alura

Atualizando o preço na remoção

Até o momento já temos um carrinho que nos permite remover o produto e aprendemos usar várias funcionalidades como: parent(), text(), click(), remove() e closest().

Nessa aula, o que precisamos fazer é alterar o valor total. Não tem muito segredo, é basicamente o que já estamos fazendo.

Em nosso javascript, vamos pegar o valor total com o jQuery e parsear para **float**:

```
var valorComoString = $('#valor-total').text();
var precoAtual = parseFloat(valorComoString);
```

Agora temos o valor total pronto para subtrair. Mas por qual valor subtrair? Precisamos pegar o preço do produto que estamos removendo. Quando clicamos no botão remover, o nosso **this** é o link `<a>` e nele não temos a quantia, a quantia está no elemento anterior em relação ao pai do **this**, que seria 1 `<td>` antes.

```
    //resto do código
    <td class="item-total">499.99</td>
    <td><a class="remove-item" href="">(remover)</a></td>
</tr>
```

Para poder chegar nesse elemento vamos usar o **parent()** para encontrar o elemento pai e em seguida o **prev()** que nos retornará o elemento anterior:

```
var itemTotal = self.parent().prev();
```

Chamamos o text() para pegar o valor dele.Lembrando que temos nosso **this** na var **self**.

```
var itemTotal = self.parent().prev();
var preco = parseFloat(itemTotal.text());

var precoFinal = precoAtual - preco;
```

Escrevemos esse valor de volta no valor total:

```
$('#valor-total').text(precoFinal);
```

Voltamos ao navegador e atualizamos a página para testar. Pronto, nossa funcionalidade está pegando, mas voltamos para o código javascript feio. No capítulo anterior usamos **closest()** para evitar o **parent()**, e agora estamos utilizando novamente com o **prev()**. Será que isso não irá causar problema ?

Se o designer do nosso site mudar a ordem das colunas, e clicarmos em remover, o campo de valor total irá aparecer com o valor NaN (not a number). Por quê? Porque o **prev()** não conseguiu encontrar o valor e ele devolve essa informação. Perceba que estamos muito atrelados a nosso design, precisamos melhorar isso.

No lugar do **parent()** novamente podemos substituir por **closest()** passando o `<td>`, e o **prev()** já não vai mais nos ajudar. No lugar dele poderiamos usar o **siblings()** para procurar entre os irmãos que tenham a classe **item-total**. Isso já nos ajudaria.

```
var itemTotal = self.closest("td").siblings(".item-total");
var preco = parseFloat(itemTotal.text());
```

No nosso caso podemos dar mais liberdade para o designer passando o `<tr>` no closest e depois pegar uma tag filho, neto, bisneto que tenha a classe **item-total**, usando o **find()**.

```
var itemTotal = self.closest("tr").find(".item-total");
var preco = parseFloat(itemTotal.text());
```

Atualizamos a página e vemos que está tudo funcionando. Voltando para o javascript, repare que temos muito código repetido, então podemos criar uma função chamada **atualizadaDados** e depois chamá-la:

```
<script>
    var atualizaDados = function(){
        var items = $(".item-total");
            var total = 0;
            for(var i=0; i < items.length; i++) {
                    var conteudo = $(items[i]).text();
                    var preco = parseFloat(conteudo);
                    total += preco;
                }
            $("#valor-total").text(total);
            $("#quantidade-de-itens").text(items.length);
    };

    var removeItem = function(event) {
        event.preventDefault();

        var self = $(this);
        self.closest("tr").remove();
        atualizaDados();
    };
```

```
    var aposInicializado = function() {

            $(".remove-item").click(removeItem);
            atualizaDados();
        };

        $(aposInicializado);
</script>
```

Com essas refatorações aprendemos o closest(), o find(), o siblings e além desses existem outros que podem nos ajudar. Teste novamente a nossa página e veja o resultado.

Ver video | Começar exercícios