



UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Trabalhando com formulário

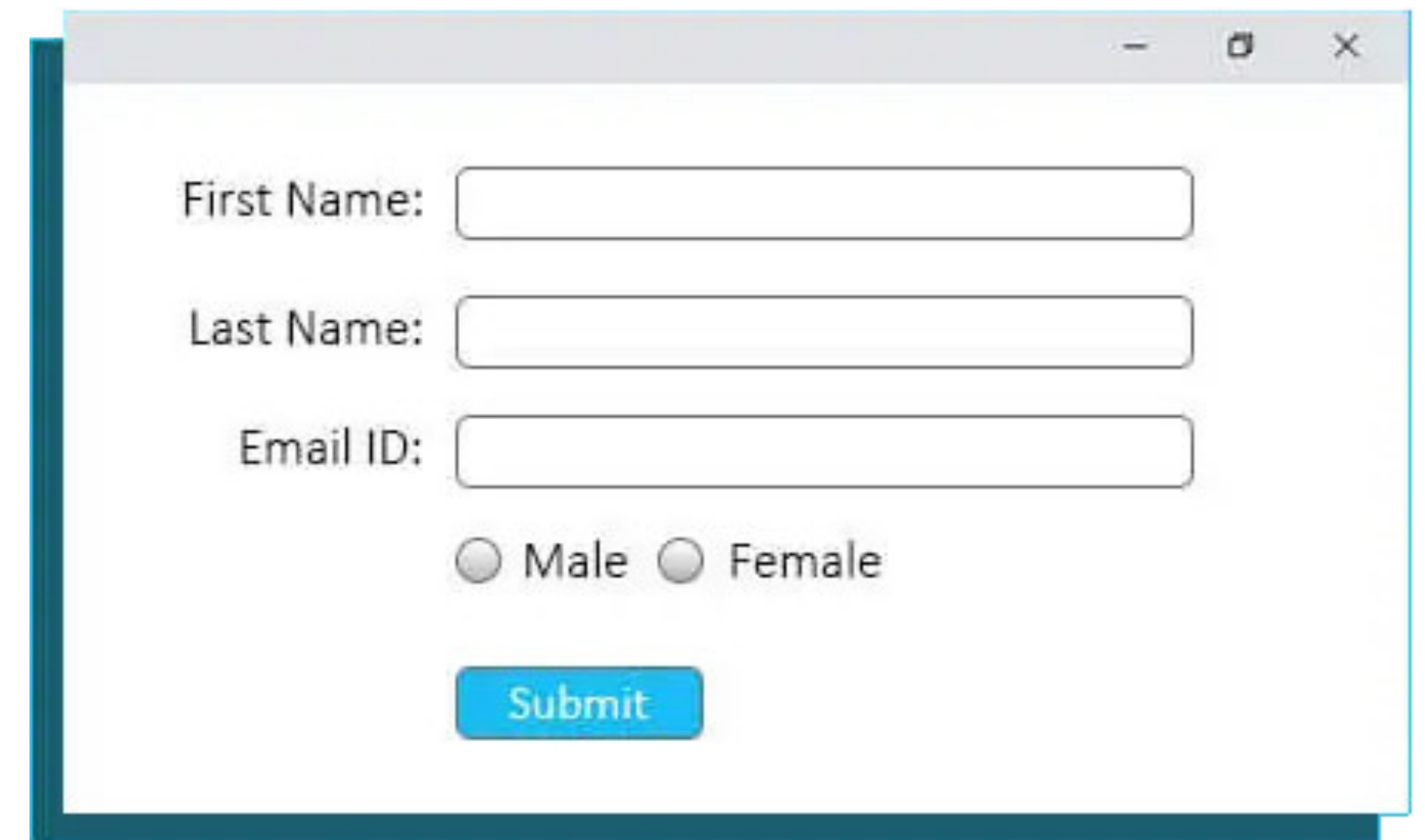
QXD0020 - Desenvolvimento de Software para Web

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Agenda

- Introdução
- Elementos de um formulário
- Enviando dados de um formulário
- Validando formulário

Introdução



A screenshot of a web form with the following fields and controls:

- First Name:
- Last Name:
- Email ID:
- Gender selection: Male Female
- Submit button:

Introdução

- Remontam aos primórdios da Web e são anteriores à própria JavaScript
- São o mecanismo por trás da primeira geração de aplicativos Web
- Neles, a entrada do usuário é obtida em elementos de formulário
 - O envio do formulário remete essa entrada para o servidor
 - O servidor processa a entrada e gera uma nova página HTML (normalmente com novos elementos de formulário) para exibição pelo cliente

Introdução

Elementos de formulário

- São um grupo de componentes de interface do usuário que aceita informações do usuário
 - Como qualquer elemento HTML, é possível manipulá-los com as técnicas de DOM
- São uma excelente maneira de obter entrada do usuário
- No cliente:
 - São utilizados mesmo quando os dados nunca são enviados para o servidor
 - No cliente um botão Submit nunca é necessário (embora ainda possa ser útil)
- Para um servidor, um formulário não tem utilidade, a não ser que possua um botão Submit

