

JavaScript

Formulários e Validação

Acessando Elementos

Acesso a Elementos em Forms

Obtendo a referência de um form

- ▷ Inicialmente, vamos considerar o formulário abaixo. Você pode melhorá-lo se quiser.

```
<form name="formDados">
  <legend>Dados Pessoais</legend>
  <input name="nome" value="">
  <input name="email" value="">
  <select id="sexo">
    <option value="masculino">Masculino</option>
    <option value="feminino">Feminino</option>
  </select>
  <fieldset name="dados">
    <input type="radio" name="dependentes" value="0">
    <input type="radio" name="dependentes" value="1 a 2">
    <input type="radio" name="dependentes" value="3 ou mais">
  </fieldset>
</form>
```

Obtendo a referência de um form

- ▷ Os formulários são membros de uma coleção chamada *document.forms*.
- Elementos podem ser acessados nesta coleção através do:

- nome:

```
const formulario = document.forms.formDados;
```

- índice:

```
const formulario = document.forms[0];
```

Elementos de um form

- ▷ Os formulários possuem uma coleção chamada *elements* que retorna os elementos contidos no formulário:
 - Elementos podem ser acessados nesta coleção através do:

```
for (i = 0; i < formulario.elements.length; i++) {  
    console.log(formulario.elements[i]);  
}
```

- Um elemento específico pode ser acessado através do seu nome:

```
const inputNome = formulario.elements.nome;
```

- Outra forma:

```
const inputNome = formulario.nome;
```

fieldsets

- ▷ Os *fieldsets* também possuem a coleção *elements* que retorna a lista de elementos dentro do *fieldset*.

```
console.log('Elementos do fieldset:');  
for (i = 0; i < dados.elements.length; i++)  
{  
    console.log(dados.elements[i]);  
}
```

backreference

- ▷ É possível acessar o formulário de cada elemento através da propriedade *form*.

```
const nome = formDados.nome;  
console.log(nome.form);
```



```
▶ <form name="formDados"> ⚙
```

Alterando elementos

- ▷ É possível alterar valores dos *inputs* e outros componentes, conforme apresentando a seguir.

```
const nome = formDados.nome;  
const sexo = formDados.sexo;  
const dependentes = formDados.dependentes;  
  
nome.value = "João";  
  
dependentes[1].checked = true;  
  
sexo.value = 'feminino';  
//sexo.selectedIndex = 1;  
//document.getElementsByTagName('option')[1].selected = true;
```


Validação manual

- ▶ Se for necessária uma validação personalizada de um campo, os eventos *onblur* e *onfocus* podem ser utilizados.

```
<body>
  <form name="formDados">
    <input name="nome" value="">
    <div id="nomeErro"></div>
  </form>
  <style>
    .invalid { border-color: red; }
    #nomeErro { color: red }
  </style>
</body>
```

```
const nome = formDados.nome;

nome.onblur = function() {
  if (nome.value.length < 3) {
    nome.classList.add('invalid');
    nomeErro.innerHTML =
      'O nome precisa ter três caracteres.';
  }
};

nome.onfocus = function() {
  if (this.classList.contains('invalid')) {
    this.classList.remove('invalid');
    nomeErro.innerHTML = "";
  }
};
```

change vs input

- ▷ O evento *change* é disparado quando a alteração é finalizada (quando o elemento perde o foco). O evento *input* é disparado a cada alteração.

```
<body>
  <form name="formDados">
    <input name="nome" value="">
    <div id="nomeErro"></div>
  </form>
  <style>
    .invalid { border-color: red; }
    #nomeErro { color: red }
  </style>
</body>
```

```
const nome = formDados.nome;

nome.onChange = function() {
  console.log(`change: ${this.value}`);
};

nome.oninput = function() {
  console.log(`input: ${this.value}`);
};
```

submit

- ▷ É possível tratar a submissão do *form*. Inicialmente, considere o *HTML*:

```
<body>
  <form name="formDados">
    <input id="nome" name="nome" value="">
    <div id="nomeErro" class="erro"></div>
    <input id="email" name="email" value="">
    <div id="emailErro" class="erro"></div>
    <input type="submit" value="Enviar">
  </form>
  <style>
    .invalid { border-color: red; }
    .erro { color: red }
  </style>
</body>
```

submit

- ▶ Vamos criar funções para nos ajudar a apresentar o elemento com erro e para remover o erro do elemento.

```
const adicionarErro = (input, divMsg, msg) => {  
  input.classList.add('invalid');  
  divMsg.innerHTML = msg;  
};  
  
const removerErro = (input, divMsg) => {  
  if (input.classList.contains('invalid')) {  
    input.classList.remove('invalid');  
    divMsg.innerHTML = "";  
  }  
};
```

submit

- ▷ Função para validação:

```
const validar = (event) => {  
  let cancelarEnvio = false;  
  if (nome.value.length < 3) {  
    adicionarErro(nome, nomeErro, 'O nome precisa ter três caracteres.');
```

```
    cancelarEnvio = true;  
  }  
  if (!email.value.includes('@')) {  
    adicionarErro(email, emailErro, 'E-mail inválido!');
```

```
    cancelarEnvio = true;  
  }  
  if (cancelarEnvio) {  
    event.preventDefault();  
  }  
};
```

submit

- ▷ A validação do formulário será feita no evento *submit*.

```
nome.onfocus = () => removerErro(nome, nomeErro);  
email.onfocus = () => removerErro(email, emailErro);  
  
formDados.addEventListener('submit', validar);
```

Fim de material

Dúvidas?