

HTML5: Fundamentos para Construção de Páginas Web Modernas - Turma 2025B

5.1.1 Criando um formulário básico

A tag **<form>** serve como um contêiner para diferentes tipos de inputs e controles de formulário, como caixas de texto, botões de opção, caixas de seleção, botões de envio, entre outros.

Esta tag tem um conjunto de atributos que ajudam na sua formatação, de maneira que veremos abaixo os três atributos mais usados.

Atributo action

O atributo **action** da tag **<form>** especifica o URL do servidor ao qual os dados do formulário serão enviados para processamento. Esse atributo define a localização (endereço) onde os dados coletados pelo formulário devem ser encaminhados após o usuário submeter o formulário.

Veja no quadro abaixo um exemplo:

```
<form action="submit.php">  
    <!-- conteúdo do formulário -->  
</form>
```

Neste exemplo, **action** aponta para **submit.php**, indicando que os dados serão enviados para este script no servidor.

Observe que quando um formulário é submetido, os dados inseridos pelo usuário são empacotados e enviados ao servidor especificado no atributo **action**. O servidor, então, pode processar esses dados conforme necessário, o que pode incluir salvar as informações em um banco de dados, enviar e-mails, autenticar usuários, entre outras ações.

OBS: Quando não usados o atributo action, os dados são enviados para a própria página que inclui o formulário.

Atributo method

O atributo **method** da tag **<form>** define como os dados do formulário devem ser enviados ao servidor via protocolo HTTP. Ele pode ter dois valores principais: **GET** ou **POST**, cada um indicando um método HTTP diferente para a transmissão dos dados.

OBS: O HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação fundamental da World Wide Web, utilizado para transferir dados entre um navegador web (cliente) e um servidor web.

Para compreender a diferença entre esses dois métodos HTTP, é crucial explorar seu funcionamento básico. Sempre que um recurso na web é solicitado, seu navegador envia uma requisição a um determinado URL. Essa requisição HTTP se divide em dois componentes principais: o cabeçalho, que carrega metadados detalhando as capacidades do navegador, e o corpo, que pode incluir dados específicos requeridos pelo servidor para processar adequadamente a requisição.

Método GET

O **método GET** é utilizado por padrão para o envio de dados quando, na criação de um formulário, não se especifica explicitamente um método através do atributo method na tag **<form>**. Com esse método, os dados submetidos são anexados à URL da página destinada a processá-los, acompanhando o nome da página na própria URL. Assim, a URL incluirá os nomes e valores dos campos do formulário, visíveis para qualquer pessoa.

Veja um exemplo no quadro abaixo:

```
<form action="submit.php" method="get">
  <label for="query">Pesquisar:</label>
  <input type="text" id="query" name="query">
  <input type="submit" value="Enviar">
</form>
```

Veja como fica o formulário no navegador, conforme figura abaixo:

Pesquisar:

Figura 1 - Exemplo de uso da tag <form> no navegador.

Descrição: No navegador, este código cria um formulário de pesquisa. Ele contém uma etiqueta "Pesquisar:" seguida de um campo de texto onde o usuário pode digitar sua consulta. Abaixo do campo de texto, há um botão "Enviar" para enviar a consulta.

Quando o usuário preenche o campo de pesquisa e clica em "**Enviar**", os dados do formulário são enviados para o servidor web. Se o usuário digitar, por exemplo, "**IFRS**" no campo de pesquisa, a URL final para onde o navegador redireciona parecerá algo como mostrado no quadro abaixo:

submit.php?query=IFRS

Assim, o servidor, ao receber essa URL, processa os parâmetros (neste caso, **query=IFRS**) para realizar a ação desejada, como realizar uma busca no site com base na entrada do usuário.

Como os dados são dados submetidos são anexados à URL da página destinada a processá-los, há limitações de tamanho para a quantidade de dados que podem ser enviados (depende do navegador e do servidor). Este modelo é adequado, portanto, para buscas e qualquer situação em que os dados enviados não sejam sensíveis.

Método POST

O método **POST**, entretanto, é usado para enviar dados do cliente (normalmente, um navegador) para o servidor de uma maneira que não expõe as informações na URL, uma vez que transmite os dados no corpo da requisição HTTP. Dessa forma, esse método é adequado para transferir dados sensíveis ou grandes quantidades de informação.

