Trabalhando com Formulários HTML

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Objetivo

Dominar técnicas avançadas de criação e validação de formulários HTML, abordando atributos avançados, Regex, segurança e boas práticas para formulários complexos.

Tag <form>

A tag <form> define um formulário em HTML e é usada para enviar dados ao servidor. Os principais atributos da tag são:

- action: Define a URL para onde os dados do formulário serão enviados.
- method: Define o método HTTP usado para enviar os dados (geralmente GET ou POST).
- enctype: Define o tipo de codificação usado no envio dos dados (importante em uploads de arquivos).

Tag <form>

```html
<form action="/processa-dados" method="post" enctype="multipart/form-data">
    <!-- campos do formulário -->
    <button type="submit">Enviar</button>
  </form>
```

- action="/processa-dados": Os dados serão enviados para o endpoint /processa-dados no servidor.
- method="post": O método POST será usado para enviar os dados.
- enctype="multipart/form-data": Usado quando há upload de arquivos no formulário.

Diferença entre GET e POST

GET:

O método GET envia os dados do formulário anexados à URL como parâmetros de consulta (query strings).
É ideal para buscas ou quando os dados não são confidenciais.

```
1  <form action="/buscar" method="get">
2      <input type="text" name="query" placeholder="Buscar...">
3      <button type="submit">Buscar</button>
4    </form>
```

Diferença entre GET e POST

- Quando o formulário é enviado, a URL pode ser algo como:
- /buscar?query=palavra-chave
- Limitações: O tamanho dos dados é limitado (cerca de 2048 caracteres) e os dados ficam visíveis na URL

POST:

- O método POST envia os dados no corpo da requisição HTTP, sendo mais seguro que o GET.
- Ideal para envio de informações sensíveis ou grandes volumes de dados.

Diferença entre GET e POST

```html
<form action="/login" method="post">
    <input type="text" name="usuario" placeholder="Usuário">
    <input type="password" name="senha" placeholder="Senha">
    <button type="submit">Entrar</button>
</form>
```

- Os dados não são visíveis na URL e podem ser muito maiores do que os enviados via GET.

Atributo enctype

O atributo enctype define como os dados do formulário serão codificados ao serem enviados ao servidor.

Existem três valores principais:
application/x-www-form-urlencoded (padrão):
Os dados são codificados como pares chave=valor e são enviados no corpo da requisição.

```
1  <form action="/salvar" method="post" enctype="application/x-www-form-urlencoded">
2      <input type="text" name="nome" placeholder="Seu Nome">
3      <button type="submit">Salvar</button>
4    </form>
```

Atributo enctype

multipart/form-data:

- Usado quando o formulário contém arquivos para upload. Sem este valor, arquivos não podem ser enviados corretamente.

```
1  <form action="/upload" method="post" enctype="multipart/form-data">
2      <input type="file" name="arquivo">
3      <button type="submit">Enviar Arquivo</button>
4    </form>
```

Atributo enctype

text/plain:

- Os dados são enviados como texto puro, com muito pouca codificação.

```html
<form action="/enviar" method="post" enctype="text/plain">
    <input type="text" name="mensagem" placeholder="Digite sua mensagem">
    <button type="submit">Enviar Mensagem</button>
</form>
```

Validação de Formulário

A validação de formulários HTML é uma forma de garantir que os dados fornecidos pelo usuário estejam corretos antes do envio ao servidor. HTML5 introduziu novos atributos que tornam a validação de formulários mais simples e poderosa, sem a necessidade de usar JavaScript para tarefas básicas.

Atributos de Validação

required

- O atributo required garante que o campo não seja deixado em branco. Se o usuário tentar enviar o formulário sem preencher o campo, o navegador exibirá uma mensagem de erro automática.

Atributos de Validação

```html
<form>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" required>
    <button type="submit">Enviar</button>
  </form>
```

Se o usuário não preencher o campo "Nome", o formulário não será enviado, e uma mensagem será exibida informando que o campo é obrigatório.

