

CAPÍTULO 11

Bootstrap e formulários HTML5

"O trabalho é a melhor das regularidades e a pior das intermitências"

— Victor Marie Hugo

11.1 – BOOTSTRAP E FRAMEWORKS DE CSS

Uma tendência em alta no mundo front-end é o uso de frameworks CSS com estilos base para nossa página. Ao invés de começar todo projeto do zero, criando todo estilo na mão, existem frameworks que já trazem toda uma base construída de onde partiremos nossa aplicação.

Existem muitas opções mas o **Twitter Bootstrap** talvez seja o de maior notoriedade. Ele foi criado pelo pessoal do Twitter a partir de código que eles já usavam internamente. Foi liberado como opensource e ganhou muitos adeptos. O projeto cresceu bastante em maturidade e importância no mercado a ponto de se desvincular do Twitter e ser apenas o **Bootstrap**.

<http://getbootstrap.com>

O Bootstrap traz uma série de recursos:

- Reset CSS
- Estilo visual base pra maioria das tags
- Ícones
- Grids prontos pra uso
- Componentes CSS
- Plugins JavaScript
- Tudo responsivo e mobile-first

Como o próprio nome diz, é uma forma de começar o projeto logo com um design e recursos base sem perder tempo com design no início.

11.2 – ESTILO E COMPONENTES BASE

Para usar o Bootstrap, apenas incluímos seu CSS na página:

```
<link rel="stylesheet" href="css/bootstrap.css">
```

Só isso já nos traz uma série de benefícios. Um reset é aplicado, e nossas tags ganham estilo e tipografia base. Isso quer dizer que podemos usar tags como um H1 ou um P agora e elas terão um estilo característico do Bootstrap.

Além disso, ganhamos **muitas classes** com componentes adicionais que podemos aplicar na página. São várias opções. Por exemplo, pra criar um título com uma frase de abertura em destaque, usamos o jumbotron:

```
<div class="jumbotron">
  <div class="container">
    <h1>Ótima escolha!</h1>
    <p>Obrigado por comprar na Mirror Fashion.</p>
  </div>
</div>
```

No exercício a seguir vamos usar vários outros componentes.

Nova editora Casa do Código com livros de uma forma diferente



Editoras tradicionais pouco ligam para ebooks e novas tecnologias. Não conhecem programação para revisar os livros tecnicamente a fundo. Não têm anos de experiência em didáticas com cursos.

Conheça a **Casa do Código**, uma editora diferente, com curadoria da **Caelum** e obsessão por livros de qualidade a preços justos.

[Casa do Código, ebook com preço de ebook.](#)

11.3 – A PÁGINA DE CHECKOUT DA MIRROR FASHION

Neste capítulo, vamos desenvolver a página de **checkout** da Mirror Fashion.

Após escolher o produto desejado, o usuário cai nessa página para efetivar a compra.

Nossa loja foi otimizada pra compra direta, sem carrinho de compras. O cliente escolhe o produto e compra direto, com um clique. Só precisamos coletar os dados de dele, do pagamento e da entrega.

O foco dessa nova página então é a coleta de informações para efetivação da compra. Um grande formulário complexo com os campos necessários. Vamos usar o Bootstrap para desenvolver essa página com mais facilidade e rapidez.

Ótima escolha!

Obrigado por comprar na Mirror Fashion! Preencha seus dados para efetivar a compra.

Sua compra



Produto
Cardigan Thelure Basic
Cor
Azul
Tamanho
38

Dados pessoais

Nome completo

E-mail

@ ama 1@exemplo.com

CPF

000.000.000-00

☒ Quero receber spam da Mirror Fashion

Cartão de crédito

Número - CVV

0000 0000 0000 0000 - 000

Bandeira

MasterCard

Validade

Confirmar pedido

Figura 11.1: Site visto no Desktop

E, como aprendemos antes, vamos desenvolver tudo **mobile-first**. Nesse momento, portanto, ainda não teremos o design Desktop mostrado acima, mas uma versão mobile em uma coluna. Veremos como adaptar a versão Desktop com Bootstrap depois.

