01

Escutando eventos

Transcrição

Nossa tabela faz o cálculo do IMC de todos pacientes com os dois tipos de validações, inclusive sinalizando visualmente quando uma linha tem problemas, e assim facilitaremos a vida da Aparecida em relação ao cadastro de pacientes.

Atualmente, para adicionar um novo paciente, ela deve acessar o arquivo `index.html` e incluir uma nova `<tr>` na tabela, juntamente com as tags `<td>` , e modificar os dados da pessoa. Depois disso, uma nova linha surgirá na tabela.

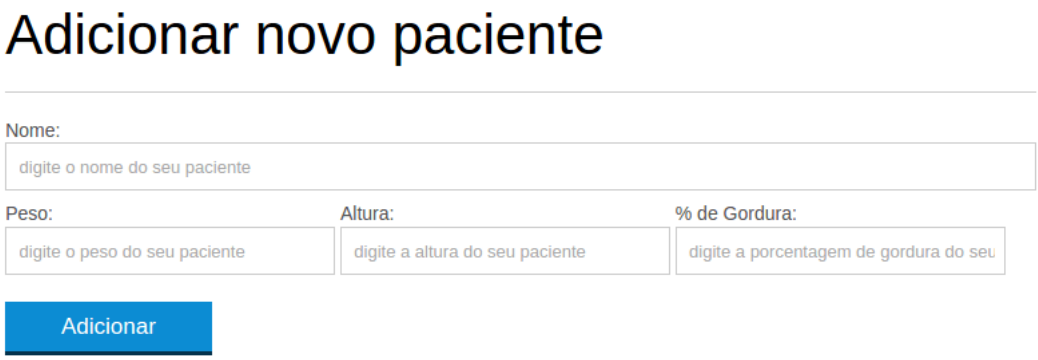
Ou seja, o HTML é alterado manualmente, o que não é prático. Seria muito simples se em vez de adicionarmos o paciente diretamente no HTML, houvesse um formulário no site, o qual ela pudesse preencher com os dados do paciente. E, após clicar-se no botão "Adicionar", o paciente seria adicionado à tabela. É exatamente isso que iremos implementar.

Como esse é um curso focado em JavaScript, disponibilizaremos o formulário pronto, basta adicioná-lo abaixo do fechamento da tag `<main>` , e antes de `<script>` :

```
<!-- ... -->
<section class="container">
    <h2 id="titulo-form">Adicionar novo paciente</h2>
    <form>
        <div class="grupo">
            <label for="nome">Nome:</label>
            <input id="nome" name="nome" type="text" placeholder="digite o nome do seu paciente'
        </div>
        <div class="grupo">
            <label for="peso">Peso:</label>
            <input id="peso" name="peso" type="text" placeholder="digite o peso do seu paciente'
        </div>
        <div class="grupo">
            <label for="altura">Altura:</label>
            <input id="altura" name="altura" type="text" placeholder="digite a altura do seu pa
        </div>
        <div class="grupo">
            <label for="gordura">% de Gordura:</label>
            <input id="gordura" type="text" placeholder="digite a porcentagem de gordura do seu
        </div>

        <button id="adicionar-paciente" class="botao bto-principal">Adicionar</button>
    </form>
</section>
```

O formulário possui os campos de preenchimento dos seguintes dados: "nome", "peso", "altura" e "% de gordura":



Quando a Aparecida preencher esses campos e clicar no botão "Adicionar", uma nova linha na tabela deverá ser criada, com a exibição dos dados recém adicionados através do formulário. Mas para fazê-lo, primeiramente devemos entender algumas funcionalidades do JavaScript.

Como faremos para detectar o exato momento em que o usuário clicará no botão e, então, executar uma ação?

A ação de "perceber o que o usuário está fazendo na página" é o que chamamos de **evento** do browser, que pode ser escutado com o JavaScript. Ações como clicar, duplo clique, *scrollar*, passar o mouse em cima, são exemplos dos tipos de interação que o usuário pode fazer com a página.

