



Formulários



Campos input e textarea

Nesta primeira parte do estudo sobre formulários em HTML, focaremos nos campos input e textarea, elementos fundamentais para a construção de interfaces de usuário que envolvem a coleta de dados.

O Campo input

O campo input é um dos elementos mais versáteis em HTML, utilizado principalmente para capturar informações curtas e específicas, como nome, e-mail, senha, ou números. Sua sintaxe básica é a tag `<input>`, que deve sempre incluir o atributo `type`, determinando o tipo de dado que será inserido. Por exemplo, para capturar um texto, utilizamos `<input type="text">`.

Exemplo Prático:

```
<input type="text" id="nome_usuario" name="nome_usuario" required>
```

Neste exemplo, o campo input foi configurado para receber o nome do usuário. O atributo `id` identifica unicamente o campo, enquanto `name` associa o dado inserido a uma chave que será enviada ao servidor. O atributo `required` indica que o preenchimento desse campo é obrigatório.

Além de textos, o campo input pode ser utilizado para números:



```
<input type="number" id="quantidade" name="quantidade" step="1" required>
```

Aqui, o campo aceitará apenas números inteiros. O atributo step define o incremento permitido, e required novamente torna o preenchimento obrigatório.

O Campo textarea

Para textos mais longos e detalhados, o campo textarea é a escolha ideal. Diferente do input, ele é capaz de exibir e capturar grandes blocos de texto, como descrições, comentários ou mensagens.

A sintaxe do textarea é um pouco diferente:

```
<textarea id="mensagem" name="mensagem" required></textarea>
```

Neste exemplo, o campo textarea está configurado para capturar uma mensagem do usuário. Assim como o input, ele utiliza os atributos id e name para identificar o campo e associar o dado ao envio do formulário. O atributo required garante que o campo não seja deixado em branco.

Um aspecto notável do textarea é sua flexibilidade em termos de tamanho. Diferente do input, que possui um tamanho fixo, o textarea permite ao usuário ajustar sua área de inserção de texto conforme necessário. Esse comportamento pode ser controlado via CSS ou atributos HTML, caso seja necessário limitar essa flexibilidade.

Comparando input e textarea



Embora ambos os campos sejam usados para entrada de dados, input e textarea servem a propósitos distintos. O input é ideal para entradas concisas e bem definidas, como nomes e números, enquanto o textarea é mais apropriado para textos longos, onde é importante oferecer ao usuário espaço para expressar suas ideias ou fornecer descrições detalhadas.

Por exemplo, em um formulário de cadastro de produtos, o campo input pode ser utilizado para capturar o nome do produto e seu preço, enquanto o textarea pode ser usado para uma descrição detalhada do produto.

```
<form>
```

```
<label for="nome_produto">Nome do Produto:</label>
```

```
<input type="text" id="nome_produto" name="nome_produto" required>
```

```
<label for="descricao_produto">Descrição do Produto:</label>
```

```
<textarea id="descricao_produto" name="descricao_produto" required></textarea>
```

```
<label for="preco_produto">Preço:</label>
```

```
<input type="number" id="preco_produto" name="preco_produto" step="0.01"
required>
```

```
</form>
```

Neste exemplo, o input é usado para o nome e o preço do produto, enquanto o textarea é empregado para a descrição, proporcionando uma interface clara e funcional para a inserção de dados.

Campos combobox e fieldset

Agora, veremos sobre os campos ComboBox e FieldSet, elementos essenciais para a criação de formulários organizados e eficientes em HTML.



O Campo ComboBox

O ComboBox é utilizado para permitir a seleção de opções em um formulário. Quando você clica em um elemento e uma lista de opções aparece, isso é um ComboBox. Para criar um ComboBox em HTML, utilizamos a tag `<select>`, que gera o elemento de seleção. As opções disponíveis para o usuário dentro desse ComboBox são criadas usando a tag `<option>`.

Exemplo Prático:

```
<select id="produtos">
```

```
<option value="camisa">Camisa</option>
```


```
<option value="tenis" selected>Tênis</option>
```

```
<option value="bolsa">Bolsa</option>
```

```
</select>
```

Neste exemplo, a tag `<select>` cria um ComboBox com três opções: “Camisa”, “Tênis” e “Bolsa”. Cada opção é representada por uma tag `<option>`. O atributo `value` define o valor que será enviado ao servidor quando essa opção for selecionada. Note que a opção “Tênis” possui o atributo `selected`, o que faz com que ela apareça como a opção pré-selecionada quando a página é carregada.

