

Linguagens de Marcação e Scripts

Prof. Aníbal Cavalcante de Oliveira

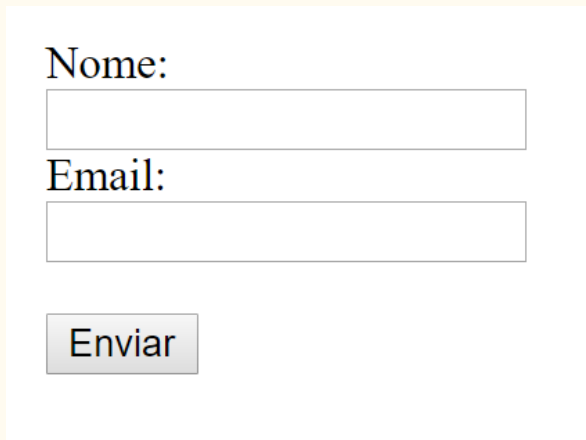
UFC - QXD0164 - 2019.2

Agenda - Aula 15

- Formulários em HTML.

Formulários em HTML

A linguagem HTML disponibiliza formulários para criar uma seção de uma página HTML, que possua elementos de entrada de dados que permitem ao usuário submeter uma determinada informação a um servidor na web. Por exemplo:



Nome:

Email:

Formulários em HTML <form>

Resumidamente:

- Um formulário é um elemento do tipo bloco e é criado pela tag <form>.
- Serve como um "container" para outros elementos de entrada de dados.
- Para criarmos os elementos de entrada utilizamos tag especiais como a <input>.
- Utilizamos os métodos GET e POST do protocolo http para enviar as informações preenchidas no formulário, para o servidor web.
- O envio de dados do formulário é chamado de **requisição**.
- Para toda **requisição** ao servidor web, ele envia uma "resposta".
- A resposta possui dentro dela informações, por exemplo, uma página html.
- Devemos dizer a URL do **servidor web** que queremos utilizar com o atributo **action**.
- Devemos identificar cada elemento de entrada do nosso <form>.

Conversa entre Cliente (Dados de um Formulário e um Servidor Web)



FORM ELEMENTS

(FRONTEND HTML & CSS)



FORM PROCESSING

(BACKEND SERVER)

HTML requisição para um formulário.

Exemplo: Vamos realizar uma requisição para um servidor web na minha máquina, utilizando o método **GET**, está rodando um serviço em Node.js, como o nome **form**. Falaremos mais sobre requisições nas próximas aulas.



```
<form method="POST" action="http://127.0.0.1:3000/usuarios">  
  Nome:<br>  
    <input type="text" name="nome" id="nome">  
  <br>  
  Email:<br>  
    <input type="text" name="email" id="email">  
  <br><br>  
    <input type="submit" value="Enviar">  
</form>
```

Elementos de entrada: A tag `<input>`

1. A tag `<input>` é o elemento mais básico de entrada em um formulário.
2. Ela especifica uma **caixa de entrada** onde o usuário pode inserir dados.
3. Um campo do tipo `<input>` pode variar de muitas formas, dependendo do seu tipo.
4. Definimos o seu tipo através do valor do atributo **type**.
5. Dentre os valores possíveis para o type temos: **text**, **password**, **email**, **number**, **checkbox**, **reset**, **radio**, **submit**, **range**, **button**, **color**, **date**, **datetime-local**, e **file**.

`<input type="text">`

Exemplo: Define um campo de texto em uma linha.

```
<form>
  Nome:<br>
  <input type="text" name="nome"><br>
  Sobrenome:<br>
  <input type="text" name="sobrenome">
</form>
```


`<input type="password">`

Exemplo: Define um campo para entrada de senhas em uma linha.

```
<form>
  Nome:<br>
  <input type="text" name="nome"><br>
  Senha:<br>
  <input type="password" name="senha">
</form>
```

`<input type="password">`

Exemplo: Define um campo para entrada de senhas em uma linha.

```
<form>
  Usuário:<br>
  <input type="text" name="username"><br>
  Senha:<br>
  <input type="password" name="senha">
</form>
```

