Formulários HTML

1. Fundamentos de formulários

Os formulários são uma das partes mais importantes do seu site. Eles são a porta de entrada do seu usuário para o seu back-end – o usuário fornece dados em um formulário e você faz coisas com eles.

Você precisa especificar os tipos adequados de entradas para cada item de dados possível, pois geralmente há várias maneiras de coletar um dado, mas apenas uma maneira é mais fácil para o usuário.

Nesta lição, exploraremos os fundamentos dos formulários HTML e alguns dos diferentes tipos de entradas disponíveis para você.

Resultados de Aprendizagem

Ao final desta lição, você deverá ser capaz de:

- Criar formulários com HTML
- Ter uma ideia básica de como estilizar formulários

O elemento do formulário

O elemento de formulário é um elemento de contêiner como o elemento div que aprendemos anteriormente no currículo. O elemento de formulário envolve todas as entradas com as quais um usuário irá interagir em um formulário.

O elemento form aceita dois atributos essenciais; o primeiro é o atributo `action` que recebe um valor de URL que informa ao formulário para onde deve enviar seus dados para serem processados. Mais adiante no currículo, aprenderemos a conectar sistemas de back-end a formulários de front-end

usando esse atributo. Por enquanto, é apenas essencial saber para que o `action` atributo é usado.

O segundo é o atributo `method` que informa ao navegador qual método de solicitação HTTP ele deve usar para enviar o formulário. Os métodos de solicitação GET e POST são os dois que você mais usará.

Usamos GET quando queremos recuperar algo de um servidor. Por exemplo, o Google usa uma solicitação GET quando você pesquisa à medida que *obtém* os resultados da pesquisa.

POST é usado quando queremos alterar algo no servidor, por exemplo, quando um usuário faz uma conta ou faz um pagamento em um site.

A marcação para criar um elemento de formulário é assim:

```
<form action="example.com/path" method="post">

// os campos do formulário vão aqui

</form>
```

Controles de formulário

Para começar a coletar dados do usuário, precisamos usar elementos de controle de formulário. Esses são todos os elementos com os quais os usuários irão interagir no formulário, como caixas de texto, menus suspensos, caixas de seleção e botões. Nas próximas seções, exploraremos alguns dos controles de formulário que você usará com mais frequência.

O elemento de entrada - *input*

O elemento input é o mais versátil de todos os elementos de controle de formulário. Ele aceita um atributo `type` que informa ao navegador que *tipo* de dados ele deve esperar e como ele deve renderizar o elemento de entrada.

Uma entrada de texto se parece com isso:

```
<form action="example.com/path" method="post">

    <input type="text">

</form>
```

As entradas de texto aceitam qualquer entrada de texto. Por exemplo, você o usaria para coletar coisas como nome e sobrenome dos usuários.

Rótulos - *label*

Uma entrada por si só não é muito útil, pois o usuário não saberá que tipo de dados deve fornecer. Em vez disso, podemos atribuir um rótulo às nossas entradas para informar aos usuários que tipo de dados eles devem inserir.

Para criar um rótulo, usamos o elemento `<label>`. O texto que queremos exibir no rótulo ficará entre suas tags de abertura e fechamento:

```
<form action="example.com/path" method="post">

    <label for="first_name">First Name:</label>
    <input type="text" id="first_name">

</form>
```

Os rótulos aceitam um atributo `for`, que o associa a uma entrada específica. A entrada que queremos associar a um rótulo precisa de um atributo `id` com o mesmo valor do atributo do rótulo for.

O atributo id #

- O atributo id atribui um identificador ao elemento <input>.
- O id permite que o JavaScript acesse facilmente o elemento <input>.
- Também é usado para apontar para um seletor de id específico em uma folha de estilo.

Quando um rótulo é associado a uma entrada e clicado, ele focalizará o cursor nessa entrada, pronto para o usuário inserir alguns dados. Isso ajuda a tornar nossos formulários mais acessíveis aos usuários que dependem de tecnologias assistivas.

Atributo de espaço reservado - placeholder

Para orientar os usuários sobre o que inserir nos elementos do formulário, podemos incluir texto de espaço reservado nos campos de entrada.