Atributos de Validação

min, max, minlength, maxlength:

Esses atributos permitem controlar o valor mínimo/máximo numérico ou o número de caracteres que o usuário pode inserir.

- min e max: Controlam o valor numérico mínimo e máximo que um campo de número pode receber.

```
<label for="idade">Idade (entre 18 e 99):</label>
<input type="number" id="idade" name="idade" min="18" max="99" required>
```

Atributos de Validação

- Se o valor digitado estiver fora do intervalo, o formulário mostrará uma mensagem de erro automática.

minlength e maxlength: Definem o número mínimo e máximo de caracteres que podem ser inseridos em campos de texto.

```
1  <label for="senha">Senha (mínimo 8 caracteres):</label>
2  <input type="password" id="senha" name="senha" minlength="8" required>
```

Exercício 1: Formulário de Cadastro de Idade e Senha

Crie um formulário HTML que tenha dois campos: um para a idade e outro para a senha. A idade deve ser validada de forma que o usuário possa inserir um valor entre 18 e 65 anos. A senha deve ter pelo menos 8 caracteres e no máximo 20 caracteres.
Instruções:

- O campo de idade deve usar os atributos min e max para limitar o valor entre 18 e 65.
- O campo de senha deve usar minlength e maxlength para garantir que a senha tenha pelo menos 8 caracteres e no máximo 20.
- Ao submeter o formulário, se as restrições não forem atendidas, o navegador deve exibir o feedback automático de erro.

Atributos de Validação

pattern

O atributo pattern permite usar Regex (Expressões Regulares) para validar o formato de entrada. Se o dado inserido não corresponder ao padrão definido, o formulário não será enviado e o navegador exibirá uma mensagem de erro.

```
<label for="email">Email:</label>
<input type="email" id="email" name="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" required>
```

Atributos de Validação

- Esse Regex define um formato básico para e-mails (com um @ e domínio).

Expressões Regulares (Regex)

Regex é uma sequência de caracteres que define um padrão de busca, sendo muito útil para validação de formulários.

Expressões Regulares (Regex)

Exemplo de Regex para CPF (formato: 000.000.000-00):

```html
<label for="cpf">CPF:</label>
<input type="text" id="cpf" name="cpf" pattern="\d{3}\.\d{3}\.\d{3}-\d{2}" title="Digite no formato 000.000.000-00" required>
```

- \d{3}: Espera exatamente 3 dígitos.
- \.: O ponto literal no CPF.
- -: O hífen literal no CPF.

Se o usuário inserir um CPF no formato errado, ele verá uma mensagem de erro automática informando que o formato não é válido.

Feedback de Erro Automático

O HTML5 fornece feedback de erro automaticamente para campos que não atendem aos requisitos. O navegador exibe mensagens padrão que podem ser personalizadas.

Exemplo de personalização da mensagem de erro (title):

```
1  <label for="telefone">Telefone (Formato: 123-4567):</label>
2  <input type="text" id="telefone" name="telefone" pattern="\d{3}-\d{4}" title="Por favor, insira no formato 123-4567" required>
```

- Se o padrão não for atendido, a mensagem personalizada será exibida.

Técnicas Avançadas com Formulários

Agora vamos mergulhar em técnicas avançadas para trabalhar com formulários em HTML, explorando não apenas os atributos básicos que já discutimos (como required, min, max, pattern e enctype), mas também outros atributos poderosos que podem ser utilizados para melhorar a experiência do usuário e a flexibilidade dos formulários.

autofocus

- O atributo autofocus faz com que o campo receba automaticamente o foco quando a página é carregada, facilitando o início da interação com o formulário.

Técnicas Avançadas com Formulários

```html
<form>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" autofocus required>
  </form>
```

autocomplete

- O atributo autocomplete sugere valores ao usuário com base em entradas anteriores ou padrões definidos pelo navegador.
- Ele pode ser ativado (on) ou desativado (off).