O Campo FieldSet

O FieldSet é utilizado para agrupar elementos relacionados dentro de um formulário, tornando-o mais organizado e visualmente agradável. Ele  especialmente útil para dividir um formulário em seções lógicas. O FieldSet é acompanhado da tag <legend>, que serve como um título ou legenda para o grupo de campos.

Exemplo Prático:

```
<fieldset>
```

```
<legend>Cadastro de Produtos</legend>
```

```
<label for="nome_produto">Nome do Produto:</label>
```

```
<input type="text" id="nome_produto" name="nome_produto" required>
```

```
<label for="descricao_produto">Descrição do Produto:</label>
```

```
<textarea id="descricao_produto" name="descricao_produto" required></textarea>
```

```
<label for="preco_produto">Preço:</label>
```

```
<input type="number" id="preco_produto" name="preco_produto" step="0.01"
required>
```

```
<label for="produtos">Categoria:</label>
```

```
<select id="produtos">
```

```
<option value="camisa">Camisa</option>
```

```
<option value="tenis" selected>Tênis</option>
```

```
<option value="bolsa">Bolsa</option>
```

</select>



</fieldset>

Nesse exemplo, o FieldSet agrupa todos os campos relacionados ao cadastro de produtos, com uma legenda “Cadastro de Produtos” fornecida pela tag <legend>. O uso do FieldSet não apenas organiza o código, mas também melhora a apresentação visual do formulário no navegador, criando uma borda ao redor do grupo de campos e destacando a legenda.

Como Funciona no Navegador

Ao carregar o código acima em um navegador, o formulário exibirá uma borda ao redor dos campos agrupados pelo FieldSet, com a legenda “Cadastro de Produtos” no topo. Dentro desse grupo, o ComboBox permitirá ao usuário escolher entre “Camisa”, “Tênis” e “Bolsa”. Como o atributo selected foi atribuído à opção “Tênis”, essa opção aparecerá como a pré-selecionada. Se o atributo selected fosse removido ou aplicado a outra opção, o comportamento do ComboBox seria alterado de acordo.

Elementos label, email e data

Labels

O elemento label em HTML é essencial para associar um rótulo descritivo a um campo de entrada (input). Quando criamos um campo input sem nenhuma identificação, o label é o que nos permite mostrar ao usuário que aquele campo tem um propósito específico. Por exemplo, ao criar um campo para o código de um

produto, utilizamos um label para informar claramente que aquele campo é destinado à inserção do código.



Exemplo Prático:

```
<label for="codigo_produto">Código do Produto:</label>
```

```
<input type="text" id="codigo_produto" name="codigo_produto" required>
```

Neste exemplo, o label está associado ao campo input por meio do atributo for, que deve corresponder ao valor do atributo id do campo. Isso garante que o usuário saiba exatamente o que deve ser inserido em cada campo do formulário.

Campo de Email

O campo input com o tipo email é específico para a inserção de endereços de email. Este campo realiza uma validação básica para verificar se o valor inserido possui o formato típico de um email (incluindo um "@" e um domínio, como ".com"). Embora essa validação não verifique se o email realmente existe, ela garante que o formato esteja correto.

Exemplo Prático:

```
<label for="email_usuario">Email:</label>
```

```
<input type="email" id="email_usuario" name="email_usuario" required>
```

Aqui, o input é do tipo email, garantindo que o usuário insira um endereço de email no formato adequado.



Campo de Data

O campo input do tipo date é utilizado para a seleção de datas. Este campo não apenas aceita a inserção manual de uma data, mas também fornece uma interface amigável, geralmente um calendário, para que o usuário escolha a data desejada. Isso facilita a inserção de datas de forma precisa e intuitiva.

Exemplo Prático:

```
<label for="data_atualizacao">Data de Atualização:</label>
```

```
<input type="date" id="data_atualizacao" name="data_atualizacao" required>
```

Neste caso, o campo input é configurado para aceitar uma data, e o navegador exibirá um calendário quando o usuário clicar no campo, permitindo a seleção fácil de uma data.

Organização dos Campos em Tabela

Para melhorar a organização dos campos em nosso formulário, podemos utilizar uma tabela (table). Essa abordagem ajuda a estruturar o formulário de maneira mais clara e visualmente agradável, facilitando a interação do usuário com a interface.