Isso é feito adicionando um atributo `placeholder` a uma entrada. O valor será o texto do *espaço reservado* que queremos exibir na entrada:

```
<label for="first_name">First Name:</label>
<input type="text" id="first_name" placeholder="Bob...">
```

Use o texto de espaço reservado para demonstrar como o texto deve ser inserido e formatado.

O atributo de nome - name

Precisamos usar rótulos para que os usuários entendam o que os dados inseridos em um campo de entrada irão representar. Assim, também

precisamos deixar o **backend**, para onde enviamos nossos dados, saber o que cada dado representa.

Fazemos isso adicionando um atributo `name` às nossas entradas:

```
<label for="first_name">First Name:</label>
<input type="text" id="first_name" name="first_name">
```

O atributo `name` serve como referência aos dados inseridos em um controle de formulário após enviá-lo. Você pode pensar nisso como **um nome de variável para a entrada**. A entrada de formulário deve sempre ter um atributo `name`; caso contrário, será ignorado quando o formulário for enviado.

Para entender melhor como isso se parece, podemos enviar um formulário para [httpbin](#) . Este serviço enviará de volta uma resposta que nos permitirá visualizar quais dados foram enviados. Preencha o formulário abaixo e clique em enviar:

```
<form action="http://httpbin.org/post" method="post">
  <div>
    <label for="first_name">First Name:</label>
    <input type="text" name="first_name" id="first_name">
  </div>

  <div>
    <label for="last_name">Last Name:</label>
    <input type="text" name="last_name" id="last_name">
  </div>

  <div>
    <label for="age">Age:</label>
    <input type="number" name="age" id="age">
  </div>
```

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

A saída que nos interessa da resposta é o objeto “form”. Deve ser algo assim:

```
"form": { "age": "33", "first_name": "John", "last_name":
"Doe" },
```

Tente alterar os atributos `name` de alguns dos campos de entrada no formulário e remover o atributo completamente e, em seguida, envie o formulário novamente para ver como os dados do formulário na resposta são alterados.

Usando controles de formulário fora dos formulários

Vale a pena mencionar que você pode usar qualquer um dos controles de formulário que o HTML fornece fora do `<form>`elemento, mesmo quando você não tem um servidor de backend para onde você pode enviar dados.

Por exemplo, você pode querer ter uma entrada que obtenha alguns dados de um usuário e os exiba em outro lugar da página com JavaScript:

Precisaremos manipular bastante os dados de controles de formulário como este quando chegarmos aos projetos do curso JavaScript Fullstack.

O atributo de tipo

`Inputs de email` são entradas de texto especializadas apenas para endereços de e-mail. Eles são diferentes das entradas de texto, pois exibirão um teclado diferente que incluirá o símbolo @ em dispositivos móveis para facilitar a inserção de endereços de e-mail.

Eles também validam que o usuário inseriu um endereço de e-mail formatado corretamente, mas haverá mais validações posteriormente.

Para criar uma entrada de email, usamos um elemento de entrada com `type` atributo de “email”:

```
<label for="user_email">Email Address:</label>
<input type="email" id="user_email" name="email"
placeholder="you@example.com">
```

`Password inputs` são outra entrada de texto especializada. Eles diferem das entradas de texto regulares, pois mascaram os dados inseridos com asteriscos(*) para impedir que qualquer pessoa veja o que foi inserido.

Uma entrada de senha pode ser criada usando um elemento de entrada com um tipo de “senha”:

```
<label for="user_password">Password:</label>
<input type="password" id="user_password" name="password">
```

O **`input de number`** único aceita valores numéricos e ignora quaisquer outros caracteres que o usuário tente inserir.

Criamos uma entrada numérica usando o elemento input com um atributo **`type`** de “number”:

```
<label for="amount">Amount:</label>
<input type="number" id="amount" name="amount">
```

Para coletar datas de um usuário, podemos usar um arquivo `date input`. Essa entrada é exclusiva porque fornece uma melhor experiência do usuário para escolher as datas renderizando um calendário simples do selecionador de datas.

