08

Refatorando o Arquivo Carrinho js

Transcrição

Para darmos um toque final à aplicação, abriremos o arquivo JavaScript `carrinho.js` .

Iremos refatorar a classe `Carrinho` , trocando os nomes das viriáveis com a finalidade de deixar nossa aplicação mais clara. Vamos observá-la em seu atual estado:

```
class Carrinho {
    clickIncremento(btn) {
        let data = this.getData(btn);
        data.Quantidade++;
        this.postQuantidade(data);
    }

    clickDecremento(btn) {
        let data = this.getData(btn);
        data.Quantidade--;
        this.postQuantidade(data);
    }

    updateQuantidade(input) {
        let data = this.getData(input);
        this.postQuantidade(data);
    }

    getData(elemento) {
        var linhaDoItem = $(elemento).parents('[item-id]');
        var itemId = $(linhaDoItem).attr('item-id');
        var novaQtde = $(linhaDoItem).find('input').val();

        return {
            Id: itemId,
            Quantidade: novaQtde
        };
    }
```

Em `clickIncremento()` , temos como parâmetro `btn` , mas isso não confere muita clareza ao nosso código. Tocaremos esse parâmetro pelo nome `button` . Faremos o mesmo procedimento para `clickDecremento()` e nos outras linhas de código que apresentam `btn` .

```
class Carrinho {
    clickIncremento(button) {
        let data = this.getData(button);
        data.Quantidade++;
        this.postQuantidade(data);
    }

    clickDecremento(button) {
        let data = this.getData(button);
```

```
            data.Quantidade--;
            this.postQuantidade(data);
        }
```

Mais abaixo teremos a função `getData()`, que contém algumas variáveis. Uma delas é `novaQtde`, também presente em `return`.

```
    getData(elemento) {
            var linhaDoItem = $(elemento).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            return {
                Id: itemId,
                Quantidade: novaQtde
            };
        }
```

Substituiremos `novaQtde` por `novaQuantidade`.

```
    getData(elemento) {
            var linhaDoItem = $(elemento).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQuantidade = $(linhaDoItem).find('input').val();

            return {
                Id: itemId,
                Quantidade: novaQuantidade
            };
        }
```

Mais adiante encontraremos a chamada `ajax()`, e ao final veremos a linha com a instrução `debugger`, responsável por parar a aplicação quando o programa estive sendo depurado no browser.

```
    $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            }).done(function (response) {
                let itemPedido = response.itemPedido;
                let linhaDoItem = $('[item-id=' + itemPedido.id + ']')
                linhaDoItem.find('input').val(itemPedido.quantidade);
                linhaDoItem.find('[subtotal]').html((itemPedido.subtotal).duasCasas());
                let carrinhoViewModel = response.carrinhoViewModel;
                $('[numero-itens]').html('Total: ' + carrinhoViewModel.itens.length + ' itens');
                $('[total]').html((carrinhoViewModel.total).duasCasas());

                if (itemPedido.quantidade == 0) {
                    linhaDoItem.remove();
                }

                debugger;
```

```
            });
        }
    }
```

Contudo, essa instrução não é necessária quando formos enviar o projeto para a produção, por isso retiraremos a linha `debugger` do código.

```
<****!****>

            if (itemPedido.quantidade == 0) {
                linhaDoItem.remove();
            }
        });
    }
}
```

Dessa forma, concluímos a refatoração do nosso código.