04

Personalizando erros

Transcrição

Continuaremos melhorando o formulário, e para isto tentaremos validar o campo destinado ao CEP. Alguns destes campos, como para CEP, email e nome, estão com o atributo `required`, do HTML5, para serem campos de preenchimento obrigatório.

Já que queremos validar este CEP, o que acontecerá se usarmos a tecla "Enter", sem marcarmos nada? Ao testarmos, aparece um pequeno modal com a mensagem "Por favor, preencha este campo.". Então, em `index.html`, incluiremos `value` para cada um dos campos:

```
<div class="contatoCampo contatoCampo--sucesso">
    <label for="nome">Nome:</label>
    <input type="text" id="nome" class="contatoCampo--validouFoi" placeholder="Nome..." required val
</div>

<div class="contatoCampo contatoCampo--erro">
    <label for="email">Email:</label>
    <input type="email" id="email" class="contatoCampo--validouErro" placeholder="Email..." requirec
    <span class="contatoCampo-msg contatoCampo-msg--erro"> Você quis dizer <a href="#">gmail.com?</a
</div>

<!-- ... -->

<div class="contatoCampo">
    <label>Mensagem: </label>
    <textarea name="" cols="30" rows="10" required>lorem ipsum</textarea>
</div>
```

Com isso, poderemos testar apertando o botão "Enviar mensagem" na aplicação. Receberemos o aviso de que precisamos preencher o campo de CEP, que possui uma leve borda avermelhada em cima da borda preta. Vamos validá-lo, porém antes verificaremos como o NVDA se comporta diante destas mensagens de erro.

O leitor de telas ia ler algo, mas foi interrompido e disse "Por favor, preencha este campo". Pensando em acessibilidade, poderíamos customizar esta mensagem, no entanto, o HTML5 já resolve este problema para nós, por meio da validação com `required`.

> É importante validarmos o formulário não apenas na parte de front-end, mas em back-end também.

E por curiosidade, na parte do email, preencheremos o campo digitando um falso, como "asdasdsa" e, ao submetermos a mensagem, o NVDA lê "Por favor, digite um endereço de e-mail". Isto é, a mensagem será personalizada de acordo com o tipo do campo a ser preenchido.

Digamos que o designer, o *copy writer*, ou seu chefe, digam que esta mensagem de alerta está muito formal, e solicite algo mais amigável, que fale a linguagem do usuário - o que tem a ver com a experiência do usuário. Sendo assim, personalizaremos esta mensagem com JavaScript.

Criaremos um arquivo, que salvaremos na pasta "js" e chamaremos de `validacao-cep.js`. Nele, precisaremos do campo de CEP. Antes disso, voltaremos a `index.html` para incluirmos os `id`s e `for`s nos campos:

```
<div class="contatoCampo contatoCampo--erro">
    <label for="telefone">Telefone:</label>
    <input type="text" class="contatoCampo--validouErro" placeholder="(11)5571-2751" id="telefone">
    <span class="contatoCampo-msg contatoCampo-msg--erro">Faltou o DDD</span>
</div>

<div class="contatoCampo">
    <label for="cep">CEP: </label>
    <input id="cep" type="text" maxlength="8" placeholder="Exemplo: 04101-300" required>
</div>

<div class="contatoCampo">
    <label for="endereco">Endereço: </label>
    <input id="endereco" type="text" class="contatoCampo-campoDesabilitado" readonly value="Rua Verg
</div>

<div class="contatoCampo">
    <label for="bairro">Bairro: </label>
    <input id="bairro" type="text" class="contatoCampo-campoDesabilitado" readonly value="Vila Maria
</div>

<div class="contatoCampo">
    <label for="cidade">Cidade: </label>
    <input id="cidade" type="text" class="contatoCampo-campoDesabilitado" readonly value="São Paulo'
</div>

<div class="contatoCampo">
    <label for="estado">Estado: </label>
    <input id="estado" type="text" class="contatoCampo-campoDesabilitado" readonly value="SP">
</div>

<div class="contatoCampo">
    <label for="mensagem">Mensagem: </label>
    <textarea id="mensagem" name="" cols="30" rows="10" required>lorem ipsum</textarea>
</div>
```

Agora voltaremos ao CEP e, para ajudar no *debug*, poderíamos mandar um `alert()`, o que não seria tão interessante, então utilizaremos `console.log()`.

```
var campoCep = document.querySelector('#cep')

console.log(campoCep)
```

Salvaremos, iremos ao navegador e usaremos "Ctrl + Shift + I" para abrir o console, atualizaremos... E veremos que nada acontece, pois esquecemos de chamar este arquivo no HTML, o que faremos a seguir:

```
<script src="js/carousel.js"></script>
<script src="js/inert.js"></script>
```

```
<script src="js/dialog.js"></script>
<script src="js/validacao-cep.js"></script>
```

Vamos salvar o arquivo, atualizar o navegador, em que veremos que no console está sendo mandado o CEP como gostaríamos, portanto deletaremos a linha com `console.log(campoCep)`. Uma vez que a mensagem de erro não aparece quando submetemos ou clicamos no formulário, o próximo passo consiste em implementarmos uma função para quando ocorre algum erro de validação.

Em seguida, teremos que chamar `setCustomValidity` - mas de quem iremos customizar a mensagem? Do próprio `campoCep`, por isto usaremos `this`. A nova validação primeiramente precisará remover a mensagem anterior.

Para verificarmos se o campo realmente está inválido, usaremos o `if()`:

```
var campoCep = document.querySelector('#cep')

campoCep.oninvalid = function() {
    this.setCustomValidity('');

    if (!this.validity.valid) {
        this.setCustomValidity('Ops! Tem algo errado neste campo!')
    }
}
```

Salvaremos o código acima e testaremos no navegador. A mensagem de erro foi trocada corretamente! Para deixar o código JS mais sucinto ainda, o deixaremos assim:

```
document.querySelector('#cep').oninvalid = function () {
    this.setCustomValidity('');

    if (!this.validity.valid) {
        this.setCustomValidity('Ops! Tem algo errado neste campo!');
    }
}
```

Desta forma, com menos linhas de código conseguimos personalizar a mensagem de erro padrão do browser. Lembrando que é possível customizarmos ainda mais o visual da Apeperia, mas isto requer mais código JS, e como o foco deste curso é a acessibilidade, esta customização de mensagem de erro já supre o que precisamos.

Último detalhe: quando dá erro, queremos que isto seja indicado pelo círculo vermelho com xis branco no meio, no campo correspondente. E se acessarmos `index.html`, notaremos que é a classe `contatoCampo--erro:before` que puxa a imagem com a bolinha vermelha.

Precisaremos colocar a classe `contatoCampo--erro` quando o campo for inválido, e na `div`, isto é, no pai do `input`. Vamos abrir `validacao-cep.js` e digitar:

```
document.querySelector('#cep').oninvalid = function () {
    this.setCustomValidity('');

    if (!this.validity.valid) {
        this.setCustomValidity('Ops! Tem algo errado neste campo!');
```

```
                this.parentNode.classList.add('contatoCampo--erro')
        }
    }
```

Salvaremos e testaremos com o NVDA. O xis na bolinha vermelha aparece quando submetemos o formulário sem o preenchimento do campo de CEP, bem como é exibida a mensagem de erro.