Técnicas Avançadas com Formulários

autocomplete

- O atributo autocomplete sugere valores ao usuário com base em entradas anteriores ou padrões definidos pelo navegador.
- Ele pode ser ativado (on) ou desativado (off).

```
1  <form autocomplete="on">
2      <label for="email">Email:</label>
3      <input type="email" id="email" name="email" required>
4    </form>
```

Técnicas Avançadas com Formulários

novalidate

- O atributo novalidate desativa a validação automática dos campos no formulário, permitindo que o envio ocorra sem verificação de regras impostas pelos atributos (required, pattern, etc.). Pode ser útil para formulários controlados por JavaScript.

```
1  <form novalidate>
2      <label for="email">Email:</label>
3      <input type="email" id="email" name="email">
4      <button type="submit">Enviar</button>
5    </form>
```

Técnicas Avançadas com Formulários

formnovalidate

- Usado em botões de submissão (<button> ou <input type="submit">), o formnovalidate ignora a validação ao clicar no botão. É útil quando se tem botões como "Salvar rascunho" que não precisam de validação.

```
<form>
    <input type="email" name="email" required>
    <button type="submit" formnovalidate>Salvar Rascunho</button>
  </form>
```

Técnicas Avançadas com Formulários

formenctype

- O formenctype permite sobrescrever o atributo enctype padrão do formulário. Pode ser útil ao enviar diferentes tipos de dados com diferentes botões de submissão.

```
<form action="/upload" method="post">
    <input type="file" name="arquivo">
    <button type="submit" formenctype="multipart/form-data">Enviar Arquivo</button>
  </form>
```

Técnicas Avançadas com Formulários

formaction

- O formaction permite especificar uma URL de envio diferente para um botão de envio específico, ignorando o valor action do formulário principal.

```
<form action="/padrao">
    <button type="submit" formaction="/salvar">Salvar</button>
    <button type="submit" formaction="/enviar">Enviar</button>
  </form>
```

Técnicas Avançadas com Formulários

readonly

- O atributo readonly permite que um campo seja exibido e seu valor seja enviado no formulário, mas o usuário não pode modificar seu conteúdo.

```
1  <label for="id">ID do Usuário:</label>
2  <input type="text" id="id" name="id" value="12345" readonly>
```

Técnicas Avançadas com Formulários

disabled

- O disabled desabilita completamente o campo, tornando-o não modificável e excluído do envio do formulário.

```
1   <label for="codigo">Código de Promoção:</label>
2   <input type="text" id="codigo" name="codigo" disabled>
```

Técnicas Avançadas com Formulários

multiple

- Usado em campos como <input type="file"> ou <select>, o atributo multiple permite selecionar vários arquivos ou múltiplas opções.

```
1  <label for="arquivos">Escolha arquivos:</label>
2  <input type="file" id="arquivos" name="arquivos" multiple>
```

Técnicas Avançadas com Formulários

step

- O atributo step é usado com campos numéricos (number, date, time, etc.) para definir intervalos de valores válidos. É útil para definir incrementos personalizados (ex: de 5 em 5, de 0.5 em 0.5).

```
1  <label for="quantidade">Quantidade (passo 5):</label>
2  <input type="number" id="quantidade" name="quantidade" min="0" max="100" step="5">
```

Técnicas Avançadas com Formulários

Resumo

- autofocus: Foco automático no carregamento.
- autocomplete: Sugerir valores com base em históricos.
- novalidate e formnovalidate: Controlam a necessidade de validação.
- formenctype e formaction: Modificam comportamento de envio para botões individuais.
- readonly e disabled: Controlam se o campo pode ser modificado ou enviado.
- multiple: Permite múltiplas seleções de arquivos ou opções.
- step: Define incrementos em valores numéricos.

Regex e Validação Avançada em Formulários HTML

Regex (expressões regulares) são padrões usados para combinar sequências de caracteres em strings. Em formulários HTML, o uso de Regex pode proporcionar uma validação muito mais poderosa e flexível, permitindo que você defina padrões específicos que os campos devem seguir, além de trabalhar com atributos como pattern para impor esses padrões diretamente no HTML.

Vamos explorar a aplicação de Regex em HTML para validação de formulários, assim como técnicas de validação avançada usando JavaScript para cenários mais complexos.

