa prática armazenar o conteúdo (no banco de dados, por exemplo) após passar pela função **htmlspecialchars**

Manipulando Senhas

```
<?php
```

```
// o hash gerado terá 60 caracteres, mas pode aumentar  
$senhaHash = password_hash($senha, PASSWORD_DEFAULT);
```

```
// armazenar o hash da senha e não a senha  
gravaNoBD($senhaHash);
```

```
?>
```

Cadastro do Usuário

Uso da função **password_hash** do PHP para gerar um hash de senha seguro utilizando o algoritmo **BCrypt** antes de armazenar

```
<?php
```

```
// resgatar do BD a senha hash do usuário  
$senhaHash = resgataSenhaHashDoBD();
```

```
if (password_verify($senhaFornecida, $senhaHash) {  
    // Login efetuado com sucesso!
```

```
}
```

```
?>
```

Validação de Login

Uso da função **password_verify** para comparar a senha fornecida com o hash da senha armazenado.

Redirecionando com PHP

```
<?php
    // pagina.php
    header("Location: nova-pagina.php");
    exit();
?>
```

- **header** envia um cabeçalho HTTP
- Deve ser chamada antes do script produzir qualquer saída
 - Antes de código HTML, antes de **echo**, etc.
- É uma boa prática chamar **exit()** depois do redirecionamento para garantir que nenhum conteúdo seja produzido a partir desse ponto

Uso Inadequado da Função header

```
<html>
<body>
  <h1>Ex. de uso INADEQUADO da função header</h1>

  <?php
    header("Location: nova-pagina.php");
    exit();
  ?>

</body>
</html>
```

Não faça isso!

Neste exemplo a função `header` não terá efeito, pois o arquivo PHP produz uma saída HTML antes de sua chamada (por causa do código HTML no início do arquivo)

include e require

- **include** e **require** incluem outro arquivo PHP no script
- O arquivo incluído pode conter variáveis, funções, classes, etc.
- **include** gera um *warning* em caso de falha na inclusão
- **require** gera um *erro fatal* e encerra o script
- Não são funções!

```
<?php
    // mysqlConnection.php pode conter
    // dados de conexão com o MySQL,
    // por exemplo
    include "mysqlConnection.php";
?>
```

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Cadastro de Contato</title>
</head>
<body>
  <form action="cadastraContato.php" method="post">
    <div>
      <label for="nome">Nome:</label>
      <input type="text" id="nome" name="nome">
    </div>
    <div>
      <label for="email">Email:</label>
      <input type="email" id="email" name="email">
    </div>
    <div>
      <label for="telefone">Telefone:</label>
      <input type="tel" id="telefone" name="tel">
    </div>
    <button>Cadastrar</button>
  </form>
</body>
</html>

```

Arquivo [formCadastro.html](#)

```

<?php

require "contatos.php";

// coleta os dados do formulário
$nome = $_POST["nome"] ?? "";
$email = $_POST["email"] ?? "";
$tel = $_POST["tel"] ?? "";

// cria um novo contato e armazena no arquivo
$novoContato = new Contato($nome, $email, $tel);
$novoContato->AddToFile("contatos.txt");

// redireciona para a página de cadastro
header("location: formCadastro.html");

?>

```

Arquivo [cadastraContato.php](#)

Observe que este arquivo depende do arquivo **contatos.php**. Tal arquivo contém a definição da classe **Contato** e do método **AddToFile**. O exemplo completo está disponível no website.

include_once e require_once

- `include_once` e `require_once` são similares a `include` e `require`
- Porém, não incluem o arquivo caso ele já tenha sido incluído
- `include` e `require` produzem warnings/erros nessa situação

Referências

- <https://www.php.net/docs.php>
- NIXON, R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. 5. ed. O'Reilly Media, 2018