Ótima escolha!

Obrigado por comprar na Mirror Fashion! Preencha seus dados para efetivar a compra.

Sua compra

Produto
Cardigan Thelure Basic

Cor
azul

Tamanho
38

Dados pessoais

Nome completo

E-mail

@email@exemplo.com

CPF

000.000.000-00

☒ Quero receber spam da Mirror Fashion

Cartão de crédito

Número - CVV

0000 0000 0000 0000 - 000

Bandeira

MasterCard

Validade

Confirmar pedido

Figura 11.2: Site visto no Mobile

11.4 – EXERCÍCIO OPCIONAL: INÍCIO DO CHECKOUT SEM PHP

1. Se você não fez os capítulos com PHP, crie agora sua página **checkout.html** com HTML simples pra poder seguir esse capítulo. Não há dependência obrigatória de PHP no curso.

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Checkout Mirror Fashion</title>
  <meta name="viewport" content="width=device-width">
</head>
<body>

  <h1>Ótima escolha!</h1>
  <p>Obrigado por comprar na Mirror Fashion!
  Preencha seus dados para efetivar a compra.</p>

  <h2>Sua compra</h2>
```

```

<dl>
  <dt>Produto</dt>
  <dd>Fuzzy Cardigan</dd>

  <dt>Cor</dt>
  <dd>verde</dd>

  <dt>Tamanho</dt>
  <dd>40</dd>

  <dt>Preço</dt>
  <dd>R$ 129,00</dd>
</dl>

</body>
</html>

```

Teste a página simples no navegador.

11.5 – EXERCÍCIOS: PÁGINA DE CHECKOUT

1. Abra a página de checkout no navegador e veja que está com o estilo padrão do navegador.

O primeiro passo é incluirmos o arquivo CSS do bootstrap na nossa página. Você vai ver uma mudança sutil no estilo da página, principalmente nos aspectos tipográficos.

Coloque no <head> da página de checkout o CSS do bootstrap:

```
<link rel="stylesheet" href="css/bootstrap.css">
```

Teste novamente a página.

2. O primeiro componente pronto do bootstrap que vamos usar é o **jumbotron**. É basicamente a abertura do site, contendo sua chamada principal. Para usá-lo basta criar um div com a classe jumbotron.

Envolva as chamadas de abertura que **já tínhamos** com h1 e p em dois <div>. O primeiro div contém class="jumbotron" e o segundo, class="container".

```

<div class="jumbotron">
  <div class="container">

    <!-- h1 e p que já tínhamos -->
    <h1>Ótima escolha!</h1>
    <p>Obrigado....</p>

```

```
</div>
</div>
```

Abra a página e note que um estilo diferente aparece. Teste redimensionar o navegador e veja que o tamanho da fonte e espaçamento do componente se ajustam automaticamente. O Bootstrap usa responsive design automaticamente em seus componentes.

Para saber mais do jumbotron: <http://getbootstrap.com/components/#jumbotron>

3. Use um outro componente do Bootstrap, o **panel** para organizar a seção que mostramos as informações da compra do cliente. Cuidado com o exercício, com os nomes das classes, que confundem bastante.

Adapte o HTML do H2 "Sua compra" e do DL que temos para se adequar ao componente de panel:

```
<div class="panel panel-default">
  <div class="panel-heading">
    <h2 class="panel-title">Sua compra</h2>
  </div>
  <div class="panel-body">