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

Exemplo: Define um botão para enviar dados de formulário para uma URL de um servidor web. Lembra de sempre devemos especificar o atributo **action**.

```
<form action="localhost:1337/form" method="GET">  
  Usuário:<br>  
  <input type="text" name="username"><br>  
  Senha:<br>  
  <input type="password" name="senha">  
  <input type="submit">  
</form>
```

`<input type="submit">`

Exemplo: Define um botão para enviar dados de formulário para uma URL de um servidor web. Lembra de sempre devemos especificar o atributo **action**. Se não especificarmos o atributo **value** o browser dá um nome padrão para o botão.

```
<form action="localhost:1337/form" method="GET">
  Usuário:<br>
  <input type="text" name="username"><br>
  Senha:<br>
  <input type="password" name="senha">
  <input type="submit">
</form>
```

`<input type="reset">`

Exemplo: Define um botão de reset que irá repor todos os valores do formulário para seus valores padrão. O atributo **value** define o valor padrão.

```
<form>
  Nome:<br>
  <input type="text" name="nome" value=""><br>
  Sobrenome:<br>
  <input type="text" name="sobrenome" value="">
  <br><br>
  <input type="submit">
  <input type="reset">
</form>
```

`<input type="radio">`

Exemplo: Define um botão do tipo radio. Botões de radio permitem que o usuário selecionar apenas uma opção dentro formulário. Observe o atributo **name**.

```
<p> CAMPEÃO BRASILEIRO DE 2019 </p>
<form>
  <input type="radio" name="time" value="Flamengo" checked> Flamengo
  <br>
  <input type="radio" name="time" value="Ceará"> Ceará
  <br>
  <input type="radio" name="time" value="Outro">Outro
  <br><br>
  <input type="submit">
</form>
```

`<input type="checkbox">`

Exemplo: Define uma caixa de verificação. Checkboxes permitem ao usuário selecionar zero ou mais opções por formulário.

```
<p> MELHORES TIMES DO BRASILEIRO DE 2019 </p>
<form>
  <input type="checkbox" name="time1" value="Flamengo" checked> Flamengo
  <br>
  <input type="checkbox" name="time2" value="Ceará"> Ceará
  <br>
  <input type="submit">
</form>
```

`<input type="button">`

Exemplo: Define um botão. Não precisa estar dentro de um formulário, pode chamar um script em **Javascript**, por exemplo para exibir um caixa de alerta.

```
<input type="button" onclick="alert('Bora Vozão!')" value="Ceará!">
```

```
<input type="button" onclick="alert('Bora Mengão!')" value="Flamengo!">
```


`<input type="date">`

Exemplo: Define um campo de uma data. Dependendo do browser pode exibir uma caixa pop-up com um calendário.

```
<form>
  Aniversário: <input type="date" name="bday">
  <br><br>
  <input type="submit">
  <input type="reset">
</form>
```

`<input type="email">`

Exemplo: O `<input type="email">` é usado para campos de entrada que devem conter um endereço de e-mail. Dependendo suporte ao navegador, o endereço de e-mail podem ser validados automaticamente quando submetido. Alguns smartphones reconhecem o tipo de e-mail, e adicionar ".com" para o teclado para corresponder à entrada de e-mail.

```
<form>  
  E-mail:  
  <input type="email" name="email">  
  <input type="submit">  
</form>
```

`<input type="file">`

Exemplo: Define um campo de arquivo de seleção e um botão "Procurar" para o upload de arquivos para o servidor.

```
<form >  
  Selecione um arquivo: <input type="file" name="arquivo"><br><br>  
  <input type="submit">  
</form>
```

`<input type="number">`

Exemplo: Define um numérico campo de entrada. Você também pode definir restrições sobre o que os números são aceitos, e o passo de incremento. O exemplo a seguir exibe um campo de entrada numérico, onde você pode inserir um valor de 1 a 5:

```
<form>  
  Quantidade:  
  <input type="number" name="unidades" min="0" max="100" step="10">  
  <input type="submit">  
</form>
```

