03

Delegando eventos

Transcrição

Para resolver o problema da remoção de pacientes, vamos nos aproveitar de uma característica do JavaScript chamada *Event Bubbling*, ou "borbulhamento" de eventos. Quando escutamos um evento no JavaScript, ele na verdade não acontece só no dono do evento (no nosso caso, na linha do paciente), ele acontece também no pai do paciente, no pai do pai do paciente, e assim vai subindo.

Na nossa estrutura, ao darmos um duplo clique na `<tr>` do paciente, o pai (`<tbody>`) também escuta o evento, assim com a tag `<table>` , até chegar no `<body>` .

```
<thead>
    <tr>
        <th>Nome</th>
        <th>Peso</th>
        <th>Altura</th>
        <th>Gordura Corporal(%)</th>
        <th>IMC</th>
    </tr>
</thead>
<tbody id="tabela-pacientes">
    <tr class="paciente" id="primeiro-paciente">
        <td class="info-nome">Paulo</td>
        <td class="info-peso">100</td>
        <td class="info-altura">-2.00</td>
        <td class="info-gordura">10</td>
        <td class="info-imc">0</td>
    </tr>
</body>
```

Podemos fazer uma analogia com um copo de refrigerante, já que quando colocamos o refrigerante em um copo, suas bolhas vão do fundo até estourar na boca do copo. O evento seria semelhante a essa bolha.

Vamos comentar o código antigo e fazer um teste, primeiramente, selecionando o `<tbody>` :

```
var pacientes = document.querySelectorAll(".paciente");

var tabela = document.querySelector("#tabela-pacientes");
```

Iremos colocá-lo para escutar um evento de clique, imprimindo uma mensagem no console ao ser clicado:

```
var pacientes = document.querySelectorAll(".paciente");

var tabela = document.querySelector("table");

tabela.addEventListener("click", function() {
    console.log("Fui clicado");
});
```

Ao voltarmos para a página, veremos que independente do ponto onde clicarmos no corpo da tabela, o evento será disparado, logo a mensagem será exibida no console. Independentemente se colocarmos o escutador de eventos em cada paciente ou no `<tbody>`, ele só precisará saber que um evento aconteceu ali dentro. O evento subirá como a bolha de refrigerante para a superfície do copo. Se colocarmos um `addEventListener` em cada paciente ou nas tags pai, tanto faz. Basta informá-lo de que houve um evento.

Descobrindo o elemento clicado

Para não termos que ficar escutando o evento a cada linha da tabela, e para as linhas adicionadas por meio do formulário, iremos delegar a responsabilidade de escutar os eventos para o pai de todos, no caso, a tag `<table>`. Faremos isso com `addEventListenter()` no `remover-paciente.js`:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick", function(event) {

});
```

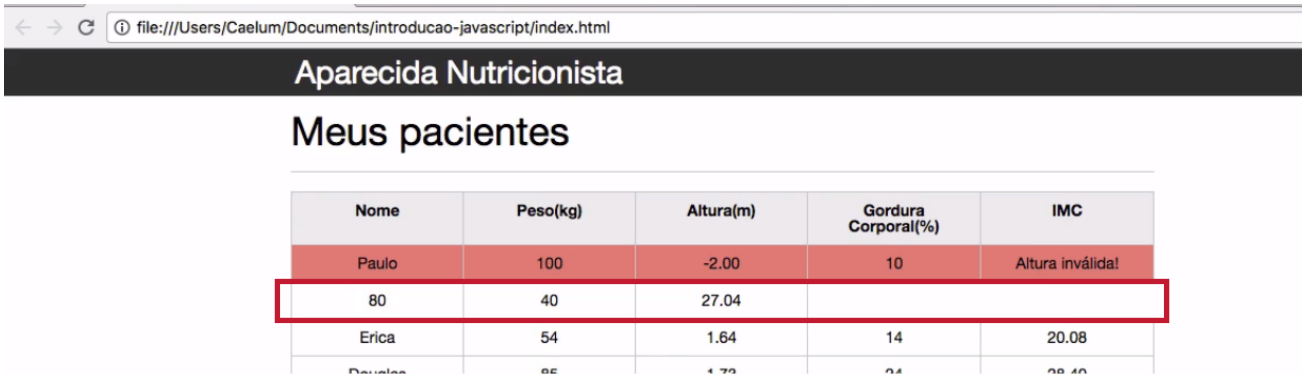
Dentro da função, perguntamos ao pai qual filho recebeu o clique, pois é ele que será removido. Desta vez não podemos utilizar o `this`, já que o dono do evento é a tabela, logo, ela acabará sendo removida. Para descobrirmos qual filho foi clicado, utilizaremos o `event` como parâmetro na função:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick", function(event) {
    console.log(event.target);

});
```

Ele irá nos dizer quem foi clicado, quem foi o alvo, pela propriedade `target`. Enquanto o `this` se refere ao dono do evento, o `event.target` será quem sofreu propriamente o evento. O próximo passo será remover o elemento após o duplo clique:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick", function(event) {
    event.target.remove();
});
```

Mas se testarmos, veremos que isso **quase** funciona. Ao clicarmos em algum campo da tabela, apenas o `<td>` clicado será removido. No entanto, queremos remover a linha completa, ou seja, a tag `<tr>`, pai do `<td>`.



Para selecionarmos o pai de um elemento, trabalharemos com a propriedade `parentNode`. A seguir selecionaremos quem foi clicado e removeremos o seu pai, uma tag `<tr>`:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick", function(event) {
    event.target.parentNode.remove();
});
```

Poderemos até separar em variáveis para deixar mais explícito:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick", function(event) {
    var alvoEvento = event.target;
    var paiDoAlvo = alvoEvento.parentNode; // TR = paciente = remover
    paiDoAlvo.remove();
});
```

Assim, mesmo que adicionarmos novos pacientes, no momento em que ele receber um duplo clique, o evento irá subir até chegar à tabela. Esta por sua vez, estará escutando. Desta forma, o paciente será removido.

Quando clicamos em qualquer filho, o evento consegue chegar até o pai (`table`). Essa estratégia é muito boa por economizarmos código, deixando-o mais sucinto. E é possível simplificarmos ainda mais o código:

```
var tabela = document.querySelector("table");

tabela.addEventListener("dblclick",function(event){
    event.target.parentNode.remove();
});
```

Nosso código funcionará igualmente.