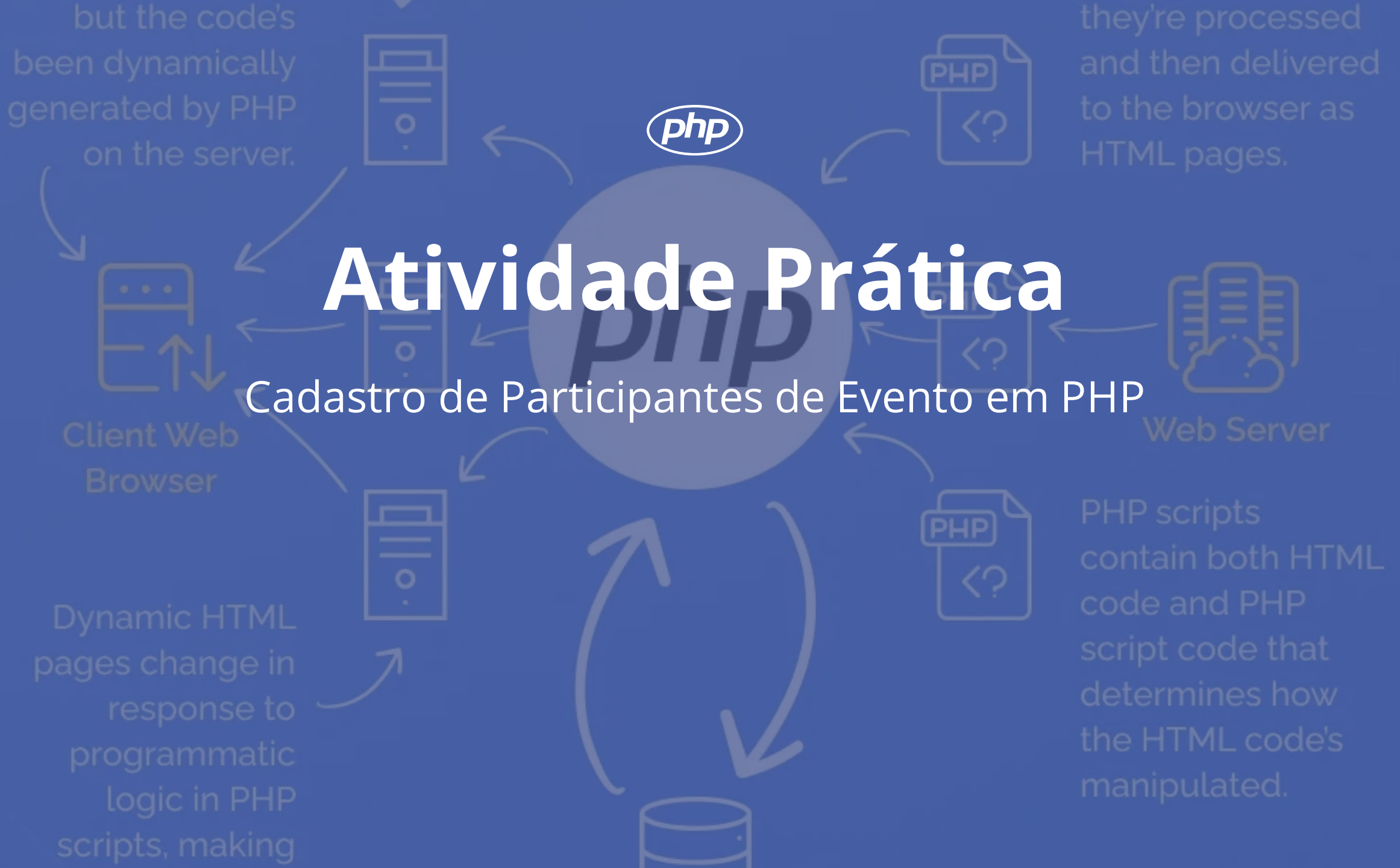


# Atividade Prática

## Cadastro de Participantes de Evento em PHP



# Objetivo

---

- ✓ Criar uma aplicação simples utilizando **HTML** e **PHP** para cadastro de participantes de um evento de tecnologia.
- ✓ Desenvolver um formulário para coleta de dados dos participantes.
- ✓ Implementar tratamento e validação adequada das informações no backend.
- ✓ Exibir os dados processados na tela ou armazená-los de forma segura.

