02

Exibindo mensagens de erro

Transcrição

Além de impedirmos que um paciente com dados inválidos seja adicionado na tabela, queremos mostrar uma mensagem para o usuário informando qual problema tivemos na validação.

Atualmente, o código da função `validaPaciente(paciente)` do `form.js` está assim:

```
function validaPaciente(paciente) {

    if (validaPeso(paciente.peso)) {
        return true;
    } else {
        return false;
    }
}
```

Em vez de retornarmos verdadeiro ou falso (`true` ou `false`), poderemos **retornar uma mensagem de erro** caso a validação seja false e uma String vazia se a validação for true.

```
function validaPaciente(paciente) {

    if (validaPeso(paciente.peso)) {
        return "";
    } else {
        return "Peso é inválido";
    }
}
```

No momento de chamarmos a função validaPaciente(paciente), extrairemos o retorno da função para a variável `erro` . Se o tamanho da String for maior que `0` , significa que ocorreu algum erro.

```
var pacienteTr = montaTr(paciente);
var erro = validaPaciente(paciente);

if(erro.length > 0){
    return;
}
```

Exibiremos a mensagem de erro em cima do `form` . Para isto, faremos ajustes no `index.html` , adicionando uma tag `<span>` com o id `mensagem-erro` .

```
<section class="container">
    <h2 id="titulo-form">Adicionar novo paciente</h2>
    <span id="mensagem-erro"></span>
    <form id="form-adiciona">
        <div class="">
```

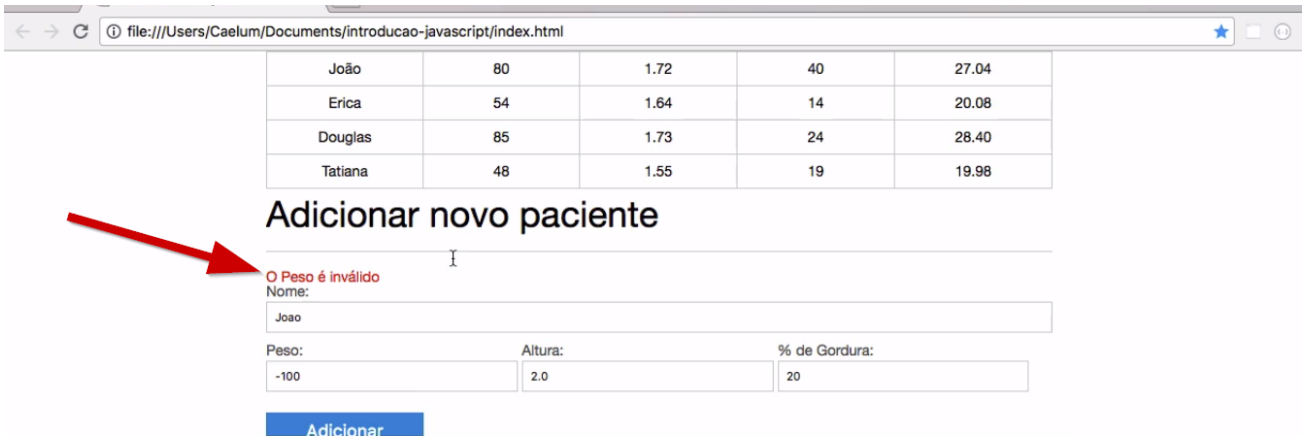
```
            <label for="nome">Nome:</label>
            <input id="peso" name="peso" type="text" placeholder="Digite o peso do seu paciente'
        </div>
//...
```

De volta ao `form.js`, se ocorrer algum erro na validação, selecionaremos o `span` que possui o id `#mensagem-erro`, e depois alteraremos o conteúdo de texto.

```
var erro = validaPaciente(paciente);

if (erro.length > 0) {
    var mensagemErro = document.querySelector("#mensagem-erro");
    mensagemErro.textContent = erro;
    return;
}
```

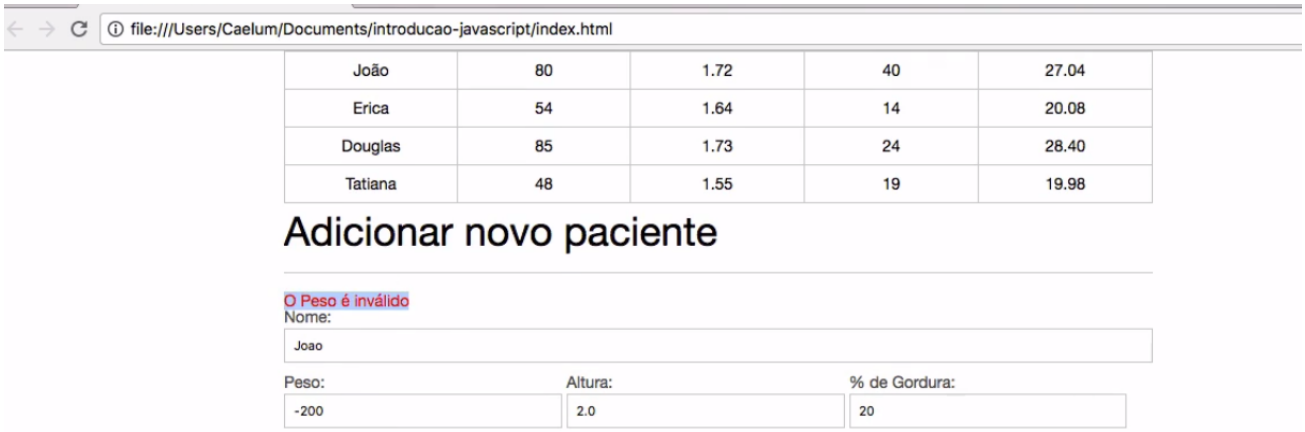
Vamos fazer um experimento e tentar preencher o formulário com dados inválidos. Após clicarmos no botão "Adicionar", será exibido uma mensagem acima do formulário.



