Meu primeiro formulário HTML

Este é um artigo introdutório para formulários HTML. Através de um simples formulário de contato, nós veremos os requisitos básicos para construir formulários HTML. Esse artigo assume que você não sabe nada sobre formulários HTML, mas presume que você conhece o básico de HTML e CSS.

Antes de começarmos

O que são formulários HTML?

Formulários HTML são um dos principais pontos de interação entre um usuário e um web site ou aplicativo. Eles permitem que os usuários enviem dados para o web site. Na maior parte do tempo, os dados são enviados para o servidor da web, mas a página da web também pode interceptar para usá-los por conta própria.

Um formulário HTML é feito de um ou mais widgets. Esses widgets podem ser campos de texto (linha única ou de várias linhas), caixas de seleção, botões, checkboxes ou radio buttons. A maior parte do tempo, estes elementos são emparelhados com uma legenda que descreve o seu objetivo.

O que você precisa para trabalhar com formulários?

Você não precisa de nada mais do que o que é requisitado para trabalhar com HTML: Um editor de texto e um navegador. É claro, que se você esta acostumado a usá-los você pode ter vantagens de uma IDE completa como Visual Studio , Eclipse , Aptana , etc., mas cabe somente a você.

A principal diferença entre um formulário de HTML e um documento regular de HTML é que, maioria das vezes, o dado coletado é enviado ao servidor. Nesse caso, você precisa configurar um servidor web para receber e processar os dados. Como configurar um servidor está além do escopo deste artigo, mas se você quer saber mais, consulte o artigo dedicado a este tema: Envio e recuperação de dados do formulário (en-US).

Desenhando seu formulário

Antes de começar a codificar, é sempre melhor dar um passo atrás e tomar o tempo para pensar sobre o seu formulário. Desenhando um rascunho rápido irá ajudar a definir o conjunto correto de dados que você quer perguntar ao usuário. De um ponto de vista da experiência do usuário (UX), é importante lembrar que quanto maior o seu formulário, maior o risco de perder os usuários. Mantenha o formuário simples e mantenha o foco: **peça apenas o que é absolutamente necessário**. A criação de formulários é um passo importante quando você está construindo um site ou um aplicativo. Está além do escopo deste artigo cobrir as formas, mas se você quiser se aprofundar neste tópico você deve ler os seguintes artigos:

- A Smashing Magazine tem ótimos artigos sobre UX nos formulários, mas talvez o mais importante é o Extenso Guia para a usabilidade em formulários Web .
- UXMatters também é um recurso muito atencioso com bons conselhos de melhores práticas básicas para conceitos complexos, tais como formulários de várias páginas .

Neste artigo vamos construir um formulário de contato simples. Vamos fazer um esboço.

O nosso formulário terá três campos de texto e um botão. Basicamente, pedimos ao usuário o seu nome, seu e-mail e a mensagem que deseja enviar. Ao apertar o botão apenas irá enviar os dados para o servidor web.

Sujar as mãos com HTML

Ok, agora estamos prontos para ir para o código HTML do nosso formulário. Para construir o nosso formulário de contato, vamos utilizar os seguintes elementos `<form>` , `<label>` ,

`<input>` , `<textarea>` , e `<button>` .

O Elemento `<form>`

Todos formulários HTML começam com um elemento `<form>` como este:

HTML

```
<form action="/pagina-processa-dados-do-form" method="post"></form>
```

Este elemento define um formulário. É um elemento de container como um elemento `<div>` ou `<p>` , mas ele também suporta alguns atributos específicos para configurar a forma como o formulário se comporta. Todos os seus atributos são opcionais, mas é considerada a melhor prática sempre definir pelo menos o atributo `action` e o atributo `method` .

- O atributo ***action*** define o local (uma URL) em que os dados recolhidos do formulário devem ser enviados.
- O atributo ***method*** define qual o método HTTP para enviar os dados (ele pode ser "**GET**" ou "**POST**" (veja as diferenças aqui).