Para criar uma entrada de data, usamos o elemento de entrada com um atributo `type` de “data”:

```
<label for="dob">Date of Birth:</label>
<input type="date" id="dob" name="dob">
```

Área de texto

Embora tecnicamente não seja um elemento de entrada, o elemento de área de texto fornece uma caixa de entrada que pode aceitar texto que abrange várias linhas, como comentários e revisões do usuário. Ele também pode ser redimensionado clicando e arrastando o canto inferior direito para torná-lo maior ou menor.

Para criar uma área de texto, usamos o elemento `<textarea>`:

```
<textarea></textarea>
```

Ao contrário dos elementos input, os elementos Textarea possuem uma tag de fechamento. Isso permite que você envolva algum conteúdo inicial que deseja que a área de texto exiba:

```
<textarea>Algum conteúdo inicial</textarea>
```

Os elementos da área de texto aceitam alguns atributos exclusivos que outros controles de formulário não aceitam. Estes são os atributos `rows` e `cols`. Eles permitem que você controle a altura inicial (linhas) e a largura (colunas) da área de texto:

```
<textarea rows="20" cols="60"></textarea>
```

Elementos de Seleção

Às vezes, você desejará que os usuários selecionem um valor de uma lista predefinida. É aqui que os elementos selecionados serão úteis.

Selecionar menu suspenso

O elemento select renderiza uma lista suspensa onde os usuários podem selecionar uma opção. Sintaticamente, os elementos select têm marcação semelhante a listas não ordenadas. O elemento select envolve elementos de opção que são as opções que podem ser selecionadas.

Para criar uma lista suspensa de seleção, usamos o `<select>`elemento. Quaisquer opções que queremos exibir dentro do elemento select são definidas usando `<option>`elementos:

```
<select name="Car">
    <option value="mercedes">Mercedes</option>
    <option value="tesla">Tesla</option>
    <option value="volvo">Volvo</option>
    <option value="bmw">BMW</option>
    <option value="mini">Mini</option>
    <option value="ford">Ford</option>
</select>
```

Todos os elementos de opção precisam ter um atributo `value`. Este valor será enviado ao servidor quando o formulário for enviado.

Podemos definir uma das opções como o elemento selecionado padrão quando o navegador renderiza o formulário pela primeira vez, dando a uma das opções o atributo `selected`:

```
<select name="Car">
    <option value="mercedes">Mercedes</option>
    <option value="tesla">Tesla</option>
    <option value="volvo" selected>Volvo</option>
    <option value="bmw">BMW</option>
    <option value="mini">Mini</option>
    <option value="ford">Ford</option>
</select>
```

Também podemos dividir a lista de opções em grupos usando o elemento `<optgroup>`. O elemento optgroup recebe um atributo `label` que o navegador usa como rótulo para cada grupo:

```
<select name="fashion">
    <optgroup label="Clothing">
        <option value="t_shirt">T-Shirts</option>
        <option value="sweater">Sweaters</option>
        <option value="coats">Coats</option>
    </optgroup>
    <optgroup label="Foot Wear">
        <option value="sneakers">Sneakers</option>
        <option value="boots">Boots</option>
        <option value="sandals">Sandals</option>
    </optgroup>
</select>
```

Botões do rádio

As listas suspensas de seleção são ótimas para economizar espaço na página quando temos uma extensa lista de opções que queremos que os usuários escolham. No entanto, quando temos uma lista menor de 5 ou menos opções para escolher, geralmente é uma melhor experiência do usuário exibi-las na página em vez de ocultar atrás de um menu suspenso.

Nesse caso, podemos usar botões de opção. Os botões de opção nos permitem criar várias opções que o usuário pode escolher. Para criar botões de rádio, usamos o elemento input sempre adaptável novamente com um `type`atributo de “radio”:

```
<h1>Ticket Type</h1>
<div>
    <input type="radio" id="child" name="ticket_type"
value="child">
    <label for="child">Child</label>
</div>
<div>
    <input type="radio" id="adult" name="ticket_type"
value="adult">
    <label for="adult">Adult</label>
</div>
<div>
    <input type="radio" id="senior" name="ticket_type"
value="senior">
    <label for="senior">Senior</label>
</div>
```

Quando selecionamos um dos botões de opção e, em seguida, selecionamos outro, ele desmarcará o primeiro. Os botões de opção sabem fazer isso porque têm o mesmo atributo `name`. É assim que o navegador sabe que esses elementos fazem parte do mesmo grupo de opções.