Introdução

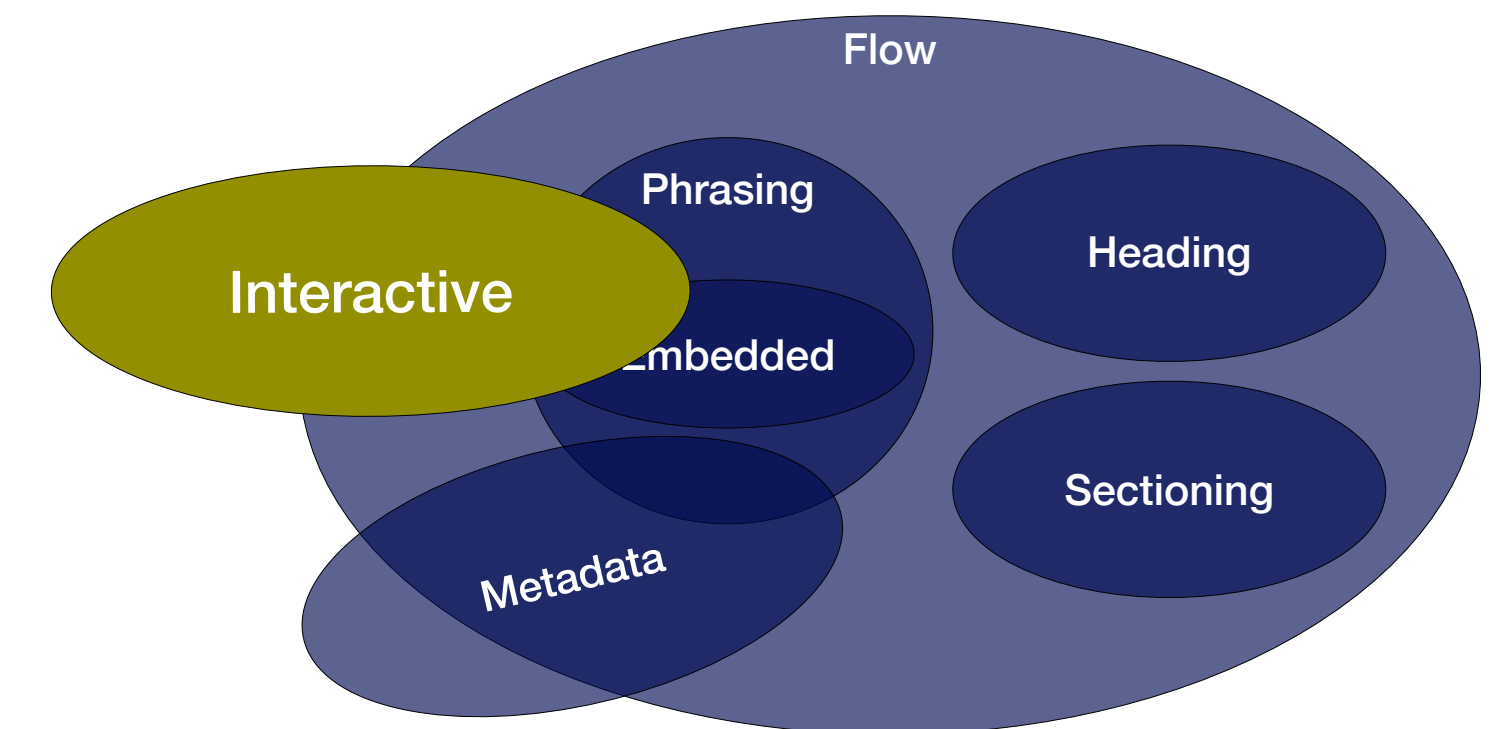
Criando um formulário

- Um formulário HTML é demarcado pela tag **<form>**
 - Cada novo formulário deve ser iniciado com essa tag
- Os elementos que compõem um formulário devem estar contidos neles
 - As tags devem ser aninhadas
- É proibido aninhar formulários
- É possível utilizar elementos de formulário sem formulários
- É possível associar elementos fora de uma tag **<form>** com um formulário