Se você quiser se aprofundar em como esses atributos funcionam, está detalhado no artigo Enviando e recebendo dados de um formulário (en-US)

Adicionar campos com os elementos `<label>` , `<input>` , e `<textarea>`

O nosso formulário de contato é muito simples e contém três campos de texto, cada um com uma etiqueta. O campo de entrada para o nome será um campo básico texto de linha única("input"); o campo de entrada do e-mail será um campo de texto com uma única linha("input") que vai aceitar apenas um endereço de e-mail; o campo de entrada para a mensagem será um campo de texto de várias linhas("textarea").

Em termos de código HTML, teremos algo assim:

HTML

```
<form action="/pagina-processa-dados-do-form" method="post">
  <div>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" />
  </div>
  <div>
    <label for="email">E-mail:</label>
    <input type="email" id="email" />
  </div>
  <div>
    <label for="msg">Mensagem:</label>
    <textarea id="msg"></textarea>
  </div>
</form>
```

Os elementos `<div>` estão lá para estruturar nosso código e deixar a estilização mais fácil (ver abaixo). Observe o uso do atributo ***for*** em todos os elementos `<label>` ; é uma maneira para vincular uma ***label*** à um campo do formulário. Este atributo faz referência ao ***id*** do campo correspondente. Há algum benefício para fazer isso, é a de permitir que o usuário clique no rótulo para ativar o campo correspondente. Se você quer uma melhor compreensão dos outros benefícios deste atributo, tudo é detalhado no artigo: How to structure an HTML form (en-US)(en).

No elemento `<input>` , o atributo mais importante é o atributo `type` . Esse atributo é extremamente importante porque define a forma como o elemento `<input>` se comporta. Ele pode mudar radicalmente o elemento, então preste atenção a ele. Se você quiser saber mais sobre isso, leia o artigo native form widgets (en-US). Em nosso exemplo, nós usamos somente o `type="text"` , valor padrão para este atributo. Ele representa um campo de texto com uma única linha que aceita qualquer tipo de texto sem controle ou validação. Nós também usamos o `type="email"` que define um campo de texto com uma única linha que só aceita um endereço de e-mail bem-formados. Este último valor torna um campo de texto básico em uma espécie de campo "inteligente", que irá realizar alguns testes com os dados digitados pelo usuário. Se você quiser saber mais sobre a validação de formulário, detalharemos melhor no artigo Validação de dados de formulário (en-US).

Por último, mas não menos importante, observe a sintaxe de `<input />` e `<textarea> </textarea>` . Esta é uma das esquisitices do HTML. A tag **`<input />`** é um elemento que se auto-fecha, o que significa que se você quiser encerrar formalmente o elemento, você tem que adicionar uma barra "**/**" no final do próprio elemento e não uma tag de fechamento. No

entanto, o tipo `<textarea>` não é um elemento de auto-fechamento, então você tem que fechá-lo com a tag final adequada. Isso tem um impacto sobre um recurso específico de formulários HTML: a maneira como você define o valor padrão. Para definir o valor padrão de um elemento `<input>` você tem que usar o atributo `value` como este:

HTML

```
<input
  type="text"
  value="Por padrão, este elemento será preenchido com este texto " />
```

Pelo contrário, se você deseja definir o valor padrão de um elemento `<textarea>` , você só tem que colocar esse valor padrão no meio das tags, entre tag inicial e a tag final do elemento `<textarea>` , como abaixo:

HTML

```
<textarea>Por padrão, este elemento será preenchido com este texto </textarea>
```

E um elemento `<button>` para concluir

O nosso formulário está quase pronto; nós temos apenas que adicionar um botão para permitir que o usuário envie seus dados depois de ter preenchido o formulário. Isto é simplesmente feito usando o elemento `<button>` :

HTML

```
<form action="/pagina-processa-dados-do-form" method="post">
  <div>
    <label for="name">Nome:</label>
    <input type="text" id="name" />
  </div>
  <div>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" />
  </div>
  <div>
    <label for="msg">Mensagem:</label>
    <textarea id="msg"></textarea>
  </div>
  <div class="button">
    <button type="submit">Enviar sua mensagem</button>
```

```
  </div>
</form>
```

Um botão pode ser de três tipos: `submit`, `reset`, ou `button`.