Podemos definir o botão de opção selecionado padrão adicionando o atributo `checked` a ele:

```
<h1>Ticket Type</h1>
<div>
    <input type="radio" id="child" name="ticket_type"
value="child">
    <label for="child">Child</label>
</div>
<div>
    <input type="radio" id="adult" name="ticket_type"
value="adult" checked>
    <label for="adult">Adult</label>
</div>
<div>
    <input type="radio" id="senior" name="ticket_type"
value="senior">
    <label for="senior">Senior</label>
</div>
```

```
  <form action="www.siteexemple.com/xxxx" method="post">

    <h1>Tipo Tiquete</h1>
    <div>
      <input type="radio" id="crianca" name="ticket_type"
value="crianca">
      <label for="crianca">Criança</label>
    </div>

    <div>
      <input type="radio" id="adult" name="ticket_type"
value="adult">
      <label for="adult">Adulto</label>
```

```
    </div>

    <div>
      <input type="radio" id="senior" name="ticket_type"
value="senior">
      <label for="senior">Senior</label>
    </div>

  </form>
```

Caixas de seleção

As caixas de seleção são semelhantes aos botões de opção, pois permitem que os usuários escolham entre um conjunto de opções predefinidas. Mas, ao contrário dos botões de opção, eles permitem que várias opções sejam selecionadas de uma só vez.

Para criar um checkbox, usamos o elemento input com um atributo `type` de “checkbox”:

```
<h1>Pizza Toppings</h1>
<div>
    <input type="checkbox" id="sausage" name="topping"
value="sausage">
    <label for="sausage">Sausage</label>
</div>
<div>
    <input type="checkbox" id="onions" name="topping"
value="onions">
    <label for="onions">Onions</label>
</div>
<div>
    <input type="checkbox" id="pepperoni" name="topping"
value="pepperoni">
```

```
    <label for="pepperoni">Pepperoni</label>
</div>
<div>
    <input type="checkbox" id="mushrooms" name="topping"
value="mushrooms">
    <label for="mushrooms">Mushrooms</label>
</div>
```

Também podemos ter uma única caixa de seleção quando queremos que os usuários alternem se desejam que algo seja verdadeiro ou falso. Como se inscrever em um boletim informativo quando eles criam uma conta, por exemplo:

```
<div>
    <input type="checkbox" id="newsletter" name="news_letter">
    <label for="newsletter">Send me the news letter</label>
</div>
```

Podemos definir caixas de seleção para serem marcadas por padrão no carregamento da página, dando a elas um atributo `checked`:

```
<div>
    <input type="checkbox" id="newsletter" name="news_letter"
checked>
    <label for="newsletter">Send me the news letter</label>
</div>
```

```
  <form action="www.siteexemple.com/xxxx" method="post">
```

```
    <h1>Pizza Toppings</h1>

    <div>
      <input type="checkbox" id="sausage" name="topping"
value="sausage">
      <label for="sausage">Sausage</label>
    </div>

    <div>
      <input type="checkbox" id="onions" name="topping"
value="onions">
      <label for="onions">Onions</label>
    </div>

    <div>
      <input type="checkbox" id="pepperoni" name="topping"
value="pepperoni">
      <label for="pepperoni">Pepperoni</label>
    </div>

    <div>
      <input type="checkbox" id="mushrooms" name="topping"
value="mushrooms">
      <label for="mushrooms">Mushrooms</label>
    </div>

  </form>
```

Botões

O elemento botão cria botões clicáveis com os quais o usuário pode interagir para enviar formulários e acionar outras ações.

Para criar um botão, usamos o elemento `<button>` . O conteúdo ou texto que queremos exibir dentro do botão ficará dentro das tags de abertura e fechamento:

```
<button>Clique Me</button>
```

O elemento button também aceita um atributo `type` que informa ao navegador com qual tipo de botão ele está lidando. Existem três tipos de botões disponíveis para nós.