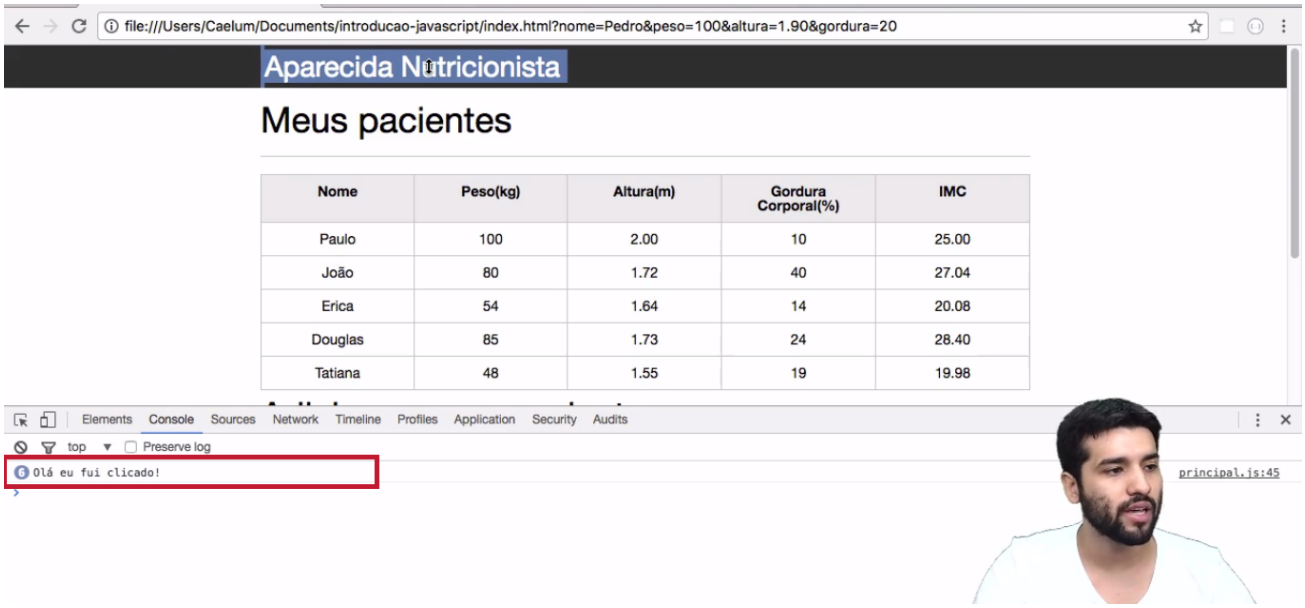
Para escutarmos um evento de clique, devemos especificar em qual elemento queremos fazê-lo. Se queremos identificar o momento do clique do botão, usaremos a variável `titulo` seguida de um "escutador de evento" que escutará as interações dos usuários, `addEventListener()`. Ela será responsável por adicionar o escutador de evento, como já diz o nome traduzido para o português. Em que tipo de evento estamos interessados? No caso, justamente no evento de `click`. Queremos que algo aconteça quando o usuário clicar no elemento. Indicaremos o que deve ser feito usando-se a função `mostraMensagem()`. Dentro dela, adicionaremos `console.log("Olá, eu fui clicado!")`.

```
titulo.addEventListener("click", mostraMensagem);

function mostraMensagem(){
    console.log("Olá eu fui clicado!");
}
```

Observe que na variável `titulo`, chamamos a função `mostraMensagem()` ao passá-la como segundo parâmetro do `addEventListener()`.

Em seguida, vamos atualizar a página do browser e verificar se as alterações estão funcionando. Clicaremos no título "Aparecida Nutricionista", que é o conteúdo da tag `<h1>`, e para cada clique a mensagem será exibida no console.



Nós conseguimos adicionar o escutador de eventos no título da página. Quando ele percebe que o usuário interagiu a partir de um clique, ele chama a função `mostraMensagem()`. Desta forma, atrelamos o evento com um determinado trecho de código HTML.

