

Bootstrap

Um Layout responsivo

Felipe Sobreira Cassimiro

Layout

Formulário de login

- Login:
- Senha:

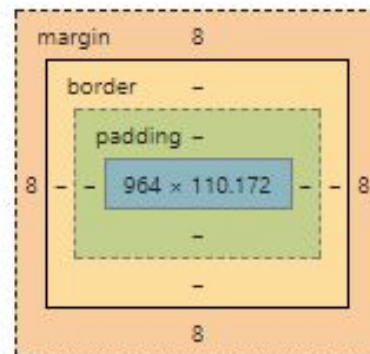
Login

html | 988 × 134.34

```
<form action="${ linkTo[LoginController].login }" method="post">  
<h2>Formulário de login</h2>
```

```
  <ul>  
    <li>  
      Login: <input type="text" name="Login">  
    </li>  
    <li>  
      Senha: <input type="text" name="senha">  
    </li>  
  </ul>
```

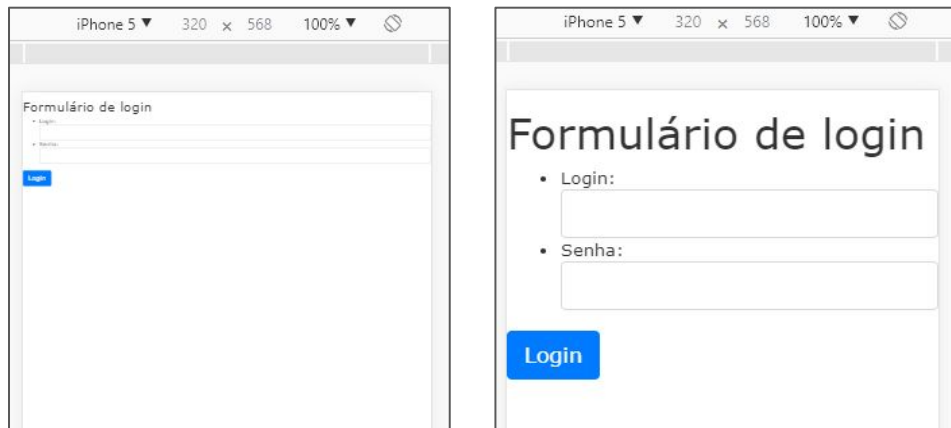
```
  <input type="submit" value="Login"/>  
</form>
```



Layout

- Viewport meta tag
 - largura
 - escala
- Responsividade
- Componentes

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```



```
<form class="form-group" action="{ linkTo[LoginController].login }" method="post">
<h2>Formulário de login</h2>

<ul>
  <li>
    Login: <input class="form-control" type="text" name="Login">
  </li>
  <li>
    Senha: <input class="form-control" type="text" name="senha">
  </li>
</ul>

<input class="btn btn-primary" type="submit" value="Login"/>
</form>
```

Layout

- Container
 - largura fixa
 - responsivo
 - centralizado
- 12 Colunas
 - row children
 - conteúdo
- Linhas
 - grupos
- Breakpoints

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				

Layout

- Exemplo de layout com breakpoints diferentes para os inputs de texto.
- Medium (md)
- auto (no prefix)

Responsive 473 x 376 100%

Formulário de login

Usuário:

Senha:

Login

Layout - Código Formulário

```
<div class="container">
  <fieldset class="form-group">
    <legend>Login</legend>
    <hr>
    <form action="{ linkTo[LoginController].login }" method="post">
      <div class="form-group row">
        <label for="login" class="col-2 col-form-label">Usuário:</label>
        <div class="col-10">
          <input class="form-control" type="text" name="login" id="login">
        </div>
      </div>

      <div class="form-group row">
        <label for="senha" class="col-2 col-form-label">Senha:</label>
        <div class="col-10">
          <input class="form-control" type="text" name="senha" id="senha">
        </div>
      </div>

      <div class="form-group row justify-content-around">
        <div class="col-8">
          <input class="btn btn-primary" type="submit" value="Login" id="btnLogin"/>
        </div>
      </div>
    </form>
  </fieldset>
</div>
```