Botões de envio

Quando um usuário terminar de preencher um formulário, ele precisará de uma maneira de enviá-lo. Existe um botão especializado para isso; o botão enviar. Quando um botão enviar é clicado, ele enviará o formulário em que está contido. O atributo **`type` tem um valor de submit por padrão**, ou seja, se o `type` não for especificado ou o valor fornecido for inválido.

Para criar um botão de envio, usamos o elemento de botão com um atributo `type` de “enviar”:

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

Botão de reset

Um botão de reinicialização limpa todos os dados que o usuário inseriu no formulário e define todas as entradas no formulário de volta ao que eram inicialmente.

Para criar um botão de reset, usamos o elemento button com um atributo `type` de “reset”:

```
<button type="reset">Reinicializar</button>
```

Botão genérico

O terceiro e último tipo de botão é simplesmente um botão genérico que pode ser usado para qualquer coisa. É comumente usado com Javascript para criar interfaces de usuário interativas.

Para criar um botão genérico, usamos o elemento button com um atributo `type` de “button”:

```
<button type="button">Clique para alternar</button>
```

Nota : É importante lembrar que um botão dentro de um formulário com o valor `type` de submit (que por acaso é o valor padrão) sempre tentará fazer uma nova solicitação e enviar os dados de volta ao servidor. Portanto, para botões usados em um formulário para finalidades diferentes do envio de dados, o atributo `type` deve sempre ser especificado para evitar efeitos indesejados no envio de um formulário.

Organizando elementos de formulário

Usar as entradas corretas para os dados que queremos que os usuários insiram ajuda bastante a tornar nossos formulários amigáveis. No entanto, em formulários maiores, os usuários podem facilmente ficar sobrecarregados e desencorajados se houver muitas entradas para preencher.

Felizmente, o HTML fornece alguns elementos que facilitam a organização de formulários em seções que são visualmente distintas e gerenciáveis para digerir.

Elemento do conjunto de campos

O elemento fieldset é um elemento container que nos permite agrupar entradas de formulário relacionadas em uma unidade lógica.

Para criar um fieldset, usamos o elemento `<fieldset>`. Quaisquer entradas de formulário que queremos agrupar vão dentro das tags de fieldset de abertura e fechamento:

```
<fieldset>
    <label for="first_name">First Name</label>
    <input type="text" id="first_name" name="first_name">

    <label for="last_name">Last Name</label>
    <input type="text" id="last_name" name="last_name">
</fieldset>
```

Legenda - legend

O elemento de legenda é usado para fornecer um cabeçalho ou legenda aos conjuntos de campos para que o usuário possa ver para que serve um agrupamento de entradas.

Para criar uma legenda, usamos o elemento `<legend>` com o texto que queremos exibir dentro de suas tags de abertura e fechamento. Uma legenda deve sempre vir logo após uma tag de fieldset de abertura:

```
<fieldset>
    <legend>Contact Details</legend>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <label for="phone_number">Phone Number:</label>
    <input type="tel" id="phone_number" name="phone_number">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email">
```

```
</fieldset>
<fieldset>
    <legend>Delivery Details</legend>
    <label for="street_address">Street Address:</label>
    <input type="text" id="street_address"
name="street_address">
    <label for="city">City:</label>
    <input type="text" id="city" name="city">
    <label for="zip_code">Zip Code:</label>
    <input type="text" id="zip_code" name="zip_code">
</fieldset>
```

Um caso de uso comum para esses elementos é usar um conjunto de campos para agrupar botões de opção e usar uma legenda para comunicar ao usuário para que serve cada uma das opções:

```
<fieldset>
    <legend>What would you like to drink?</legend>
    <div>
        <input type="radio" name="drink" id="coffee"
value="coffee">
        <label for="coffee">Coffee</label>
    </div>
    <div>
        <input type="radio" name="drink" id="tea" value="tea">
        <label for="tea">Tea</label>
    </div>
    <div>
        <input type="radio" name="drink" id="soda"
value="soda">
        <label for="soda">Soda</label>
    </div>
</fieldset>
```

Uma nota sobre formulários de estilo

Forneceremos recursos que aprofundam os formulários de estilo na seção de atribuição que vem a seguir. No entanto, antes de chegarmos à tarefa, devemos falar sobre alguns dos desafios com o estilo de formulários HTML e como podemos contorná-los:

Estilos de navegador padrão

Cada navegador tem seus próprios estilos padrão para controles de formulário, tornando seus formulários visualmente diferentes para os usuários, dependendo do navegador que estão usando.