Exemplo Prático:

```
<fieldset>
```


<legend>Cadastro de Produtos</legend>



<table>

<tr>

<td><label for="codigo_produto">Código do Produto:</label></td>

<td><input type="text" id="codigo_produto" name="codigo_produto" required>
</td>

</tr>

<tr>

<td><label for="descricao_produto">Descrição do Produto:</label></td>

<td><input type="text" id="descricao_produto" name="descricao_produto"
required></td>

</tr>

<tr>

<td><label for="preco_produto">Preço:</label></td>

<td><input type="number" id="preco_produto" name="preco_produto"
step="0.01" required></td>

</tr>

<tr>

<td><label for="data_atualizacao">Data de Atualização:</label></td>

```
<td><input type="date" id="data_atualizacao" name="data_atualizacao" required>  
</td>  
  
</tr>  
  
</table>  
  
</fieldset>
```



Neste exemplo, os campos input são organizados dentro de uma tabela, tornando o formulário mais intuitivo e fácil de usar. Cada campo é claramente identificado por um label, garantindo que o usuário saiba exatamente o que deve ser inserido em cada campo.

Visualização no Navegador

Ao visualizar o código no navegador, você notará que os campos do formulário estão organizados em uma tabela, com cada campo identificado por um label. Isso torna o formulário mais fácil de navegar e preencher. No caso do campo de data, o usuário pode clicar no campo para abrir um calendário e selecionar a data desejada. Essa funcionalidade é especialmente útil para evitar erros na inserção de datas.

Tipos de Botões

Tipos de Botões em HTML

Existem diversos tipos de botões em HTML, cada um com uma função específica. Vamos focar nos quatro principais: botão padrão, botão de envio, botão de reset e

botão de imagem.



Botão Padrão: O botão padrão é definido com a tag `<button>` e o atributo `type="button"`. Ele é utilizado para executar ações genéricas, como chamar um script ou realizar uma navegação. O texto exibido no botão é definido entre as tags de abertura e fechamento.

Exemplo:

```
<button type="button">Clique Aqui</button>
```

Botão de Envio: O botão de envio, ou submit, é utilizado para enviar os dados do formulário ao servidor. Ele é criado usando a tag `<input>` com o atributo `type="submit"`. O texto exibido no botão é definido pelo atributo `value`.

Exemplo:

```
<input type="submit" value="Enviar">
```

Botão de Reset: O botão de reset é usado para limpar todos os campos de um formulário, restaurando os valores iniciais. Ele é criado com a tag `<input>` e o atributo `type="reset"`. O texto exibido no botão também é definido pelo atributo `value`.

Exemplo:

```
<input type="reset" value="Limpar">
```

Botão de Imagem: O botão de imagem é semelhante ao botão de envio, mas, em vez de exibir um texto, ele exibe uma imagem. Ele é criado com a tag `<input>` e o

atributo type="image", sendo necessário especificar o caminho da imagem usando o atributo src.



Exemplo:

```
<input type="image" src="caminho_da_imagem.png" alt="Enviar">
```

Aplicação Prática dos Botões

Vamos aplicar esses conceitos em um formulário HTML. Considerando nosso formulário anterior, adicionamos botões de envio e reset para ver como eles funcionam na prática.

Exemplo Prático:

```
<form>
```

```
<fieldset>
```

```
<legend>Cadastro de Produtos</legend>
```

```
<label for="codigo_produto">Código do Produto:</label>
```

```
<input type="text" id="codigo_produto" name="codigo_produto" required><br>
```

```
<br>
```

```
<label for="descricao_produto">Descrição do Produto:</label>
```

```
<input type="text" id="descricao_produto" name="descricao_produto" required>
```

```
<br><br>
```

```
<label for="preco_produto">Preço:</label>
```



```
<input type="number" id="preco_produto" name="preco_produto" step="0.01"
required><br><br>
```

```
<input type="reset" value="Limpar">
```

```
<input type="submit" value="Cadastrar">
```

```
</fieldset>
```

```
</form>
```

Neste exemplo, o botão de reset limpa todos os campos do formulário, enquanto o botão de envio submete os dados inseridos. Note que as tags `<input>` não exigem fechamento, e o navegador automaticamente renderiza esses elementos como botões com as funcionalidades específicas que definimos.

Comportamento no Navegador

Ao visualizar o formulário no navegador, você verá que os campos estão organizados e que os botões de “Limpar” e “Cadastrar” estão funcionando como esperado. Quando o usuário clica em “Limpar”, todos os dados preenchidos são removidos, retornando o formulário ao seu estado inicial. Ao clicar em “Cadastrar”, o formulário tenta enviar os dados inseridos, e, caso algum campo obrigatório não esteja preenchido, o navegador exibirá uma mensagem de erro solicitando o preenchimento.