Regex e Validação Avançada em Formulários HTML

O Atributo pattern

O atributo pattern permite que você aplique uma expressão regular (Regex) diretamente em um campo de entrada (<input>). Isso é útil para validar padrões específicos, como números de telefone, códigos postais ou até mesmo formatos de senhas.

Exemplo 1: Validando um número de telefone

Neste exemplo, usamos Regex para garantir que o usuário insira o número de telefone no formato (XX) XXXXX-XXXX:

Regex e Validação Avançada em Formulários HTML

```html
<form>
    <label for="telefone">Telefone:</label>
    <input type="text" id="telefone" name="telefone" pattern="\(\d{2}\) \d{5}-\d{4}" title="Formato: (XX) XXXXX-XXXX" required>
    <button type="submit">Enviar</button>
  </form>
```

Explicação da Regex:

- \(\d{2}\): A Regex começa com (e), que são caracteres literais. Dentro dos parênteses, temos \d{2}, que significa dois dígitos (números de 0 a 9).
- \d{5}: Cinco dígitos que representam a parte principal do número.
- -: Um hífen literal.
- \d{4}: Quatro dígitos para completar o número.

Regex e Validação Avançada em Formulários HTML

- Se o usuário digitar um número fora desse formato, o envio do formulário falhará, e a mensagem personalizada no atributo title será exibida como feedback de erro.

Validando Senhas com Regras Complexas

Senhas geralmente precisam seguir regras de segurança específicas, como conter letras maiúsculas, minúsculas, números e símbolos especiais. Podemos criar uma Regex para forçar essas regras.

Validando Senhas com Regras Complexas

Exemplo 2: Validação de senha
Neste exemplo, a senha deve ter pelo menos 8 caracteres, com pelo menos uma letra maiúscula, uma letra minúscula, um número e um caractere especial.

```
<form>
    <label for="senha">Senha:</label>
    <input type="password" id="senha" name="senha" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[\W_]).{8,}" title="A
     senha deve conter pelo menos 8 caracteres, incluindo uma
     letra maiúscula, uma letra minúscula, um número e um
     caractere especial." required>

    <button type="submit">Enviar</button>
</form>
```

Validando Senhas com Regras Complexas

Explicação da Regex:

- (?=.*\d): Pelo menos um número.
- (?=.*[a-z]): Pelo menos uma letra minúscula.
- (?=.*[A-Z]): Pelo menos uma letra maiúscula.
- (?=.*[\W_]): Pelo menos um caractere especial (qualquer coisa que não seja uma letra ou número).
- .{8,}: A senha deve ter no mínimo 8 caracteres.

Validando Senhas com Regras Complexas

Se a senha não atender a esses critérios, o formulário exibirá a mensagem de erro definida no atributo title.

Validação de E-mails Personalizados

O campo de e-mail (<input type="email">) já possui uma validação embutida pelo navegador. No entanto, em alguns casos, pode ser necessário adicionar mais restrições ao formato do e-mail, como validar domínios específicos (ex.: apenas emails corporativos).

Validação de E-mails Personalizados

Exemplo 3: Validação de e-mail corporativo:

Aqui, validamos que o e-mail fornecido deve pertencer ao domínio empresa.com.

```
<form>
    <label for="email">E-mail Corporativo:</label>
    <input type="email" id="email" name="email" pattern="^
    [a-zA-Z0-9._%+-]+@empresa\.com$" title="O e-mail deve ser
     do domínio @empresa.com" required>

    <button type="submit">Enviar</button>
  </form>
```

Validação de E-mails Personalizados

Explicação da Regex:

- ^[a-zA-Z0-9._%+-]+: O início da string deve conter letras, números ou caracteres permitidos antes do @.
- @empresa\.com$: A string deve terminar com @empresa.com.

Usando Regex com JavaScript para Validação Customizada

Embora o HTML5 ofereça validação nativa com o atributo pattern, em algumas situações mais avançadas, pode ser necessário validar dados dinamicamente, e é aí que o JavaScript entra em cena. Com JavaScript, você pode aplicar expressões regulares para validar entradas em tempo real e fornecer feedback instantâneo ao usuário.