Para ter um design consistente entre todos os navegadores, temos que substituir esses estilos padrão e estilizá-los nós mesmos.

Complicado e absolutamente impossível de estilizar controles de formulário

Os controles de formulário baseados em texto, como texto, e-mail, senha e áreas de texto, são razoavelmente simples de estilizar. Eles operam como qualquer outro elemento HTML, e a maioria das propriedades CSS podem ser usadas neles.

As coisas ficam mais complicadas ao criar estilos personalizados para botões de opção e caixas de seleção. Mas existem muitos guias por aí que você pode usar para obter o design desejado. Também foram disponibilizadas novas propriedades CSS nos últimos tempos para tornar os botões de opção e as caixas de seleção muito mais fáceis.

Certos aspectos de outros elementos são absolutamente impossíveis de estilizar, por exemplo, calendário ou selecionadores de data. Se quisermos estilos personalizados para eles, teremos que construir controles de formulário

personalizados com JavaScript ou usar uma das muitas bibliotecas JavaScript que nos fornecem soluções prontas.

2. **Validação de formulários**

As validações nos permitem definir restrições ou regras específicas que determinam quais dados os usuários podem inserir em uma entrada. Quando um usuário insere dados que violam as regras, uma mensagem aparecerá, fornecendo feedback sobre o que havia de errado com os dados inseridos e como corrigi-lo.

As validações são um ingrediente vital em formulários bem projetados. Eles ajudam a proteger nossos sistemas de back-end contra o recebimento de dados incorretos e ajudam a tornar a experiência de interação com nosso formulário o mais simples possível para nossos usuários.

Esta lição explorará algumas das validações internas que você pode usar com formulários HTML. Também vamos mergulhar nas validações de estilo com CSS.

Resultados de Aprendizagem

Ao final desta lição, você deverá ser capaz de:

- Explique o que são validações de formulário
- Saiba como usar algumas das validações HTML internas básicas
- Saiba como criar validações personalizadas

Validação necessária

Muitas vezes, queremos garantir que campos específicos tenham sido preenchidos antes de enviar o formulário, por exemplo, o e-mail e a senha em um formulário de login.

Para tornar um campo obrigatório, simplesmente adicionamos o atributo `require` a ele:

Para garantir uma boa experiência do usuário e atender às diretrizes de acessibilidade, devemos sempre indicar quais campos são obrigatórios. Isso geralmente será feito adicionando um asterisco (*) ao rótulo do campo obrigatório, como fizemos no exemplo.

Validações de comprimento de texto

Às vezes, queremos que os usuários insiram uma quantidade mínima ou máxima de texto em um campo. Exemplos reais de uso dessas validações seriam o antigo limite de 140 caracteres que o Twitter costumava ter em seu campo de status ou ter restrições de comprimento mínimo e máximo em um campo de nome de usuário.

Validação de comprimento mínimo

Para adicionar a validação de comprimento mínimo, damos ao controle de formulário um atributo `minlength` com um valor inteiro que representa a quantidade mínima de caracteres que queremos permitir no controle de formulário:

Tente inserir menos de três caracteres na área de texto e clicar no botão de tweet para ver a validação em ação.

Validação de comprimento máximo

Para adicionar uma validação de comprimento máximo, damos ao controle de formulário um `maxlength`atributo com um valor inteiro que representa a quantidade máxima de caracteres que queremos permitir no controle de formulário:

Com a validação de comprimento máximo, o navegador impedirá que os usuários insiram mais caracteres do que o valor do atributo de comprimento máximo. Tente isso por si mesmo no exemplo acima.

Combinando validações

O HTML nos permite aplicar quantas validações desejarmos a um controle de formulário. Por exemplo, podemos dar a nossa área de texto de tweet validações `minlength` e `maxlength`:

Isso nos dá muito mais escopo para controlar o que os usuários inserem.