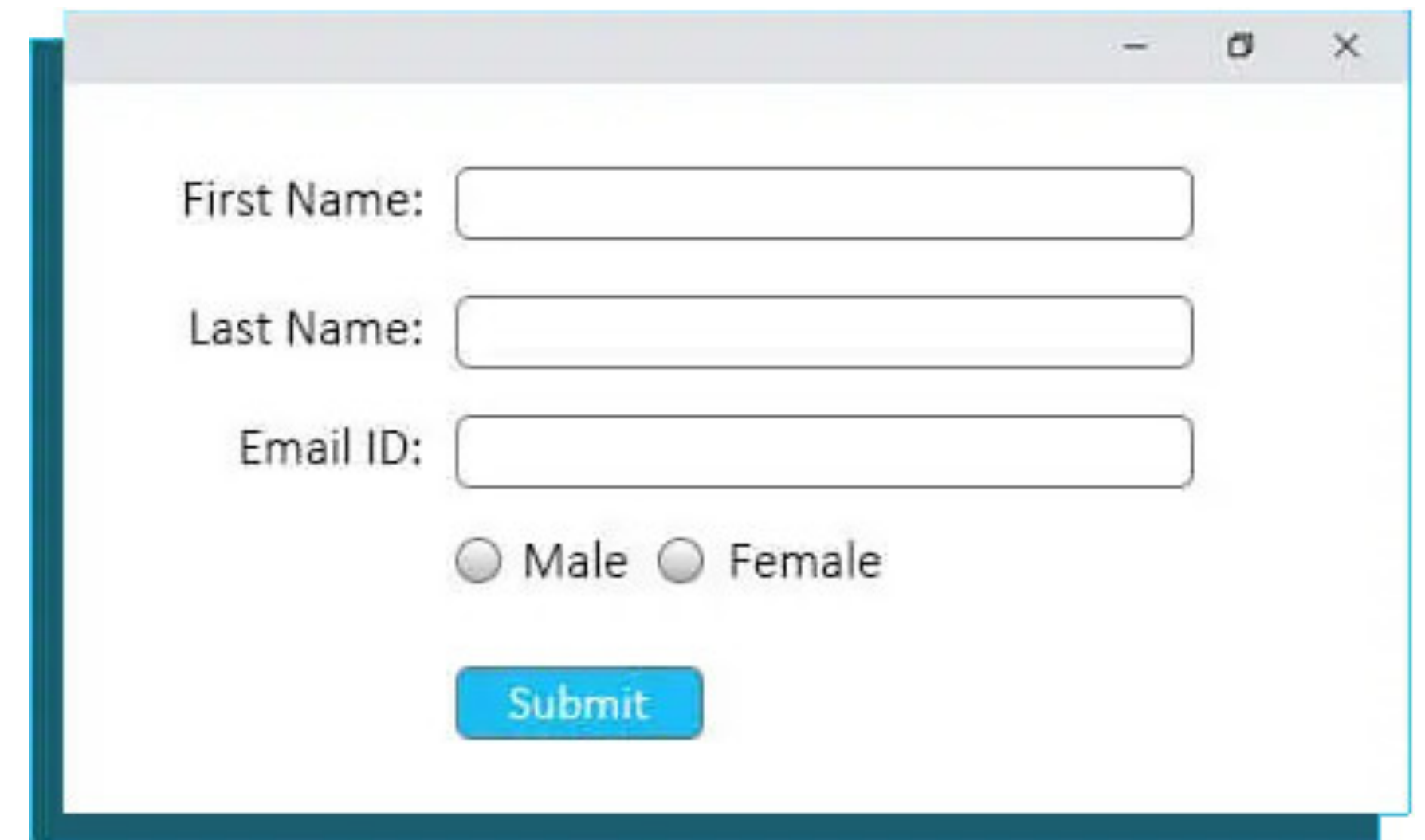
Introdução

Elementos de formulário

- Os elementos HTML que podem compor formulário fazem parte da categoria **Interactive Content**
 - Por sua vez, eles são o subconjunto a chamado **Form-associated content**
 - `<button>`, `<fieldset>`, `<input>`, `<label>`, `<meter>`, `<object>`, `<output>`, `<progress>`, `<select>` e `<textarea>`
- Possuem uma série de atributos HTML pois todos são instância de `HTMLInputElement`
 - O atributo `name` especifica o nome do parâmetro a ser enviado ao servidor
 - O atributo `value` especifica o valor inicial do campo



Elementos de um formulário



First Name:

Last Name:

Email ID:

Male Female

Elementos de um formulário

<input>

- É usado para criar vários tipos de campos para interação com o usuário
 - Campos de texto, senha, checkbox, radio, etc ...
- O tipo do campo pode ser determinado através do atributo **type**

| Tipo | Descrição |
|-----------------------|---|
| <code>text</code> | Um campo de texto com uma só linha; quebras de linha são automaticamente removidas. |
| <code>password</code> | Um campo de texto com uma só linha cujo valor é obscurecido. |
| <code>checkbox</code> | Uma caixa de marcação. Várias opções podem ser marcadas. |
| <code>radio</code> | Um botão de escolha. Apenas uma opção pode ser marcada. |
| <code>submit</code> | Um botão que envia o formulário. |

Elementos de um formulário

Tipos de input

| Tipo | Descrição | |
|-----------------------------|---|-------|
| <code>button</code> | Um botão sem comportamento padrão. | |
| <code>file</code> | Um controle que permite ao usuário selecionar um arquivo. | |
| <code>number</code> | Um controle para inserir um número de ponto flutuante. | HTML5 |
| <code>email</code> | Um campo para editar um endereço de e-mail. | HTML5 |
| <code>reset</code> | Um botão que faz o conteúdo do formulário voltar a ter seus valores padrão. | |
| <code>color</code> | Um controle para especificar cores. | |
| <code>date</code> | Um controle para inserir uma data (ano, mês e dia, sem horário). | |
| <code>datetime-local</code> | Um controle para inserir data e horário, sem fuso horário. | HTML5 |
| <code>hidden</code> | Um controle que não é exibido mas cujo valor é enviado ao servidor. | |

Elementos de um formulário

Tipos de input

| Tipo | Descrição | |
|---------------------|--|-------|
| <code>image</code> | Um botão gráfico para enviar o formulário. | |
| <code>month</code> | Um controle para inserir mês e ano, sem fuso horário. | HTML5 |
| <code>range</code> | Um controle para inserir um número limitado por um intervalo. | HTML5 |
| <code>search</code> | Um campo de texto com uma só linha para digitar termos de busca | HTML5 |
| <code>tel</code> | Um controle para inserir um número de telefone | HTML5 |
| <code>time</code> | Um controle para inserir um horário sem fuso horário. | HTML5 |
| <code>url</code> | Um campo para editar uma URL. | HTML5 |
| <code>week</code> | Um campo que permite a escolha o número (ISO 8601) da semana de um ano | HTML5 |

Elementos de um formulário

Atributos aplicáveis a tag input

| Atributo | Descrição | Aplicável |
|------------------------|--|---|
| <code>disabled</code> | Indica se o campo está desabilitado | * |
| <code>readonly</code> | Indica se o campo pode ser alterado | * |
| <code>minlength</code> | Determina a qualidade mínima de caracteres aceitos | Campos com texto |
| <code>min</code> | Determina o valor mínimo | number, range, date, datetime-local, month, time e week |
| <code>maxlength</code> | Determina a qualidade máxima de caracteres aceitos | Campos com texto |
| <code>max</code> | Determina o valor máximo | number, range, date, datetime-local, month, time e week |

Mais informações em: <https://html.spec.whatwg.org/multipage/input.html#the-input-element>

Elementos de um formulário

Atributos aplicáveis a tag input

| Atributo | Descrição | Aplicável |
|--------------------------|--|---|
| <code>pattern</code> | Determina o padrão a ser correspondido | text, date, search, url, tel, email, e password. |
| <code>placeholder</code> | Determina o texto contido no campo quando nenhum valor é informado | text, search, url, tel, email, e password. |
| <code>required</code> | Determina se um campo é obrigatório ou não | text, search, url, tel, email, password, date pickers, number, checkbox, radio, e file. |
| <code>size</code> | Determina a largura do formulário | text, search, tel, url, email, e password. |
| <code>step</code> | Determina o tamanho do incremento em input de intervalo | number, range, date, datetime-local, month, time e week |
| <code>value</code> | Valor inicial do campo | * |