Layout - Formulário

Login

Usuário:

Senha:

Login

JavaScript e jQuery

Uma comparação prática

Página HTML

Carrinho de compras

Itens para comprar

Preço

Quantidade

Nevermind - Nirvana

Em estoque

☐ Embalar para presente

R\$ 29,90

1

[Tirar do carrinho](#)

When Giants Walked the Earth: A Biography of Led Zeppelin - Mick Wall

Em estoque

☐ Embalar para presente

R\$ 59,90

1

[Tirar do carrinho](#)

The Big 4 Live from Sofia, Bulgaria - Metallica, Slayer, Megadeth and Anthrax

Em estoque

☐ Embalar para presente

R\$ 99,90

1

[Tirar do carrinho](#)

Total da compra

R\$ 189,70

JavaScript - Manipulando o valor total do carrinho

- Precisamos pegar o “texto” do HTML transformar em número, manipular esse valor, e depois convertê-lo para texto novamente e assim devolver à página.
- ex.: R\$ 25,00 - Retirar o símbolo da moeda, substituir a vírgula por ponto e usar a função `parseFloat()`;

```
function moneyTextToFloat(text) {  
    var cleanText = text.replace("R$ ", "").replace(",", ".");  
    return parseFloat(cleanText);  
}  
  
function floatToMoneyText(value) {  
    var text = Math.floor(value * 100);  
    text = "R$ " + text;  
    return text.substr(0, text.length - 2) + "," + text.substr(-2);  
}
```

JavaScript - Manipulando o valor total do carrinho

- Poderíamos criar uma função para retornar o valor total do carrinho convertido em número.

```
function readTotal() {  
    var total = document.getElementById("total");  
    return moneyTextToFloat(total.innerHTML);  
}
```

- Agora iremos determinar uma função para escrever o resultado na página.

```
function writeTotal(value) {  
    var total = document.getElementById("total");  
    total.innerHTML = floatToMoneyText(value);  
}
```

JavaScript - Calculando o subtotal dos itens

- Para calcular o subtotal de cada produto precisamos da quantidade e do valor.
- Cada linha na tabela possui um produto com essas propriedades.
- O método `getElementsByClassName` irá nos retornar um array com todos os produtos e podemos manipular esses dados.

```
<tr class="produto">
  <td>
    <div class="price">R$ 29,90</div>
  </td>
  <td>
    <input type="number" class="quantity">
  </td>
</tr>
<!-- um produto de 59,90 -->
<!-- um produto de 99,90 -->
```

JavaScript - Calculando o subtotal dos itens

- Diante das informações dos produtos já conhecidas, e das funções auxiliares que foram criadas, podemos implementar a função que calcula o total dos produtos.

```
function calculateTotalProducts() {  
    var produtos = document.getElementsByClassName("produto");  
  
    var totalProdutos = 0;  
  
    for(var pos = 0; pos < produtos.length; pos++) {  
        var priceElements = produtos[pos].  
            getElementsByClassName("price");  
        var priceText = priceElements[0].innerHTML;  
        var price = moneyTextToFloat(priceText);  
  
        var qtyElements = produtos[pos].  
            getElementsByClassName("quantity");  
        var qtyText = qtyElements[0].value;  
        var quantity = moneyTextToFloat(qtyText);  
  
        var subtotal = quantity * price;  
  
        totalProdutos += subtotal;  
    }  
  
    return totalProdutos;  
}
```