Se, por exemplo, o campo “Código do Produto” não estiver preenchido e o usuário tentar enviar o formulário, o navegador destacará o campo, indicando que ele

precisa ser preenchido antes do envio. Isso ocorre devido ao atributo required, que impõe essa validação.



Métodos de envio e submissão de formulário

Métodos GET e POST

Os métodos GET e POST são amplamente utilizados para a submissão de formulários em HTML, cada um com características e finalidades específicas.

GET: O método GET é utilizado para solicitar dados de um recurso específico. Quando um formulário é enviado utilizando GET, os dados inseridos pelo usuário são anexados à URL como parâmetros. Esse método é ideal para situações em que se deseja recuperar ou visualizar dados, já que ele solicita informações ao servidor e as retorna ao navegador.

Exemplo Prático:


```
<form action="/buscarDados" method="get">
```

```
<label for="nome">Nome:</label>
```

```
<input type="text" id="nome" name="nome">
```

```
<input type="submit" value="Buscar">
```

```
</form>
```

Neste exemplo, os dados do formulário serão enviados como parte da URL, permitindo que o servidor processe a requisição e retorne os dados solicitados. 

POST: O método POST, por outro lado, é utilizado para enviar dados ao servidor para serem processados, como em um cadastro ou uma atualização de informações. Diferente do GET, os dados enviados pelo POST não são anexados à URL, mas são enviados no corpo da requisição, tornando-o mais seguro para a transmissão de informações sensíveis.

Exemplo Prático:

```
<form action="/cadastrarProduto" method="post">
```

```
<label for="produto">Produto:</label>
```

```
<input type="text" id="produto" name="produto">
```

```
<input type="submit" value="Cadastrar">
```

```
</form>
```

Neste caso, ao clicar em “Cadastrar”, os dados serão enviados ao servidor para processamento, utilizando o método POST.

Aplicação Prática dos Métodos

Vamos aplicar esses conceitos em nosso formulário HTML. Considerando o exemplo de um cadastro de produtos, podemos configurar o formulário para utilizar o método POST, que enviará os dados inseridos para uma URL específica para processamento.



Exemplo Prático:

```
<form action="/CadastrarStocHTML" method="post">

  <label for="codigo_produto">Código do Produto:</label>

  <input type="text" id="codigo_produto" name="codigo_produto" required><br>
<br>

  <label for="descricao_produto">Descrição do Produto:</label>

  <input type="text" id="descricao_produto" name="descricao_produto" required>
<br><br>

  <label for="preco_produto">Preço:</label>

  <input type="number" id="preco_produto" name="preco_produto" step="0.01"
required><br><br>


  <input type="submit" value="Cadastrar">

</form>
```

No código acima, o atributo `action` define a URL para onde os dados do formulário serão enviados, enquanto o atributo `method` especifica o método de envio, que neste caso é POST. Isso configura o formulário para enviar os dados ao servidor para que sejam processados.

Comportamento no Navegador

Ao visualizar o formulário no navegador, o usuário poderá preencher os campos e, ao clicar em “Cadastrar”, os dados serão enviados para a URL especificada no

atributo `action`. Embora o exemplo utilize uma URL fictícia (`/CadastrarStocHTML`), o processo demonstrado é como os formulários interagem com servidores no mundo real. 


Quando o método `POST` é utilizado, o navegador envia os dados no corpo da requisição, o que os torna invisíveis na URL, garantindo maior segurança, especialmente para dados sensíveis como senhas. Se o método `GET` fosse utilizado, esses dados apareceriam na barra de endereço do navegador, o que não é ideal para informações que precisam de sigilo.

Conclusão

Ao longo deste módulo, exploramos os principais componentes dos formulários em HTML, essenciais para o desenvolvimento de interfaces web interativas e funcionais. Iniciando com os campos `input` e `textarea`, passando pelos elementos de agrupamento como `combobox` e `fieldset`, e finalizando com a utilização de `label`, campos de email e data, bem como os diversos tipos de botões, cada conceito foi abordado com o objetivo de fornecer uma compreensão sólida sobre a criação e o gerenciamento eficaz de formulários.

Além disso, examinamos os métodos de envio e submissão de dados, destacando as diferenças entre os métodos `GET` e `POST` e suas aplicações práticas. Com esse conhecimento, os alunos estão capacitados a implementar formulários que não apenas atendem às necessidades de coleta de dados, mas também garantem a segurança e eficiência na transmissão dessas informações para o servidor. A compreensão desses fundamentos é vital para o desenvolvimento de aplicações web robustas e seguras.