- Um clique sobre um botão de `submit` envia os dados do formulário para a página de web definida pelo atributo `action` do elemento `<form>`.
- Um clique sobre um botão de `reset` redefine imediatamente todos os campos do formulário para o seu valor padrão. De um ponto de vista na usabilidade do usuário(UX), isso é considerado uma má prática.
- Um clique em um botão do tipo `button` faz ...ops, nada! Isso soa bobo, mas é incrivelmente útil para construir botões personalizados com JavaScript, ou seja, ele pode assumir qualquer comportamento através desta linguagem.

Note que você também pode usar o elemento `<input>` com o tipo correspondente para produzir um botão. A principal diferença com o elemento `<button>` é que o elemento `<input>` permite apenas texto sem formatação como seu valor, enquanto que o elemento `<button>` permite que o conteúdo HTML completo como seu valor.

Vamos deixar um pouco mais legal com CSS

Agora que temos o nosso formulário HTML, se você olhar para ele em seu navegador favorito, você vai ver que ele parece meio feio.

Vamos deixar ele um pouco mais legal com os códigos CSS a seguir:

Vamos começar com o próprio formulário; vamos centralizá-lo e torná-lo visível com uma borda:

CSS

```
form {
  /* Apenas para centralizar o form na página */
  margin: 0 auto;
  width: 400px;
  /* Para ver as bordas do formulário */
```

```
  padding: 1em;
  border: 1px solid #ccc;
  border-radius: 1em;
}
```

Então, adicionaremos algum espaço entre cada conjunto de campos do form:

CSS

```
form div + div {
  margin-top: 1em;
}
```

Agora vamos focar nas `labels`. Para fazer o nosso formulário mais legível, é considerada a melhor prática ter todas as etiquetas do mesmo tamanho e alinhadas do mesmo lado. Nesse caso, vamos alinhá-los para a direita, mas em alguns casos, o alinhamento à esquerda pode ficar bem também.

CSS

```
label {
  /*Para ter certeza que todas as labels tem o mesmo tamanho e estão propriamente alinhadas
*/
  display: inline-block;
  width: 90px;
  text-align: right;
}
```

Uma das coisas mais difíceis de fazer em formulários HTML são os estilo dos próprios campos. Os campos de texto são fáceis de estilizar, mas alguns outros campos não são. Se você quiser saber mais sobre estilização de formulários HTML, leia o artigo Styling HTML forms (en-US).

Aqui vamos usar alguns truques comuns: fontes de harmonização, tamanho e bordas:

CSS

```
input,
textarea {
  /* Para certificar-se que todos os campos de texto têm as mesmas configurações de fonte.
Por padrão, textareas ter uma fonte monospace*/
```

```
  font: 1em sans-serif;

  /* Para dar o mesmo tamanho a todos os campo de texto */
  width: 300px;
  -moz-box-sizing: border-box;
  box-sizing: border-box;

  /* Para harmonizar o look & feel das bordas nos campos de texto*/
  border: 1px solid #999;
}
```

mdn web docs

está no formulário.

CSS

```
input:focus,
textarea:focus {
  /* Dar um pouco de destaque nos elementos ativos */
  border-color: #000;
}
```

Campos de texto de várias linhas precisam de alguns estilos personalizados sozinhos. Por padrão, um elemento `<textarea>` é um bloco em linha com sua parte inferior alinhada à linha de base do texto. Na maioria das vezes, não é baseline o que queremos. Nesse caso, a fim de alinhar a `label` e o campo, temos que alterar a propriedade *vertical-align* do `<textarea>` para *top*.

Observe também o uso da propriedade de *resize*, que é uma forma de permitir que os usuários redimensionar um elemento `<textarea>` .