Mais informações em: <https://html.spec.whatwg.org/multipage/input.html#the-input-element>

Elementos de um formulário

Exemplos

```
<input type="text" size="10" maxlength="8" required>  
<input type="password" size="10" minlength="4" required>
```

```
<input type="checkbox" name="lettuce" > Lettuce  
<input type="checkbox" name="tomato" checked> Tomato  
<input type="checkbox" name="pickles" checked> Pickles
```

```
<label>  
  <input type="radio" name="cc" value="visa" checked> Visa  
</label>  
<label>  
  <input type="radio" name="cc" value="mastercard"> MasterCard  
</label>  
<label>  
  <input type="radio" name="cc" value="amex"> American Express\  
</label>
```

Elementos de um formulário

<textarea>

- Define um campo de texto que permite múltiplas linhas
 - A quantidade de texto pode ou não ser limitada
- É uma boa prática usar os atributos **cols** e **rows** para determinar o tamanho do campo e garantir a consistência entre navegadores
- O texto inicial deve ser colocado dentro da tag

```
<textarea rows="4" cols="20">  
  Type your comments here.  
</textarea>
```

Elementos de um formulário

<select>

- É utilizada para definir uma lista de seleção
- <option> dentro do elemento <select> define uma das opções de escolha
- <optgroup> é utilizada para agrupar opções
- O atributo *selected* determina que a opção selecionada
- O atributo *multiple* permite a múltipla seleção

```
<select name="favoritecharacter" multiple>  
  <option>Jerry</option>  
  <option>George</option>  
  <option selected>Kramer</option>  
  <option>Elaine</option>  
</select>
```


Elementos de um formulário

Botões

- Submit
 - Quando clicado envia as informações para o servidor
 - O destino é especificado na tag form no atributo *action*
- Reset
 - Restaura todos os elementos com os seus valores padrões
- Button
 - Não possui nenhum efeito associado por padrão ao clique

```
<input type="submit" >  
<input type="reset" >  
<input type="button" value="Botão" >
```

Enviar

Redefinir

Botão

Elementos de um formulário

`<fieldset>`

- Permite o agrupamento de elementos com propósitos relacionados
 - Sejam eles semânticos ou apenas para estilização
- É possível adicionar uma legenda `<legend>` para descrever o agrupamento
 - Essa tag é utilizada por tecnologias assistivas

Elementos de um formulário

`<label>`

- Representa uma rótulo para um item em uma interface de usuário
- É o elemento mais importante na construção de formulários acessíveis
- Deve estar associado a um elemento de controle
- O elemento label pode ser alvo de regras de estilo CSS
- São clicáveis e portanto aumentam a área de clique o elemento alvo
 - Particularmente útil se usado com checkbox ou radiobutton

Elementos de um formulário

Exemplo

```
<form>
  <fieldset>
    <legend>Dados Pessoais</legend>
    <div>
      <label for="nome">Nome:</label>
      <input type="text" id="nome" name="nome" required>
    </div>
    <div>
      <label for="email">E-mail:</label>
      <input type="email" id="email" name="email" required>
    </div>
    <div>
      <label for="idade">Idade:</label>
      <input type="number" id="idade" name="idade" min="18" max="99">
    </div>
  </fieldset>
  <fieldset>
    <legend>Preferências</legend>
    <div>
      <label for="cor">Cor favorita:</label>
      <input type="color" id="cor" name="cor">
    </div>
    <div>
      <label for="animal">Animal de estimação favorito:</label>
      <select id="animal" name="animal">
        <option value="cachorro">Cachorro</option>
        <option value="gato">Gato</option>
      </select>
    </div>
  </fieldset>
  <input type="submit" value="Enviar">
</form>
```

```
<label>Gênero:</label>
  <div>
    <input type="radio" id="genero-masculino" name="genero" value="masculino">
    <label for="genero-masculino">Masculino</label>
  </div>
  <div>
    <input type="radio" id="genero-feminino" name="genero" value="feminino">
    <label for="genero-feminino">Feminino</label>
  </div>
  <div>
    <input type="radio" id="genero-outro" name="genero" value="outro">
    <label for="genero-outro">Outro</label>
  </div>
  <div>
    <label for="newsletter">Deseja receber nossa newsletter?</label>
    <input type="checkbox" id="newsletter" name="newsletter">
  </div>
</fieldset>
<input type="submit" value="Enviar">
</form>
```

Dados pessoais

Nome:

E-mail:

Idade:

Preferências

Cor favorita:

Animal de estimação favorito:

Gênero:

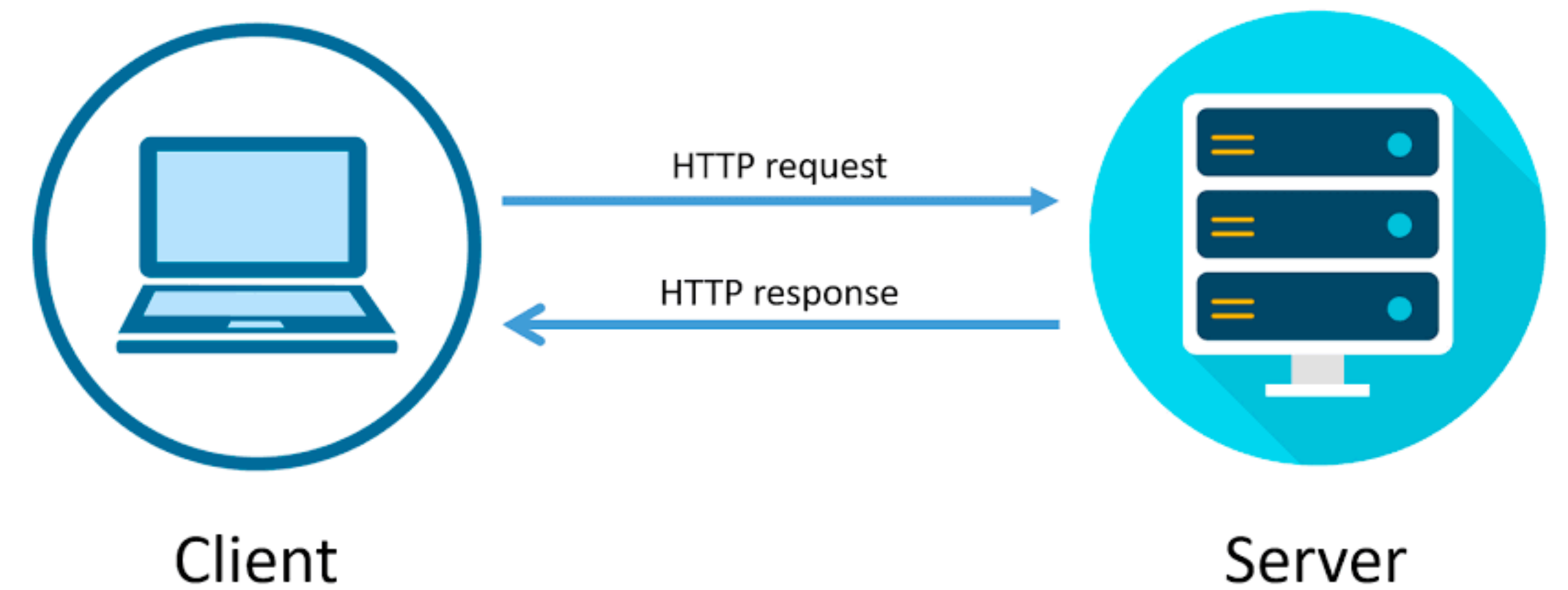
Masculino

Feminino

Outro

Deseja receber nossa newsletter?

Enviando dados de um formulário



Enviando dados de um formulário

Criando um formulário

- Para configurarmos o envio de um formulário usando alguns atributos da tag **<form>**
- *action*
 - responsável por determinar a URL da página que irá processar os dados
- *method*
 - determinar o método HTTP (GET ou POST) para o envio do dados

Enviando dados de um formulário

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Minha primeira página</title>
  </head>
  <body>
    <form action="http://foo.com" method="GET">
      <label> Nome: <input type="text" name="nome" /></label>
      <label> Sobrenome: <input type="text" name="sobrenome" /></label>
      <input type="submit" value="Gravar" />
    </form>
  </body>
</html>
```

Para onde (URL) os dados do formulário serão enviados

Método utilizado para o envio dos dados

Nome do parâmetro

Enviando dados de um formulário

Atributos da tag <form>

| Atributo | Descrição |
|-----------------------------|--|
| <code>accept-charset</code> | Determina a codificação de caracteres usada na submissão |
| <code>action</code> | Determina o local para o qual os dados serão submetidos |
| <code>autocomplete</code> | Determina se o formulário deve ser ou não autocompletado |
| <code>enctype</code> | Determina com os dados vão ser codificados (apenas no método POST) |
| <code>method</code> | Determina o método HTTP que será utilizado no envio dos dados |
| <code>name</code> | Determina o nome do formulário |
| <code>novalidate</code> | Determina se o formulário deve ou não ser validado antes do envio |

Mais informações em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/form>

Enviando dados de um formulário

O método GET

- É usado para solicitar dados de um recurso específico do servidor
- Os parâmetros da requisição (se existirem) são enviados na query string da URL
 - São visíveis na barra de endereço do navegador
 - Os dados enviados via formulário ficam visíveis
 - O corpo da requisição permanece vazio
- Há uma limitação da quantidade de dados que pode ser enviado via URL
 - São restritos aos códigos ASCII
- Adequado para enviar parâmetros de pesquisa ou identificadores
- É "cacheável" , os resultados podem ser armazenados pelo navegador

```
GET /?nome=Bruno&sobrenome=Mateus HTTP/2.0  
Host: foo.com
```

Enviando dados de um formulário

O método POST

- É usado para enviar dados para serem processados por um recurso específico no servidor
 - Leva em consideração os dados enviados corpo da requisição
- Os dados são enviados no corpo da requisição HTTP e não ficam visíveis na URL
 - É mais seguro para enviar informações sensíveis
 - Mais seguro que o método GET
- É adequado para enviar grandes quantidades de dados
 - Formulários complexos ou envios de arquivos
 - Os dados não têm restrições de tamanho
- Os dados não são armazenados em cache, pois cada solicitação é tratada como única