Conteúdo Bônus

Para aprofundar os conhecimentos adquiridos sobre formulários em HTML, recomendo assistir ao vídeo “**Curso de HTML Completo: Aula 12 - Formulários**”  disponível no YouTube, criado pelo canal Programação Web. Neste vídeo, você encontrará uma abordagem prática e detalhada sobre a criação e manipulação de formulários em HTML, alinhada aos tópicos que foram estudados no curso, como campos de entrada, botões, e métodos de envio.

Referência Bibliográfica

ASCÊNCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores**. 3. ed. São Paulo: Pearson, 2012.

BRAGA, Paulo Henrique. **Teste de software**. São Paulo: Pearson, 2016.

GALLOTTI, Guilherme Moreira Alves (Org.). **Arquitetura de software**. São Paulo: Pearson, 2017.

GALLOTTI, Guilherme Moreira Alves. **Qualidade de software**. São Paulo: Pearson, 2015.

MEDEIROS, Edson. **Desenvolvendo software com UML 2.0 definitivo**. São Paulo: Pearson, 2004.

PFLEEGER, Shari Lawrence. **Engenharia de software: teoria e prática**. 2. ed. São Paulo: Pearson, 2003.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2019.

Prática Integradora Desenvolvimento de Software

Atividade Prática 4 – Criação e Validação de Formulários Web

Objetivos:



- Compreender e implementar os diferentes tipos de campos de entrada em formulários HTML.
- Utilizar elementos HTML como `<fieldset>`, `<legend>`, e `<select>` para criar formulários organizados e funcionais.
- Aplicar os métodos GET e POST para o envio de dados de formulários e entender as diferenças entre eles.

Materiais, Métodos e Ferramentas:

- Editor de código (Visual Studio Code, Sublime Text, etc.)
- Navegador web (Google Chrome, Mozilla Firefox, etc.)
- Conhecimento básico de HTML e CSS

Atividade Prática

Primeiramente, leia atentamente o texto a seguir:

Você é responsável por criar um formulário web para um sistema de cadastro de produtos. O formulário deve coletar informações detalhadas sobre cada produto e deve ser implementado usando HTML. Além disso, o formulário deve incluir elementos para validação e deve permitir a escolha de categorias de produtos.

Agora, vamos praticar!

PASSO A PASSO DETALHADO DA ATIVIDADE:

- **Criação da Estrutura Básica:**

- Crie um arquivo HTML chamado cadastro_produtos.html.



- No arquivo, adicione a estrutura básica de um documento HTML com o título “Cadastro de Produtos”.

- **Formulário com Campos Input e Textarea:**

- Dentro do <body>, adicione um formulário que deve incluir os seguintes campos:
 - Um campo de texto para o código do produto.
 - Um campo de texto para a descrição do produto.
 - Um campo numérico para o preço do produto.
 - Um campo de texto para a data de atualização do produto.

- **Agrupamento com Fieldset e Legend:**

- Utilize a tag <fieldset> para agrupar todos os campos relacionados ao cadastro do produto.
- Adicione uma <legend> com o texto “Cadastro de Produtos” para identificar a seção do formulário.

- **Campo ComboBox:**

- Adicione um campo de seleção (ComboBox) com as opções “Camisa”, “Tênis” e “Bolsa” para a categoria do produto.

- **Botões de Envio e Reset:**



- Adicione um botão de envio que submeterá o formulário.
- Adicione um botão de reset para limpar os campos do formulário.
- **Método de Envio do Formulário:**
 - Configure o formulário para usar o método POST e defina um URL fictício no atributo action, como /cadastrarProduto.
- **Campos de Validação:**
 - Certifique-se de que todos os campos obrigatórios estejam configurados com o atributo required.

Boa sorte com a atividade!

Gabarito Esperado

- O arquivo HTML deve incluir a estrutura básica e os campos solicitados.
- O formulário deve estar corretamente agrupado com <fieldset> e <legend>.
- O ComboBox deve ter as opções “Camisa”, “Tênis” e “Bolsa”.
- Os botões de envio e reset devem estar presentes e funcionando.
- O método de envio do formulário deve ser POST e o URL no atributo action deve ser uma URL fictícia.

- Todos os campos obrigatórios devem estar configurados com o atributo required.



Ir para exercício