Veja um exemplo, conforme quadro abaixo:

```
<form action="submit.php" method="post">
  <label for="name">Nome:</label>
  <input type="text" id="name" name="name" required>

  <label for="email">E-mail:</label>
  <input type="email" id="email" name="email" required>

  <label for="password">Senha:</label>
  <input type="password" id="password" name="password" required>

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

Veja na figura abaixo como fica o formulário o navegador.

Nome: E-mail: Senha:

Figura 2 - Exemplo de uso do atributo post da tag <form> exibido no navegador.

Descrição: No navegador, este código cria um formulário com campos para o usuário preencher. O formulário inclui: 1) Um campo de texto para o nome, identificado pela etiqueta "Nome: "; 2) Um campo de e-mail, identificado pela etiqueta "E-mail: "; 3) Um campo de senha, identificado pela etiqueta "Senha: "; 4) Um botão "Enviar" para submeter os dados. Todos os campos são obrigatórios.

Neste caso, quando o usuário preenche os campos do formulário e clica em "**Enviar**", os dados são enviados para o servidor web através de uma requisição HTTP do tipo **POST**. Os dados submetidos não são visíveis na URL, oferecendo uma camada adicional de privacidade em comparação ao método **GET**. O servidor web, ao receber os dados, pode processá-los conforme necessário — por exemplo, salvar as informações em um banco de dados ou autenticar o usuário com base no e-mail e senha fornecidos.

Atributo enctype

O atributo **enctype** da tag <form> especifica como os dados do formulário devem ser codificados ao serem enviados ao servidor. Este atributo é particularmente importante quando o formulário inclui operações de envio de arquivos.

Existem três valores possíveis para `enctype`:

- **application/x-www-form-urlencoded**: Este é o valor padrão se o atributo não for especificado. Com esse método, os caracteres são codificados de forma que se ajustem aos padrões da URL, com espaços convertidos em `+` e caracteres especiais convertidos em valores ASCII HEX.
- **multipart/form-data**: Este valor é necessário quando você está fazendo upload de arquivos através do formulário.
- **text/plain**: Com esse método, os dados são enviados sem nenhuma codificação. Isso é raramente utilizado e não é recomendado para a maioria das aplicações web, pois os dados podem ser enviados de uma forma que é difícil de interpretar pelo servidor web.

Veja no quadro abaixo um exemplo:

```
<form action="submit.php" method="post" enctype="multipart/form-data">  
  <!-- Campos do formulário -->  
</form>
```

Veremos outros exemplos do uso deste atributo mais adiante neste documento.

Atributo autocomplete

O atributo **autocomplete**, quando usado em um elemento `<form>`, controla a capacidade do navegador de preencher automaticamente os campos de formulário com dados previamente inseridos pelo usuário. Esse recurso é útil para formulários que requerem informações repetidas, como endereço, e-mail ou informações de pagamento, pois pode economizar tempo para o usuário e reduzir erros de digitação.

Os valores para **autocomplete** em um elemento `<form>` são os seguintes:

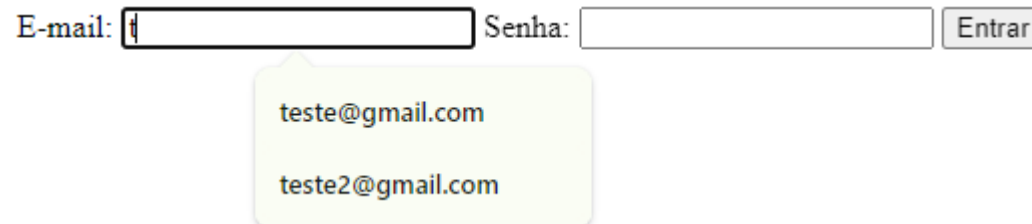
- **on**: ativa o preenchimento automático para todos os campos do formulário, a menos que o atributo **autocomplete** seja especificamente definido como **off** em um elemento `<input>` individual. Isso permite que o navegador preencha automaticamente os campos com dados correspondentes salvos anteriormente pelo usuário.
- **off**: Desativa a auto-completação para todos os campos do formulário. Isso pode ser útil em formulários que lidam com informações sensíveis ou únicas que não devem ser reutilizadas, como formulários de pagamento ou de alteração de senha.

Vejamos no quadro abaixo um exemplo ativando o preenchimento automático dos campos do formulário:

```
<form action="submit.php" method="post" autocomplete="on">  
  <label for="user-email">E-mail:</label>  
  <input type="email" id="user-email" name="email">  
  
  <label for="user-password">Senha:</label>  
  <input type="password" id="user-password" name="password">  
  
  <input type="submit" value="Entrar">  
</form>
```

Neste exemplo, o atributo **autocomplete="on"** no elemento **<form>** ativa o preenchimento automático para o formulário, permitindo que o navegador preencha automaticamente os campos de e-mail e senha, se informações correspondentes estiverem disponíveis.

Veja na figura abaixo o resultado no navegador.



The screenshot shows a web browser displaying a login form. The form consists of two input fields: "E-mail:" and "Senha:". The "E-mail:" field is active, and a dropdown menu is visible below it, showing two suggestions: "teste@gmail.com" and "teste2@gmail.com". To the right of the "Senha:" field is a button labeled "Entrar".

Figura 3 - Exemplo de uso do atributo `autocomplete="on"` da tag `<form>` exibido no navegador.

Descrição: No navegador, este código cria um formulário de login com dois campos: 1) Um campo de e-mail identificado pela etiqueta "E-mail: "; 2) Um campo de senha identificado pela etiqueta "Senha:". Há também um botão "Entrar" para submeter o formulário. O atributo `autocomplete="on"` permite que o navegador sugira automaticamente valores previamente usados para os campos.

Vejamos no quadro abaixo um exemplo desativando o preenchimento automático dos campos do formulário:

```
<form action="submit.php" method="post" autocomplete="off">
  <label for="new-email">E-mail:</label>
  <input type="email" id="new-email" name="email">

  <label for="new-password">Crie uma senha:</label>
  <input type="password" id="new-password" name="password">

  <input type="submit" value="Registrar">
</form>
```

Neste exemplo, o atributo **autocomplete="off"** no elemento **<form>** desativa o preenchimento automático para o formulário de registro, impedindo que o navegador preencha automaticamente os campos de e-mail e senha.

Veja na figura abaixo o resultado no navegador.

E-mail: Crie uma senha:

Figura 4 - Exemplo de uso do atributo `autocomplete="off"` da tag `<form>` exibido no navegador.

Descrição: No navegador, este código cria um formulário de registro com dois campos: 1) Um campo de e-mail identificado pela etiqueta "E-mail: "; 2) Um campo de senha identificado pela etiqueta "Crie uma senha:". Há também um botão "Registrar" para enviar o formulário. O atributo `autocomplete="off"` impede que o navegador sugira automaticamente valores previamente usados para os campos.

OBS: Os usuários podem ter configurado seus navegadores para ignorar as sugestões de preenchimento automático, afetando a eficácia desse atributo.

Atributo novalidate

O atributo **novalidate** na tag **<form>** é usado para desativar a validação do lado do cliente fornecida pelo HTML5 para o formulário em que é aplicado. Quando presente, impede que o navegador realize qualquer validação automática dos campos do formulário, como verificar se os campos obrigatórios (**required**) estão preenchidos, se o formato do email está correto (**type="email"**), entre outros.

O **novalidate** é útil em situações como:

- Quando você está desenvolvendo ou testando um formulário e não quer que as validações interfiram.
- Quando você prefere usar suas próprias funções de validação com JavaScript, oferecendo feedback customizado ou complexo que não pode ser conseguido com a validação automática do HTML5.

Vejamos no quadro abaixo um exemplo:

```
<form action="submit.php" method="post" novalidate>
  <label for="email">E-mail:</label>
  <input type="email" id="email" name="email" required>

  <label for="age">Idade:</label>
  <input type="number" id="age" name="age" min="18">

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

Neste exemplo, mesmo que o campo de e-mail esteja vazio ou não contenha um endereço de e-mail válido, e a idade seja menor que 18 (violando a regra **min="18"**), o navegador não exibirá mensagens de erro de validação ao tentar enviar o formulário.

OBS: Desativar a validação do lado do cliente não elimina a necessidade de validar os dados no lado do servidor.

Independentemente das validações do cliente, sempre valide e sane os dados do formulário no servidor para evitar injeções de código malicioso e garantir a integridade dos dados.

◀ 5.1 Trabalhando com formulários em documentos HTML

Seguir para...

5.1.2 Campos de entrada do usuário ►

[Baixar o aplicativo móvel.](#)