```
POST / HTTP/2.0
```

```
Host: foo.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 27
```

```
nome=Bruno&sobrenome=Mateus
```

Enviando dados de um formulário

GET vs POST

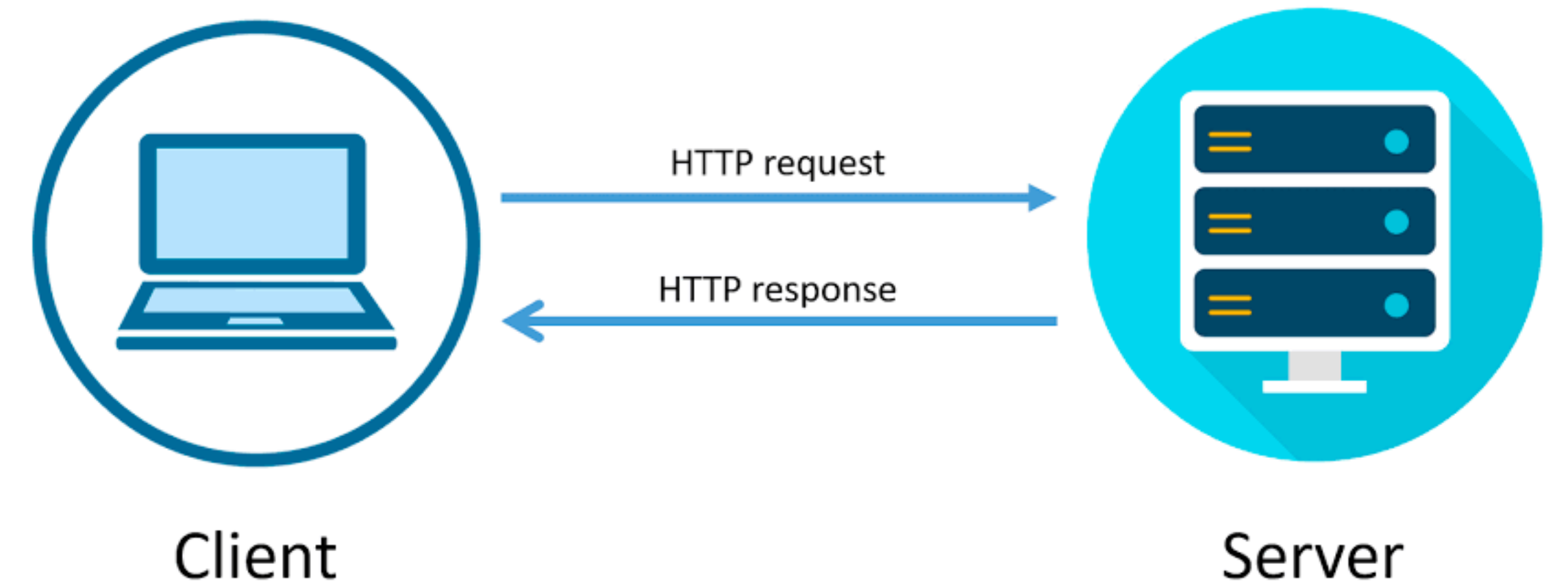
| | GET | POST |
|--------------------|-------------------------------------|---|
| Restrições de tipo | Apenas caracteres ASCII | Não há restrições. Arquivos e até dados binários podem ser enviados |
| Visibilidade | Os dados permanecem visíveis na URL | Os dados não ficam visíveis |
| Limite de tamanho | 2000 caracteres | Nenhuma limitação |
| Botão de voltar | Requisições GET são re-executadas | O navegador avisa que é necessário reenviar dados |
| Favoritar | Sim | Não |
| Hackear | Sim | Não* |
| Cache | Sim | Não |

Enviando dados de um formulário

GET vs POST - Recomendações gerais

- GET é recomendado para formulários que envolvem consultas de leitura no banco de dados
- Se os dados contém caracteres não ASCII deve-se usar POST
- Se a quantidade de dados do formulário for muito grande, deve-se usar POST
- Para envio de dados confidenciais, deve-se usar POST

Validando formulário



Validando formulário

Introdução

- Antes de enviar dados de um formulário, é importante garantir que:
 - Todos os campos obrigatórios sejam preenchidos
 - Todos os dados a serem enviados estejam no formato correto
- Isso é chamado de **validação de formulário**
 - Pode ser realizada no navegador e/ou no servidor web
 - Quando é realizada no navegador é chamada de **validação no cliente**

Validando formulário

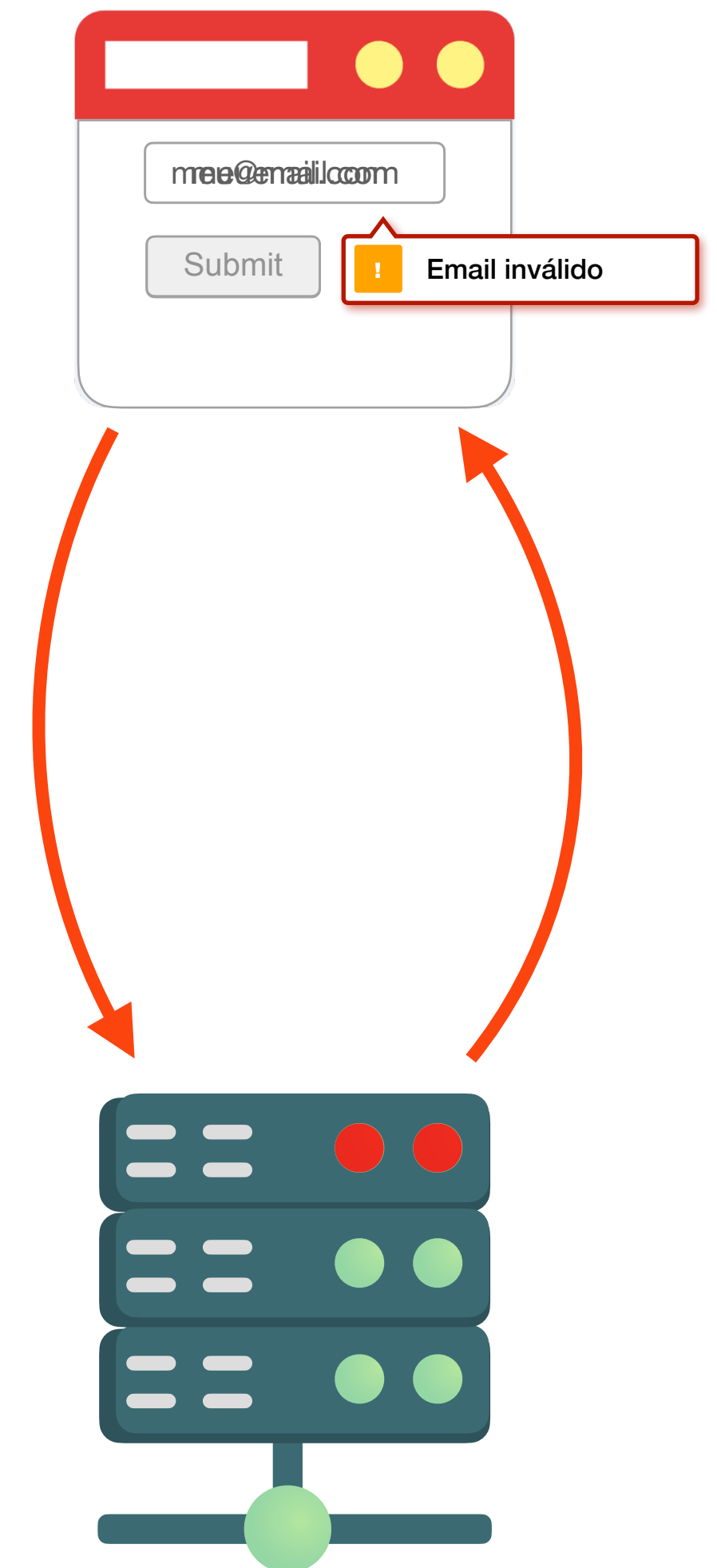
Porque a validação é tão importante?

- **Queremos obter os dados certos, no formato certo**
 - Os aplicativos não funcionarão corretamente caso os dados enviados sejam armazenados no formato errado, incorretos ou totalmente omitidos
- **Queremos proteger os dados dos nossos usuários**
 - Forçar nossos usuários a inserir senhas seguras facilita a proteção de suas informações de conta
- **Queremos nos proteger**
 - Há muitas maneiras pelas quais usuários mal-intencionados podem fazer uso indevido de formulários desprotegidos para danificar o aplicativo

Validando formulário

Validação no cliente

- É uma verificação inicial importante para a boa experiência do usuário
- Ao detectar dados inválidos, o usuário pode corrigi-los imediatamente
- Evita que dados inválidos sejam enviados ao servidor gerando **atraso**
- No entanto, não deve ser considerada uma medida de segurança exaustiva!
 - É fácil de se contornar caso o usuário esteja mal-intencionados
- **A validação no servidor é imprescindível bem como no lado do cliente**



Validando formulário

Validação no cliente

- Existem duas maneiras de validar de formulários HTML no cliente:
 - Validação interna ou embutida do **HTML5** (*bulit-in*)
 - Em geral **não requer código JavaScript**
 - Apresentar melhor desempenho
 - Não é customizável
 - Usando o JavaScript
 - **Totalmente customizável, porém depende de código JavaScript**

Validando formulário

Validação no embutida

- Foi introduzida pelo HTML 5 em 2008 possibilitando a validação de boa parte dos casos de **validação sem depender de JavaScript**
- Feito usando **atributos de validação** em elementos de formulário

| Atributo | Descrição |
|------------------------------|--|
| required | Determina se um campo precisa ser preenchido antes do formulário ser enviado |
| minlength e maxlength | Determina o comprimento mínimo e máximo dos dados textuais (strings) |
| min e max | Determina os valores mínimo e máximo dos tipos de entrada numérica |
| type | Determina se os dados precisam ser de um tipo predefinido: número, e-mail, ... |
| pattern | Especifica uma expressão regular que precisa ser respeitada pelos dados inseridos |
| step | Considerando um step de n , um valor de entrada numérica só é válido se for um múltiplo de n |

Validando formulário