`<input type="range">`

Exemplo: Define uma faixa de valores de entrada.

```
<form>  
  Pontos:  
  <input type="range" name="pontos" min="0" max="10">  
  <input type="submit">  
</form>
```

`<input type="tel">`

Exemplo: Define uma entrada para números de telefones.

```
<form>  
  Celular:  
  <input type="tel" name="cel" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">  
</form>
```

Elementos de entrada: A tag <select>

1. A tag <select> define uma lista do tipo drop-down.
2. Utilizamos a tag <option> para representar cada elemento da lista.
3. O atributo **selected** define um item da lista pré-selecionado.

```
<form>
  <select name="cars">
    <option value="volvo" selected>Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
```

Elementos de entrada: A tag <select>

Podemos ainda definir a quantidade de valores visíveis na lista através do atributo **size**.

```
<form>
  <select name="cars" size="3">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
    <option value="honda">Honda</option>
    <option value="vw">Volkswagen</option>
  </select>
  <br><br>
  <input type="submit">
</form>
```


Elementos de entrada: A tag <select>

Podemos seleccionar mais de um item na lista através do atributo **multiple**.

```
<form>
  <select name="cars" size="4" multiple>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
    <option value="honda">Honda</option>
    <option value="vw">Volkswagen</option>
  </select>
  <br><br>
  <input type="submit">
</form>
```

A tag <datalist>

A tag <datalist> especifica uma lista de opções pré-definidas para um elemento <input>. É utilizado para fornecer um recurso "**autocompletar**" em elementos <input>. Os usuários verão uma lista **drop-down** de opções pré-definidas como eles dados introduzidos.

```
<input list="browsers" name="browser">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit">
</form>
```

A tag <textarea>

Exemplo: Define um campo para entrada de texto com múltiplas linhas.

```
<h2>Textarea</h2>
<form>
  <textarea name="message" rows="10" cols="30">
    Um bom dia começa com um café e um sorriso.
  </textarea>
  <br>
  <input type="submit">
</form>
```

- O atributo **rows** especifica o número visível de linhas em uma área de texto.
- O atributo **cols** especifica a largura visível de uma área de texto.

A tag <textarea>

Podemos definir também o tamanho com css.

```
<h2>Textarea</h2>
<form>
  <textarea name="message" style="width:200px; height:600px;"
    Um bom dia começa com um café e um sorriso.
  </textarea>
  <br>
  <input type="submit">
</form>
```

A tag <button>

Diferente das tags <input type="submit"> e <input type="button">, a tag <button> permite a criação de botões mais flexíveis, que possuem maiores possibilidades de renderização. (Ver exemplo de código aula 16)

```
<div class="buttons">
  <button type="submit" class="positive">
    
    Salvar
  </button>   <a href="/password/reset/">
    
    Mudar Password
  </a>   <a href="#" class="negative">
    
    Cancelar
  </a>
</div>
```

Atributos de elementos de entrada.

Para cada tag de entrada em HTML5, temos atributos que podem configurar como os dados podem ou devem ser inseridos, além de decorar a própria tag de entrada.

Exemplos de atributos:

- value
- readonly
- disabled
- size
- maxlength
- autocomplete
- novalidate
- autofocus
- multiple
- min e max
- placeholder
- required
- step

Introdução ao Javascript para web.

JavaScript é uma das três linguagens que todos os desenvolvedores da Web precisam aprender:

1. HTML para definir o conteúdo de páginas da web
2. CSS para especificar o layout das páginas da web
3. **JavaScript** para programar o comportamento de páginas da web.

Javascript

Páginas da Web não são o único lugar onde o JavaScript é usado. Muitos programas de desktop e servidor usam a linguagem JavaScript. O Node.js é o mais conhecido. Alguns bancos de dados, como o MongoDB e o CouchDB, também usam JavaScript.

Na web com JavaScript você pode:

1. Mudar conteúdo de um elemento
2. Mudar atributos HTML
3. Mudar estilos CSS
4. Esconder elementos
5. Mostrar elementos
6. Adicionar elementos

A tag <script>

Código **JavaScript** deve ser colocado dentro de um elemento script.