    <!-- ... aqui vai o <dl> que já temos hoje ... -->

  </div>
</div>
```

Repare como os nomes das classes, apesar de serem muitos, fazem sentido para isolar cada parte do painel.

Teste novamente a página no navegador e veja o resultado. Temos um painel arredondado com título em destaque no topo.

Para saber mais sobre painéis do Bootstrap:

<http://getbootstrap.com/components/#panels>

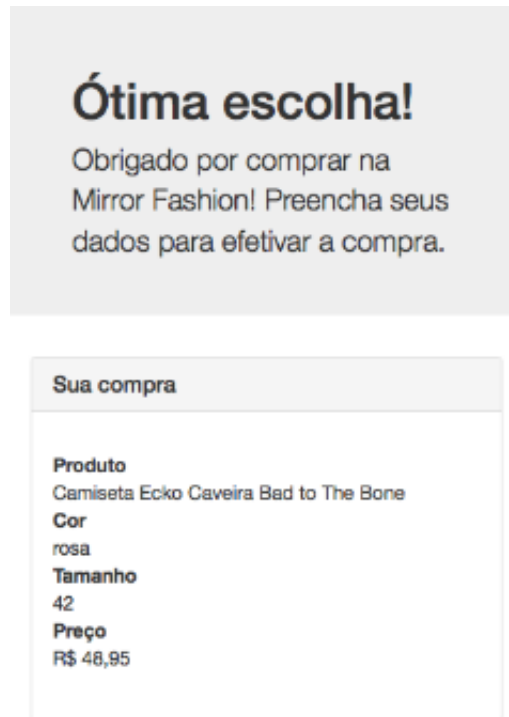
4. Repare no exercício anterior do jumbotron que o `div container` é responsável por centralizar e dar espaçamento na tela. Muito parecido aliás com o container que havíamos criado antes em nosso projeto, mas agora é uma classe do Bootstrap.

Crie um outro `div container` pra conter o `panel` que acabamos de criar e veja como ele fica melhor posicionado no centro da tela.

```
<div class="container">
```

```
<!-- ... panel aqui ... -->
```

```
</div>
```



5. Dentro do `panel-body`, logo no topo, acima da lista de definições `<d1>`, vamos colocar uma foto do produto escolhido e na cor escolhida.

O segredo é gerar o endereço da imagem levando em conta os parâmetros do ID e da cor:

```
img/produtos/foto<?= $_POST["id"] ?>-<?= $_POST["cor"] ?>.png
```

Com Bootstrap, podemos ainda acrescentar algumas classes nessa imagem para obter resultados interessantes. A classe `img-responsive` faz a imagem ficar flexível e nunca estourar o tamanho do pai. E a classe `img-thumbnail` faz a imagem ficar centralizada com uma borda de destaque.

Adicione a imagem do produto logo acima da lista `<d1>` dentro do `div panel-body`:

```
-<?= $_POST["cor"] ?>.png"
      class="img-thumbnail img-responsive">
```

Teste novamente a página.

Imagem sem PHP

Para o exercício de Bootstrap em si, você pode usar uma imagem estática sem envolver o PHP para gerar o endereço:

```

```

6. (opcional) No painel, troque a classe `panel-default` pela classe **`panel-success`**. Teste e veja o resultado. Consulte outros valores na documentação:

<http://getbootstrap.com/components/#panels>

Já conhece os cursos online Alura?



A **Alura** oferece dezenas de **cursos online** em sua plataforma exclusiva de ensino que favorece o aprendizado com a **qualidade** reconhecida da Caelum. Você pode escolher um curso nas áreas de Java, Ruby, Web, Mobile, .NET e outros, com uma **assinatura** que dá acesso a todos os cursos.

[Conheça os cursos online Alura.](#)

11.6 – FORMULÁRIOS A FUNDO

Quando solicitamos que o usuário informe seu nome, seu endereço de email, se ele quer receber uma newsletter, qualquer informação, precisamos utilizar os elementos corretos. Para isso, vamos conhecer os formulários HTML: a tag `<form>`.

Já usamos alguns antes. Agora vamos ver a fundo seus desdobramentos.

Atributos do Form