```
<form action="" method="POST">
  <label for="nome">Nome:</label>
  <input type="text" name="nome" required minlength="3" maxlength="50">

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

  <label for="telefone">Telefone:</label>
  <input type="tel" name="telefone" pattern="[0-9]{10,11}" placeholder="Apenas números">

  <label for="idade">Idade:</label>
  <input type="number" name="idade" required min="18" max="100">

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

Validando formulário

Validação no embutida

- Campos que seguem as regras de validação especificadas pelos atributos de validação são considerados válidos
 - Recebem a **pseudoclasse :valid**, podemos usá-las para customizá-los
- Quando um campo é inválido
 - Ele recebe a **pseudoclasse CSS :invalid**
 - Outras pseudoclasses podem ser atribuídas dependendo do erro
 - Ao tentar enviar os dados, **o navegador bloqueará o envio** e exibirá uma mensagem de erro

Validando formulário

Pseudoclasses associadas a validação

| Atributo | Descrição |
|------------------------|---|
| <code>:valid</code> | Presente em campos cujo valores são válidos |
| <code>:invalid</code> | Presente em campos cujo valores são inválidos |
| <code>:disabled</code> | Presente em campos com atributo "disable" |
| <code>:optional</code> | Presente em campos que não são obrigatórios |
| <code>:required</code> | Presente em campos que são obrigatórios |

Validando formulário

Validação usando JavaScript

- Necessária para realizar validações mais complexas
 - Ex: campo de senha igual ao campo de confirmação de senha
- Permite o controle maior da aparência dos erros
 - É possível customizar as mensagens nativas e suas aparências
- Faz o uso da **Constraint Validation API**
 - Engloba um conjunto de métodos e propriedades dos seguintes elementos de uma DOM:
 - *HTMLButtonElement, HTMLFieldSetElement, HTMLInputElement, HTMLOutputElement, HTMLSelectElement, e HTMLTextAreaElement*

Validando formulário

Constraint API

| Propriedade | Descrição |
|--------------------------------|--|
| <code>willValidate</code> | Retorna true se o campo deve ser validado durante a submissão do formulário |
| <code>validationMessage</code> | Retorna uma mensagem localizada descrevendo o erro de validação ocorrido. Se não houver restrições de validação ou o valor do campos as satisfazem, retorna uma string vazia. |
| <code>validity</code> | <p>Retorna um objeto <code>ValidityState</code> que possui uma série de flags booleanas que descrevem o estado de validação do campo.</p> <p>Ex:</p> <ul style="list-style-type: none">- <code>patternMismatch</code>, retorna true se o valor não corresponder ao padrão especificado (regex).- <code>tooLong</code>, retorna true se o campo ultrapassar o comprimento máximo especificado pelo atributo <code>maxlength</code>.- <code>valid</code>, retorna true se o campo atende a todas as suas restrições de validação.- Mais informações em: https://developer.mozilla.org/en-US/docs/Web/API/ValidityState |