Para deixá-la ainda mais em evidência, usaremos a cor da fonte vermelha. Ao fim de `index.css`, adicionaremos a id `#mensagem-erro`:

```
#mensagem-erro {
    color: red;
}
```

Agora, poderemos testar e ver que a mensagem está sendo exibida corretamente.



Validando a altura

Para validarmos a altura:

```
function validaPaciente(paciente) {

    if (validaPeso(paciente.peso)) {
        return "";
    } else {
        return "Peso é inválido";
    }

    if (validaAltura(paciente.altura)) {
        return "";
    } else {
        return "Altura é inválida";
    }

}
```

A função só retornará uma coisa de cada vez, e não duas strings concatenadas. Ou ela retorna que o peso é inválido, ou que a altura é inválida. Então, em vez de retornarmos uma única string, retornaremos um **array de strings**.

No momento das validações, poderemos adicioná-las em um array, previamente criado:

```
function validaPaciente(paciente) {

    var erros = [];

    if(validaPeso(paciente.peso)){
        return "";
    }else{
        return "O Peso é inválido";
    }

    if(validaAltura(paciente.altura)){
        return ""
    }else{
        return"A altura é inválida"
    }
```

```
}
```

Após declararmos o array de erros com `[]`, vamos fazer alterações na condição. Se o valor do peso tiver algum problema, pegaremos o array de erros e colocaremos algo dentro, usando o método `push()`. Faremos o mesmo para a altura. Ao final da função, retornamos o array:

```
function validaPaciente(paciente){
    var erros = [];

    if(validaPeso(paciente.peso)){
    } else {
        erros.push("Peso é inválido")
    }
    if(validaAltura(paciente.altura)){
    } else {
        erros.push("Altura é inválida!")
    }
    return erros;
}
```

No entanto, usaremos o operador de negação `!` para deixar o código mais legível. Apenas no caso do peso **não** ser válido, o `erro.push()` será adicionado. O mesmo será feito para a altura, e assim poderemos remover o `else`.

```
function validaPaciente(paciente) {

    var erros = [];

    if (!validaPeso(paciente.peso)) {
        erros.push("Peso é inválido");
    }

    if (!validaAltura(paciente.altura)) {
        erros.push("Altura é inválida");
    }

    return erros;
}
```

Um detalhe é que, por estarmos trabalhando com o `if` simples, poderemos remover as chaves (`{}`) e colocar o código em uma única linha. Ele continuará sendo compreensível para o JavaScript.

```
function validaPaciente(paciente) {

    var erros = [];

    if (!validaPeso(paciente.peso)) erros.push("Peso é inválido");

    if (!validaAltura(paciente.altura)) erros.push("Altura é inválida");

    return erros;
}
```

Mostramos essa opção para que você conhecesse essa possibilidade, mas deixaremos o código como estava anteriormente.

```
function validaPaciente(paciente){

    var erros = [];

    if (!validaPeso(paciente.peso)) {
        erros.push("Peso é inválido");
    }

    if (!validaAltura(paciente.altura)) {
        erros.push("Altura é inválida");
    }

    return erros;
}
```

Porém, o código estava adaptado para imprimir uma mensagem única. Nós gostaríamos que ele imprimisse uma lista de **erros**. Vamos ajustar o nome da variável em `form.js` :

```
var botaoAdicionar = document.querySelector("#adicionar-paciente");
botaoAdicionar.addEventListener("click", function(event){
    event.preventDefault();

    var form = document.querySelector("#form-adiciona");
    var paciente = obtemPacienteDoFormulario(form);
    var pacienteTr = montaTr(paciente);

    var erros = validaPaciente(paciente);

    if (erros.length > 0){
        var mensagemErro = document.querySelector("#mensagem-erro");
        mensagemErro.textContent = erros;
        return;
    }
```

Precisaremos adicionar uma string. Mas, por exemplo, se tivéssemos `10` erros, teríamos que adicionar a mesma quantidade de tags `<span>` s? Seria mais interessante termos uma **lista** no *HTML* e, para cada erro, adicionaríamos um item nessa lista. No `index.html` , vamos trocar o **`<span>`** por uma tag **`<ul>`** e, dentro, adicionar várias `<li>` s:

```
<section class="container">
    <h2 id="titulo-form">Adicionar novo paciente</h2>
    <ul id="mensagens-erro">
        <li></li>
        <li></li>
        <li></li>
        <li></li>
    </ul>
//...
```

A seguir, veremos como fazer isso usando o JavaScript.