Pode ser colocado no <head> ou no <body>.

É preferível colocar os scripts no final da página, pois carregamento e compilação podem atrasar a renderização da página.

```
<!DOCTYPE html>
<html>
  <body>
    <script>
    </script>
  </body>
</html>
```

A tag <script>

Podemos criar funções em JavaScript e chama-las através dos eventos realizados na página pelo usuário. Por exemplo o clique de um botão.

```
<html>
<body>
<p></p>
<button onclick=" mostrar()">Clicar</button>
<script>
    function mostrar(){
        document.getElementsByTagName("p")[0].innerHTML="My First JavaScript";
    }
</script>
</body>
</html>
```

A tag `<script>`

Também é possível importar um arquivo .js externo

```
<script src="myScript.js"></script>
```

Vantagens:

1. Separa código JS e HTML
2. Facilita manutenção
3. Navegador pode deixar o arquivo .js em cache

Obs.: Como boa prática deixamos a tag `<script>` na parte de baixo das nossas páginas html, depois de todo o conteúdo, antes de fechar a tag `<body>`. Assim, caso o arquivo de script seja muito extenso, o browser poderá renderizar a página (html e css) e depois carregar os scripts.

Trabalhando com JavaScript (recuperando elementos da página)

É possível recuperar elementos da página web através de scripts em JS. Por exemplo:

```
//recuperar um elemento pelo seu Id
var elemento1 = document.getElementById("teste");

//recuperar um elemento pelo nome da tag, pode retornar uma lista.
var elemento2 = document.getElementsByTagName("p");

//recuperar um elemento pelo nome da sua classe, pode retornar uma lista.
var elemento3 = document.getElementsByClassName("intro");

//recuperar um elemento pelo nome, pode retornar uma lista.
var elemento4 = document.getElementsByName("sobrenome");
```

Trabalhando com JavaScript (recuperando elementos da página)

A propriedade `innerHTML` altera ou retorna o conteúdo HTML de uma tag.

```
<p id="teste"> </p>
```

```
<script>
```

```
function minhaFuncao() {
```

```
    var elemento1 = document.getElementById("teste");
```

```
    alert(elemento1.innerHTML);
```

```
}
```

```
</script>
```

Trabalhando com JavaScript (recuperando elementos da página)

A propriedade **value** altera ou retorna o conteúdo HTML para tags do tipo `<input>`.

```
Sobrenome: <input type="text" name="sobrenome">
<script>
function minhaFuncao() {
    var elemento1 = document. getElementByName("sobrenome")[0];
    alert(elemento1.value);
}
</script>
```

Trabalhando com JavaScript (recuperando elementos da página)

É possível recuperar elementos com o seletor CSS que aprendemos utilizando o método `document.querySelector("")`

```
//retorna o elemento pelo seu Id (se o id for repetido retorna o primeiro)
var elemento1 = document.querySelector("#teste");

//retorna o primeiro com a tag p.
var elemento2 = document.querySelector("p");

//retorna o primeiro elemento no documento com a classe .intro
var elemento3 = document.querySelector(".intro");
```

Trabalhando com JavaScript (recuperando elementos da página)

É possível recuperar elementos também com a o método `document.querySelectorAll("")`

```
//retorna uma lista com todos os elementos p.  
var lista1 = document.querySelectorAll("p");  
  
//retorna uma lista com todos os elementos da classe .intro  
var lista2 = document.querySelectorAll(".intro");
```


Trabalhando com JavaScript (capturando eventos)

Um evento é lançado quando ocorre determinada circunstância provocada pelo usuário ou sistema.

– Tipos de eventos

- Eventos gerados pelo sistema (não dependem do usuário)
- Eventos gerados pelo usuário (dependem do usuário)

Ex: `onLoad`, `onUnload`,...

Ex: `onClick`, `onMouseDown`, `onMouseMove`,...

```
<input type=button name="somar" value="Mostrar" onClick="mostrar()">

<script>
function mostrar() {
}
</script>
```

Construa uma calculadora JavaScript