# Enunciado

---

Imagine que você está desenvolvendo o site de uma feira de tecnologia. Os organizadores pediram uma página onde os interessados possam se inscrever para participar das atividades do evento.

## ☰ Sua tarefa:




Criar um formulário de cadastro e um script PHP que receba, trate e valide os dados enviados pelos participantes interessados no evento.

# Campos do Formulário

Campo	Tipo	Regras de Validação
Nome completo <div>Obrigatório</div>	Texto	Remover espaços extras.
E-mail <div>Obrigatório</div>	Email	Validar formato de e-mail.
Idade <div>Opcional</div>	Número	Se informado, deve ser igual ou maior que 16 anos.
Área de interesse <div>Obrigatório</div>	Select (lista)	Deve escolher uma das opções disponíveis.
Deseja receber e-mails? <div>Opcional</div>	Checkbox	Valor "Sim" se marcado, "Não" se não marcado.

# Instruções - Arquivo HTML






---

-  Crie um arquivo HTML com o formulário contendo todos os campos especificados.
-  O formulário deve enviar os dados via método **POST** para um arquivo PHP (processa\_cadastro.php).
-  Utilize os tipos de entrada apropriados para cada campo (text, email, number, select, checkbox).

```
<form action="processa_cadastro.php" method="POST">
  <label for="nome">Nome completo:</label>
  <input type="text" id="nome" name="nome" required>
  ...
</form>
```

# Instruções - Arquivo PHP

---

-  Crie um arquivo `processa_cadastro.php` para receber os dados do formulário.
-  Capture todos os dados enviados via método **POST**.
-  Realize o tratamento adequado dos dados: remoção de espaços, proteção contra XSS e validação de formato.
-  Exiba mensagens de erro caso alguma validação falhe.
-  Exiba um resumo dos dados cadastrados caso todas as validações sejam bem-sucedidas.

```
$nome = $_POST['nome'];  
$email = $_POST['email'];  
// Tratamento e validação dos dados...
```

# Validações Necessárias

---



## Nome completo

Remover espaços extras.

```
$nome = trim($_POST['nome']);
```



## E-mail

Validar formato usando filter\_var().

```
if(!filter_var($email,  
FILTER_VALIDATE_EMAIL)) { /* erro */ }
```



## Idade

Verificar se é maior ou igual a 16 anos.

```
if(isset($_POST['idade']) && $_POST['idade']  
< 16) { /* erro */ }
```



## Proteção XSS

Usar htmlspecialchars() em todos os campos.

```
$dados = htmlspecialchars($dados, ENT_QUOTES,  
'UTF-8');
```

# Exemplo de Código

```
function validate_email($email) {
    if (filter_var = FILTER_VALIDATE_EMAIL) {
        return true;
    } return false;
}

function test_input($data) {
    $data = trim($data);
    $data = htmlspecialchars(ENT_QUOTES, ENT_QUOTES);
    return $data;
}

// protect xss-action XSS
$data = stripslashes($data);
}

/* Versão corrigida:
function validate_email($email) {
    if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
        return true;
    }
    return false;
}

function test_input($data) {
    $data = trim($data);
    $data = htmlspecialchars($data, ENT_QUOTES, 'UTF-8');
    $data = stripslashes($data);
    return $data;
}
*/
```

## </> Análise do Código

**validate\_email():** Função para validar o formato do e-mail usando `filter_var()`.

**test\_input():** Função para sanitizar dados de entrada:

- `trim()` - Remove espaços extras
- `htmlspecialchars()` - Protege contra XSS
- `stripslashes()` - Remove barras invertidas

**Observação:** O código original contém erros de sintaxe que precisam ser corrigidos para funcionar corretamente.