Validações de intervalo numérico

Assim como muitas vezes precisamos controlar o comprimento de controles de formulário baseados em texto, haverá muitas situações em que desejaremos controlar o intervalo de valores que os usuários podem inserir em controles de formulário baseados em números.

Podemos fazer isso com os atributos min e max, o que nos permite definir os limites inferior e superior do valor inserido no controle do formulário. Os atributos min e max só funcionam com controles de formulário baseados em números, como entradas de número, datas e hora. Você pode ver a lista completa de elementos suportados na documentação do MDN .

Alguns casos de uso do mundo real para usar essas validações seriam limitar a quantidade em um formulário de pedido de produto ou escolher o número de passageiros em um formulário de reserva de voo.

Validação mínima

Para adicionar uma validação de valor mínimo, damos ao controle de formulário um atributo `min` com um valor inteiro que representa o número mínimo que queremos que o controle de formulário aceite:

Tente enviar o formulário com uma quantidade de 0 para ver a validação em ação.

Validação máxima

Para adicionar uma validação de valor máximo, damos ao controle de formulário um atributo `max` com um valor inteiro que representa o número máximo que queremos que o controle de formulário aceite:

Tente enviar o formulário com sete passageiros para ver a validação em ação.

Validações de padrões

Para garantir que obtenhamos as informações corretas dos usuários, muitas vezes queremos garantir que os dados correspondam a um padrão específico. Os aplicativos do mundo real verificariam se um número de cartão de crédito ou um CEP está no formato correto.

Para adicionar uma validação de padrão, damos ao controle de formulário um atributo `pattern` com uma expressão regular como valor. Em nosso exemplo, estamos usando a validação de padrão para garantir que um CEP dos EUA esteja no formato correto (5 números seguidos por um traço opcional e mais 4 números):

A inserção de um CEP incorreto e o envio do formulário exibirá o seguinte erro de validação no navegador “Por favor, corresponda ao formato solicitado”. Isso não é muito útil, pois não informa como corrigir o problema.

Podemos adicionar uma mensagem de validação mais descritiva dando um atributo `title` à nossa entrada:

Quando enviamos o formulário com um CEP incorreto, seremos recebidos com uma mensagem mais útil nos informando exatamente o que deu errado e como resolvê-lo.

Também é uma boa prática usar um atributo `placeholder` para mostrar aos usuários um exemplo do padrão esperado que eles precisam inserir:

O atributo pattern só pode ser usado em elementos `<input>`. Alguns elementos de entrada já validam dados que correspondem a um determinado padrão. Por exemplo, o campo de entrada de email garantirá que um email válido seja inserido e o elemento de entrada de url verificará para garantir que o URL comece com http ou https:

Validações de estilo

Podemos direcionar controles de formulário que passaram ou falharam em validações usando as pseudo-classes `:valid` e `:invalid`.

Para ver isso em ação, usaremos nosso exemplo de e-mail e site que analisamos anteriormente:

Em primeiro lugar, direcionamos todas as entradas válidas e damos a elas uma borda verde. Nossas entradas de email e URL inicialmente têm uma borda verde, pois não são campos obrigatórios e são válidos.

Quando um campo é inválido, damos a ele uma borda vermelha. Tente inserir um endereço de e-mail e URL inválidos para ver como fica.

Conclusão

As validações integradas levarão você longe para garantir que seus usuários insiram os dados corretos. Eles são rápidos e fáceis de adicionar. No entanto, eles têm suas limitações.

Às vezes, você precisará incluir validações que as validações internas não poderão fazer. Por exemplo, validar que uma entrada de senha e uma entrada de confirmação de senha tenham o mesmo valor ou validar que um nome de usuário ainda não foi usado. Também estamos limitados com o que podemos fazer com o estilo das mensagens de validação e o conteúdo dentro delas.

Nesse caso, precisaremos ser criativos e fazer validações personalizadas usando JavaScript e CSS. Veremos como obter a validação via JavaScript em uma lição futura.

Também vale a pena notar que as validações do lado do cliente não são uma bala de prata para garantir que os usuários insiram os dados corretos. Para garantir a integridade de todos os dados do usuário que entram em nossos sistemas, também devemos ter validações no lado do servidor. Abordaremos esse lado das validações posteriormente no currículo.