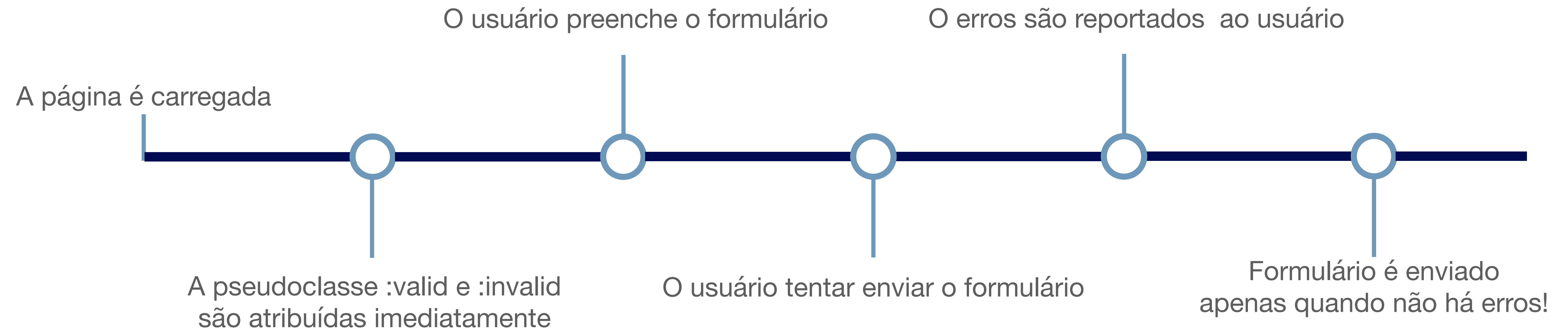
Validando formulário

Constraint API

| Métodos | Descrição |
|---|--|
| <code>checkValidity()</code> | Considerando um campo do formulário, retorna true caso possua um valor válido. Considerando um formulário, retorna true caso todos os valores dos seus campos sejam válidos |
| <code>reportValidity()</code> | Verifica se um campo é válido e em caso negativo, dispara a mensagem de erro. Ao ser chamado em uma instância de formulário, faz a checagem de todos os campos. |
| <code>setCustomValidity(message)</code> | Adiciona uma mensagem de erro personalizada ao campo. Permite a customização das mensagens pré-estabelecidas no navegador para a validação HTML padrão. Caso a mensagem seja vazia, o campo será considerado válido. |

Validando formulário

Constraint API - Fluxo geral

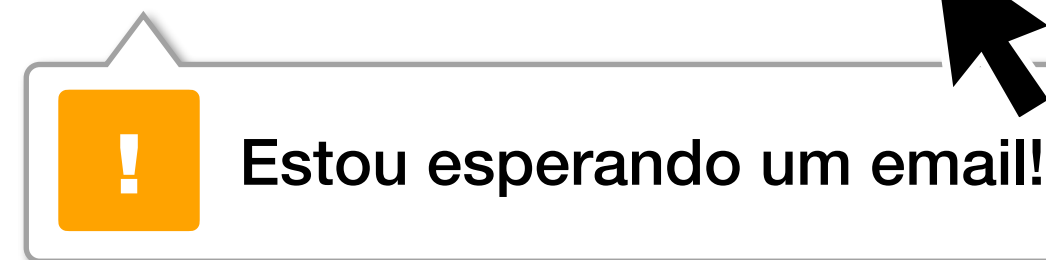


Validando formulário

Customizando a mensagem de erro

```
<form>
  <label for="mail">
    Você poderia me fornecer seu email:
  </label>
  <input type="email" id="mail" name="mail" />
  <button>Submit</button>
</form>
```

Você poderia me fornecer seu email:



```
const email = document.getElementById("mail");

email.addEventListener("input", (event) => {
  if (email.validity.typeMismatch) {
    email.setCustomValidity("Estou esperando um email!");
  } else {
    email.setCustomValidity("");
  }
});
```

Validando formulário

Validação totalmente customizada

```
<form novalidate>
  <p>
    <span>Você poderia me fornecer seu email:</span>
    <input type="email" id="mail" required >
    <span class="error"></span>
  </p>
  <button>Submit</button>
</form>
```

```
input:invalid {
  border-color: #900;
  background-color: #fdd;
}
.error {
  width: 100%;
  padding: 0.0em;
  color: white;
  background-color: #900;
  border-radius: 0 0 5px 5px;
  box-sizing: border-box;
}
.error.active {
  padding: 0.3em;
}
```

Você poderia me fornecer seu email:

Informe um email válido!

```
const form = document.querySelector("form");
const email = document.getElementById("mail");
const emailError = document.querySelector("#mail + span.error");

email.addEventListener("input", (event) => {
  if (email.validity.valid) {
    emailError.textContent = ""; emailError.className = "error";
  } else {
    showError();
  }
});
```

```
form.addEventListener("submit", (event) => {
  if (!email.validity.valid) {
    showError();
    event.preventDefault();
  }
});
```

```
function showError() {
  emailError.textContent = "Informe um email válido.";
  emailError.className = "error active";
}
```

Referências

- [Client-side form validation](#)
- [HTML Form Validation](#)
- [When and how to choose HTML for form validation](#)
- [Working with the HTML 5 Validation API](#)
- [CheckValidity e ReportValidity](#)

Por hoje é só