CSS

```
textarea {
  /* Para alinhar corretamente os campos de texto de várias linhas com sua label*/
  vertical-align: top;

  /* Para dar espaço suficiente para digitar algum texto */
  height: 5em;
```

```
  /* Para permitir aos usuários redimensionarem qualquer textarea verticalmente. Ele não
funciona em todos os browsers */
  resize: vertical;
}
```

Muitas vezes, os botões precisam de estilos especiais também. Para esse fim, nós o colocamos dentro de uma `<div>` com uma classe css *`button`*. Aqui, queremos que o botão esteja alinhado com os outros campos . Para conseguir isso, temos de imitar a presença de uma `<label>`. Isso é feito utilizando `padding` e `margin`.

CSS

```
.button {
  /* Para posicionar os botões para a mesma posição dos campos de texto */
  padding-left: 90px; /* mesmo tamanho que os elementos do tipo label */
}
button {
  /* Esta margem extra representa aproximadamente o mesmo espaço que o espaço entre as
labels e os seus campos de texto*/
  margin-left: 0.5em;
}
```

Agora o nosso formulário parece muito mais bonito.

Enviar os dados para seu servidor web

A última parte, e talvez a mais complicado, é lidar com dados de formulário no lado do servidor. Como dissemos antes, na maioria das vezes, um formulário HTML é uma forma conveniente para perguntar ao usuário os dados e enviá-lo para um servidor web.

O elemento `<form>` definirá onde e como enviar os dados, graças ao atribudo ***action*** e ao atributo ***method***

Mas não é o suficiente. Nós também precisamos dar um nome a nossos dados. Esses nomes são importantes em ambos os lados; no lado do navegador, ele informa ao navegador que nome dar a cada pedaço de dados, e no lado do servidor, ele permite que o servidor lidar com cada pedaço de dados pelo nome.

Então, para nomear seus dados, você precisará usar o atributo ***name*** em cada campo do formulário que irá recolher uma parte específica dos dados:

HTML

```
<form action="/pagina-processa-dados-do-form" method="post">
  <div>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="usuario_nome" />
  </div>
  <div>
    <label for="email">E-mail:</label>
    <input type="email" id="email" name="usuario_email" />
  </div>
  <div>
    <label for="msg">Mensagem:</label>
    <textarea id="msg" name="usuario_msg"></textarea>
  </div>

  <div class="button">
    <button type="submit">Enviar sua mensagem</button>
  </div>
</form>
```

Em nosso exemplo, o formulário irá enviar 3 informações, chamados "usuario_nome", "usuario_email" e "usuario_msg" e os dados serão enviados para a URL **"/pagina-processa-dados-do-form"** com o método HTTP: **POST** .

No lado do servidor, o script na URL "***/pagina-processa-dados-do-form***" receberá os dados como uma lista de itens 3 de chave/valor contidos na solicitação HTTP. A forma como o script vai lidar com esses dados fica a seu critério. Cada linguagem server-side (PHP, Python, Ruby, Java, C #, etc.) tem seu próprio mecanismo. Está além do escopo deste guia aprofundar o assunto, mas se você quiser saber mais, vamos dar alguns exemplos no artigo Enviando e recuperando dados de formulário (en-US).

Conclusão

Parabéns! Você construiu seu primeira formulário HTML. Aqui está um exemplo do resultado final.

Live example

Live sample failed!

An error occurred trying to render this live sample.
Consider filing an issue or trying your hands at a fix of your own.

Error details:

```
Error: From URL /en-US/docs/Web/Guide/HTML/Forms/My_first_HTML_form/Example no folder on disk could be found. Tried to find a folder called en-us/web/guide/html/forms/my_first_html_form/example
```

Agora é hora de dar uma olhada mais profunda. Formulários HTML são muito mais poderoso do que o que nós vimos aqui e os outros artigos deste guia irá ajudá-lo a dominar o resto.

This page was last modified on 3 de ago. de 2023 by MDN contributors.