```
<form action="/efetivar.php" method="POST">
</form>
```

O formulário exemplificado anteriormente apresenta o atributo obrigatório **action**. O valor desse atributo é o endereço para onde as informações do formulário serão enviadas, e esse valor depende inteiramente de como é feita a aplicação que receberá essas informações no lado do servidor.

O segundo atributo, `method`, especifica o método do HTTP pelo qual essa informação será transmitida. O valor `post`, de maneira simplista, significa que queremos **inserir** as informações desse formulário, salvá-la de alguma maneira.

Outro valor possível para esse atributo, o `get`, é utilizado quando queremos obter alguma coisa a partir das informações que estamos transmitindo, por exemplo, um formulário de busca.

Componentes

Porém, neste exemplo, não temos nenhum elemento para capturar as informações. Na verdade, somente a marcação da tag `<form>` não mostra nenhum elemento visível no navegador. Vamos supor que precisemos de uma informação como o nome do visitante do nosso site para guardar em um banco de dados. Vamos adicionar alguns elementos ao nosso formulário anterior:

```
<form action="/efetivar.php" method="POST">

  <label for="nome">Nome:</label>
  <input type="text" name="nome" id="nome">

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

Label

Adicionamos a marcação do elemento `<label>`. Esse elemento é uma tag de conteúdo, e seu texto é exibido de maneira comum dentro do nosso formulário, a única diferença é que essa marcação faz uma ligação com outro elemento qualquer em nosso formulário. Note que nosso `label` tem o atributo `for`, que recebe o valor `nome`.

Quando clicamos com o mouse sobre o texto marcado com a tag `label`, o elemento que tem o atributo `id` com o mesmo valor que o atributo `for` do `label` é selecionado para que possamos interagir com ele. No exemplo, esse elemento vinculado ao `label` é um campo de texto que declaramos com a tag `input`.

Essa marcação `<label>` é de extrema importância para a usabilidade e acessibilidade dos nossos formulários.

Input

A maioria dos elementos que utilizamos nos formulários para capturar informações dos usuários são da tag `<input>`. No exemplo anterior, utilizamos duas variações dessa tag.

Os tipos diferentes de `inputs` são determinados pelo valor do seu atributo `type`, e no exemplo nós utilizamos dois muito comuns. A seguir, vamos detalhar os

tipos aceitos para essa tag.

text

```
<input type="text" name="nome_usuario">
```

Provavelmente o tipo mais comum de input, o que tem o atributo **type="text"**, é utilizado quando queremos que o usuário envie uma informação textual simples, pois esse elemento não permite a entrada de quebras de linha.

Ao enviarmos o formulário, a informação digitada pelo usuário é acessível no lado do servidor por meio do atributo name, utilizado para identificar cada informação contida nos parâmetros da requisição. Para ter acesso à informação digitada quando tratamos o formulário com algum tipo de script, para validar o conteúdo por exemplo, é necessário obter o conteúdo da propriedade value do objeto no DOM.

password

O input que recebe o atributo **type="password"** é similar ao anterior, do tipo text, com a diferença de que ele não exibe exatamente o texto digitado pelo usuário, e sim uma série de símbolos * ou outro, dependendo do navegador e sistema operacional.

```
<input type="password" name="senha">
```

checkbox

O elemento input do tipo **checkbox** exibe uma caixa para marcação, é muito utilizado quando temos uma opção que pode ser marcada como sim ou não, por exemplo "Aceito os termos de contrato do usuário", ou "Manter a sessão ativa" em formulários de login.

Apesar de muito utilizado com o valor true, é possível determinar qualquer valor para o checkbox.

```
<input id="contrato" name="contrato" type="checkbox" value="sim">  
<label for="contrato">Aceito os termos do contrato.</label>
```

radio

```
<p>  
  <input type="radio" name="idade" id="idade5" value="5">  
  <label for="idade5">Menos de 5 anos</label>  
</p>  
<p>
```

```
<input type="radio" name="idade" id="idade10" value="10">
<label for="idade10">Menos de 10 anos</label>
</p>
<p>
  <input type="radio" name="idade" id="idade15" value="15">
  <label for="idade15">Menos de 15 anos</label>
</p>
<p>
  <input type="radio" name="idade" id="idade20" value="20">
  <label for="idade20">Menos de 20 anos</label>
</p>
```

Quando desejamos que o usuário escolha somente uma entre uma série de opções, podemos utilizar elementos input do tipo **radio**. Quando há mais de um elemento desse tipo com o mesmo valor no atributo name, somente um pode ser selecionado.

button

```
<input type="button" name="mostra_dialogo" value="Clique aqui!">
```

O elemento input com o atributo **type="button"** renderiza um botão dentro do formulário, mas esse botão não tem nenhuma função direta nele e é comumente utilizado para disparar eventos para a execução de scripts.