Uma outra abordagem que poderíamos adotar seria usarmos uma **função anônima**. Em vez de declararmos `mostraMensagem` como um parâmetro `addEventListener()`, poderíamos chamar diretamente `function()`:

```
titulo.addEventListener("click", function (){
    console.log("Olha só posso chamar uma função anônima.")
});
```

Observe que a função ainda possui as chaves (`{}`), mas não recebe um nome. Funções anônimas são bastante utilizadas, principalmente no caso de eventos. Além disso, chamamos uma função quando interagimos com um evento de clique, que só existirá quando o botão for clicado.

Resumindo, podemos trabalhar tanto com uma função **nomeada**, como a `mostraMensagem()`, ou uma função **anônima**, que será criada como segundo parâmetro da `addEventListener()`. Por enquanto, vamos praticar utilizando uma função anônima.

Porém, não queremos adicionar um evento de clique quando clicamos no título, e sim após o clique do usuário no botão "Adicionar" do formulário. A única ação que realizaremos a principio será exibir uma mensagem no console.

Teremos que selecionar o botão no JavaScript! No fim do trecho do código referente ao formulário, encontraremos a tag `<button>`, acima do fechamento de `<form>` em `index.html`:

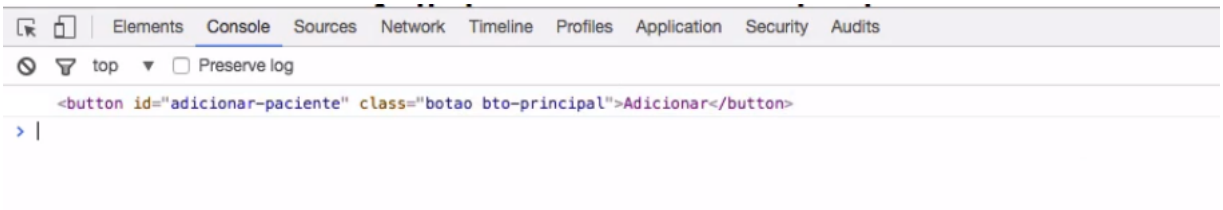
```
<!-- ... -->
    <div class="grupo">
        <label for="gordura">% de Gordura</label>
        <input for="gordura" name="gordura" type="text" placehoder="digite a porcentagem de gord
    </div>

    <button id="adicionar-paciente" class="botao bto-principal">Adicionar</button>
</form>
```

Em seguida, vamos adicionar o botão no arquivo `principal.js`. Abaixo do último `if` de validação dos dados, criaremos a variável `botaoAdicionar`, e o botão será levado pela função `document`. Usaremos a função `querySelector()` para buscar no HTML e, passaremos como parâmetro o seletor `#adicionar-paciente`:

```
var botaoAdicionar = document.querySelector("#adicionar-paciente");
console.log(botaoAdicionar)
```

Adicionaremos o `console.log()` para verificarmos se o código está funcionando. No browser, a mensagem será exibida ao atualizarmos a página, indicando que conseguimos pegar o botão como gostaríamos.



Nós também queremos escutar um evento de clique do botão, por isso, adicionaremos novamente `addEventListener()`.

```
var botaoAdicionar = document.querySelector("#adicionar-paciente");
botaoAdicionar.addEventListener("click", function(){
    console.log("Oi, cliquei no botão.");
});
```

No entanto, se tentarmos clicar no botão "Adicionar" do formulário, a mensagem não será exibida no console. Por que isso acontece?

Por padrão, sempre que clicamos em um botão contido em uma tag `<form>` do HTML, os seus dados serão enviados para outra página. Como não especificamos uma página para ser o alvo da tag `<form>`, a única ação realizada é a limpeza dos dados, e a página sendo recarregada em seguida. Ao fazermos isto, além do formulário, o console fica limpo também - por isso, não veremos a mensagem.

O evento de clique está sendo escutado corretamente, porém, como a página é recarregada rapidamente, não conseguiremos ver a mensagem impressa no console. Desta forma, não conseguiremos salvar os dados do paciente na tabela, nem exibir a mensagem.

O que devemos fazer para que a página não seja recarregada e tenha o comportamento esperado?