Usando Regex com JavaScript para Validação Customizada

Exemplo 4: Validação de CPF com JavaScript

Aqui, vamos usar Regex com JavaScript para validar o CPF (formato 000.000.000-00), e fornecer um feedback personalizado.

```
<form id="cpf-form">
    <label for="cpf">CPF:</label>
    <input type="text" id="cpf" name="cpf" required>
    <span id="cpf-error" style="color:red; display:none;">Formato de CPF inválido!</span>
    <button type="submit">Enviar</button>
  </form>

  <script>
    const cpfInput = document.getElementById('cpf');
    const cpfForm = document.getElementById('cpf-form');
    const cpfError = document.getElementById('cpf-error');

    cpfForm.addEventListener('submit', function(event) {
      const cpfRegex = /^\d{3}\.\d{3}\.\d{3}-\d{2}$/;
      if (!cpfRegex.test(cpfInput.value)) {
        event.preventDefault(); // Impede o envio do formulário
        cpfError.style.display = 'inline'; // Exibe a mensagem de erro
      } else {
        cpfError.style.display = 'none';
      }
    });
  </script>
```

Usando Regex com JavaScript para Validação Customizada

Explicação:

- \d{3}\.\d{3}\.\d{3}-\d{2}: Três grupos de três dígitos, separados por pontos, seguidos de um grupo de dois dígitos após um hífen.
- Se o CPF não seguir esse padrão, a mensagem de erro será exibida, e o envio do formulário será impedido.

Exercício 1: Formulário de Registro de Usuário

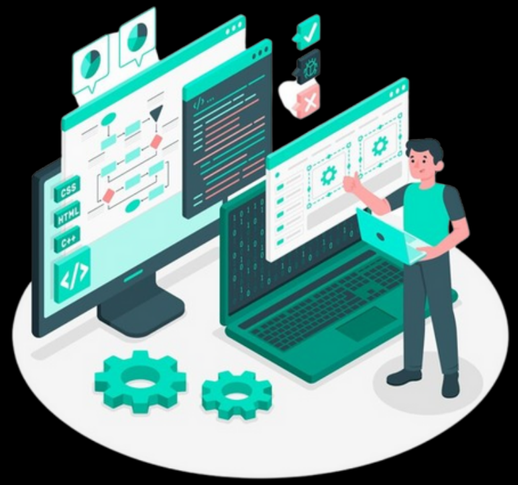
Crie um formulário de registro que contenha os seguintes campos e regras de validação:

1. Nome Completo (obrigatório, com no mínimo 5 caracteres)
2. Idade (obrigatório, entre 18 e 99 anos)
3. E-mail (obrigatório, formato de e-mail válido)
4. Senha (obrigatória, com no mínimo 8 caracteres)
5. CPF (obrigatório, formato 000.000.000-00)

Exercício 1: Formulário de Registro de Usuário

Regras:

- Utilize os atributos HTML required, minlength, min, max e pattern para validar os campos diretamente no HTML.
- Adicione uma mensagem de erro personalizada para o campo CPF, caso o formato inserido esteja incorreto.



Exercício 2: Formulário de Agendamento de Consulta Médica

Crie um formulário para agendamento de consultas médicas com as seguintes regras:

1. Nome do Paciente (obrigatório)
2. Data de Nascimento (obrigatório, use o campo date, idade mínima de 18 anos)
3. Telefone (obrigatório, formato: (XX) XXXXX-XXXX)
4. E-mail (opcional, mas se preenchido, deve seguir o formato de e-mail)
5. Data da Consulta (obrigatório, a data da consulta deve ser a partir do dia seguinte da data atual)

Exercício 2: Formulário de Agendamento de Consulta Médica

Regras:

- Utilize os atributos HTML required, min, max e pattern para validar os campos.
- No campo telefone, use Regex para garantir o formato correto.
- A data da consulta deve ser validada para ser uma data futura (não pode ser no passado).