O texto do botão é determinado pelo valor do atributo value.

submit

```
<input type="submit" name="enviar" value="Enviar">
```

O elemento input com o atributo **type="submit"** é similar ao botão, mas quando acionado esse elemento inicia a chamada que envia as informações do formulário para o endereço indicado no atributo action do <form>.

image

```
<input type="image" name="botao" src="images/enviar.png" width="20"
height="18">
```

É possível substituir o botão de envio do formulário por uma imagem, possibilitando criar um visual mais atrativo para o formulário.

reset

```
<input type="reset" name="reset" value="Limpar">
```

O input com **type="reset"** elimina os valores entrados anteriormente nos

elementos de um formulário, permitindo que o usuário limpe o mesmo.

<input> e <button>

A tag <input> dos tipos **button**, **submit** e **reset** pode ser substituída pela tag <button>. Neste caso, o texto do botão passa a ser indicado como conteúdo da tag. Ainda assim é necessário especificar o valor do atributo **type**, inclusive se ele for **button**:

```
<button type="button" name="enviar">Clique aqui</button>
```

file

```
<input type="file" name="anexo">
```

Quando é necessário que o usuário envie um arquivo para a aplicação no lado do servidor é necessário o uso do input do tipo **file**. Para o correto envio dos arquivos, muitas vezes também é necessário adicionar o atributo **enctype="multipart/form-data"** na tag <form>.

hidden

```
<input type="hidden" name="codigo" value="abc012xyz789">
```

Muitas vezes precisamos enviar e receber informações que não têm utilidade direta para o usuário e, portanto, não devem ser exibidos no formulário. Para essa finalidade, existe o input do tipo **hidden**, que somente carrega em si um valor.

Textarea

Quando desejamos que o usuário insira uma quantidade grande de informações textuais, incluindo quebras de linha, é necessário o uso da tag **textarea**

```
<textarea name="texto"></textarea>
```

Select, Optgroup e Option

Quando desejamos que o usuário selecione entre diversas opções, com a possibilidade de flexibilizar a maneira com que ele interage com o componente do formulário, podemos utilizar a tag <select>.

```
<select name="cidades">
```

```
<option value="bsb">Brasília</option>
<option value="rj">Rio de Janeiro</option>
<option value="sp">São Paulo</option>
</select>
```

Em sua configuração padrão, o controle select exhibe o que conhecemos como **menu drop-down**, permitindo que somente uma das opções possa ser selecionada. Caso seja adicionado o atributo `multiple`, é possível selecionar mais de uma opção da mesma maneira que selecionamos diversos arquivos no explorador do sistema operacional.

```
<select multiple name="cidades">
  <option value="bsb">Brasília</option>
  <option value="rj">Rio de Janeiro</option>
  <option value="sp">São Paulo</option>
</select>
```

Caso necessário, dependendo do número de opções apresentadas ao usuário, pode ser interessante agrupá-las:

```
<select name="bairro">
  <optgroup label="Brasília">
    <option value="asan_bsb">Asa Norte</option>
    <option value="asas_bsb">Asa Sul</option>
  </optgroup>
  <optgroup label="São Paulo">
    <option value="vlmariana_sp">Vila Mariana</option>
    <option value="centro_sp">Centro</option>
  </optgroup>
  <optgroup label="Rio de Janeiro">
    <option value="botafogo_rj">Botafogo</option>
    <option value="centro_rj">Centro</option>
  </optgroup>
</select>
```

11.7 – NOVOS COMPONENTES DO HTML5

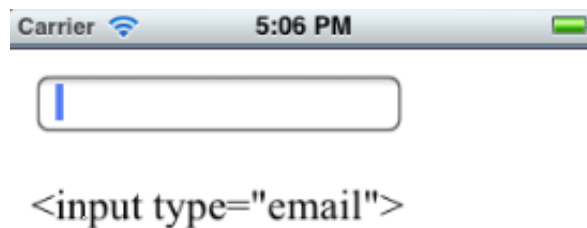
Com a nova especificação do HTML, é possível utilizar uma série de novos componentes que facilitam bastante o desenvolvimento de formulários. Até o momento em que essa apostila foi escrita, muitos componentes são incompatíveis com os navegadores, mas mostram, na maioria dos casos, um campo de texto permitindo a entrada de qualquer tipo de informação.