Evento - mudança na quantidade de produtos

- Ao mudarmos a quantidade de algum produto, queremos que o valor total seja atualizado, para isso iremos determinar um evento, que será uma função que será executada toda vez que a quantidade de qualquer um dos produtos seja atualizada.

```
function quantidadeMudou() {  
    writeTotal(calculateTotalProducts());  
}  
  
function onDocumentLoad() {  
    var textEdits = document.getElementsByClassName("quantity");  
  
    for(var i = 0; i < textEdits.length; i++) {  
        textEdits[i].onchange = quantidadeMudou;  
    }  
}
```

```
window.onload = onDocumentLoad;
```


jQuery - Simplificando as funções

- Uma expressão jQuery possui duas partes, o quê (selectors) vai ser manipulado e como isso vai acontecer.
- Iremos reescrever os métodos anteriores aplicando códigos jQuery, mas os mesmo métodos auxiliares para conversão serão utilizados.

```
function readTotal() {  
    var total = $("#total").text();  
    return moneyTextToFloat(total);  
}
```

```
function writeTotal(value) {  
    var text = floatToMoneyText(value);  
    $("#total").text(text);  
}
```

jQuery - Simplificando as funções

- Iremos agora reescrever o método que calcula o subtotal a partir da quantidade de produtos.

```
function calculateTotalProducts() {  
    var produtos = $(".produto");  
    var total = 0;  
  
    for(var pos = 0; pos < produtos.length; pos++) {  
        var $produto = $(produtos[pos]);  
        var quantity = moneyTextToFloat(  
            $produto.find(".quantity").val());  
        var price = moneyTextToFloat(  
            $produto.find(".price").text());  
        total += quantity * price;  
    }  
    return total;  
}
```

- Há outra forma de iterar sobre o array utilizando jQuery. Poderíamos utilizar o método each() do array, e passar uma função como parâmetro, que será executada para cada elemento do array.

jQuery - Simplificando as funções

- Para definirmos as funções de evento, não precisamos percorrer cada elemento necessário em um array.

```
$(".quantity").change(function() {  
    writeTotal(calculateTotalProducts());  
});
```

- Sem dúvida, nesta situação a simplicidade comparado ao código JavaScript é muito maior.
- Para que este último código seja executado, o jQuery tem um evento chamado ready que é executado assim que o documento é carregado.

```
$(document).ready(function() {  
    // faça alguma coisa  
});
```



```
$(function() {  
    $(".quantity").change(function() {  
        writeTotal(calculateTotalProducts());  
    });  
});
```

AJAX - Excluindo livros

DELETEDELETE http://localhost:8080/livraria-admin/livros 405 () jquery.min.js:4

```
@Delete
@Path("/livros/{isbn}")
public void exclui(String isbn) {
    this.estante.exclui(isbn);

    result.nothing();
}
```

```
$(document).ready(function() {
    $("#livros .remove").on("click", function(event) {
        event.preventDefault();

        // Para visualização do cliente
        var livro = $(this).closest(".livro");

        // AJAX para executar a requisição
        $.ajax({
            url : $(this).attr("href"),
            type : 'POST',
            data: { _method: "DELETE" }
        }).done(function(data, textStatus, jqXHR) {
            // executada em caso de sucesso
            livro.fadeOut();
        }).fail(function(jqXHR, textStatus, errorThrown) {
            // executada em caso de erro
            alert("O Livro não foi removido!");
        }).always(function() {
            // executada assim que a requisição termina,
            // seja com sucesso ou com erro
        });
    });
});
```

AJAX - Excluindo livros

```
<h3>Lista de livros</h3>


<ul id="livros">
  <c:forEach items="${livros}" var="livro">
    <li class="livro">

      <!--  -->
       -

      ${livro.titulo} - ${livro.descricao} -
      <a href="${linkTo[LivrosController].edita[livro.isbn]}">Modificar</a> -
      <a href="${linkTo[LivrosController].exclui[livro.isbn]}" class="remove">Excluir</a> -
      <a href="${linkTo[LivrosController].serialize[livro.isbn]}">Serializar</a>

    </li>
  </c:forEach>
</ul>
```

Lista de livros

-  - Arquitetura de Software - Uma abordagem prática - Modificar - **Excluir** - Serializar
-  - Ruby - uma linguagem show - Modificar - Excluir - Serializar

Referências

- CAVALCANTI, L. VRaptor: Desenvolvimento ágil para web com java.
- Bootstrap 4, Documentação. Disponível em: <https://getbootstrap.com/docs>.
- BALDUÍNO, P. Dominando JavaScript com jQuery.