A maioria dos novos tipos de componentes de formulário foram criados para permitir que o navegador adapte o método de entrada para o mais adequado em cada um dos casos. Alguns desses componentes já são compatíveis com navegadores de dispositivos móveis.

email

```
<input type="email" name="email">
```

O input do tipo **email** permite que os dispositivos móveis, principalmente, exibam um teclado adaptado para facilitar esse tipo de entrada. Por exemplo, o iPhone exibe um teclado com o caractere @ e com as opções de domínio .com.



number

```
<input type="number" max="100" step="5">
```

O input do tipo **number**, além de exibir um teclado numérico em dispositivos móveis, nos navegadores modernos exibe um controle que permite incrementar ou decrementar o valor do campo clicando em uma seta para cima ou para baixo.

Além dessa diferença visual, é possível determinar valores mínimos, máximos e se há uma escala de valores válidos. No exemplo anterior, o elemento deve aceitar números múltiplos de 5 com o limite do valor "100".

url

```
<input type="url" name="endereco">
```

O elemento input com tipo **url** permite que os dispositivos exibam um teclado

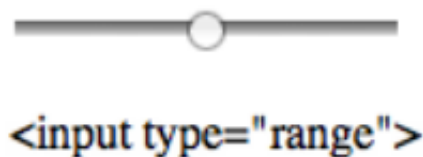
como, no exemplo do iPhone, opções como `www` e `.com`.

range

```
<input type="range" name="volume">
```

O elemento `input` do tipo **range** exibe um controle deslizante nos navegadores modernos, permitindo uma interação mais agradável quando precisamos de um valor numérico em escala. O controle guarda um valor numérico em seu atributo `value`. Assim como o `input` do tipo `number`, é possível especificar um valor mínimo, máximo e uma escala.

A renderização mais comum desse controle, em um Chrome:

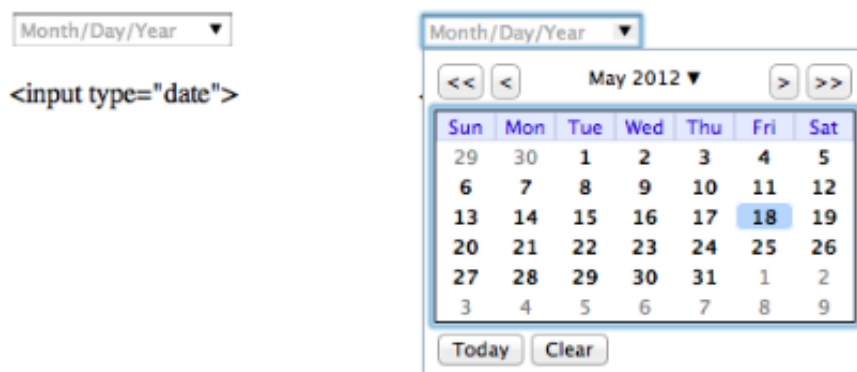


date, month, week, time, datetime e datetime-local

```
<input type="date" name="validade">
```

Os controles de "date picker" são feitos para coletar uma informação de data ou hora. São várias as possibilidades de formato de data ou hora necessárias. No navegador Opera, quando utilizado esse tipo de controle, o usuário pode selecionar uma data a partir de um calendário. É possível especificar datas mínima e máxima.

Em geral, os navegadores devem oferecer alguma funcionalidade de escolha de datas para o usuário, como o Chrome:



Ou o iPhone:

May 18, 1973

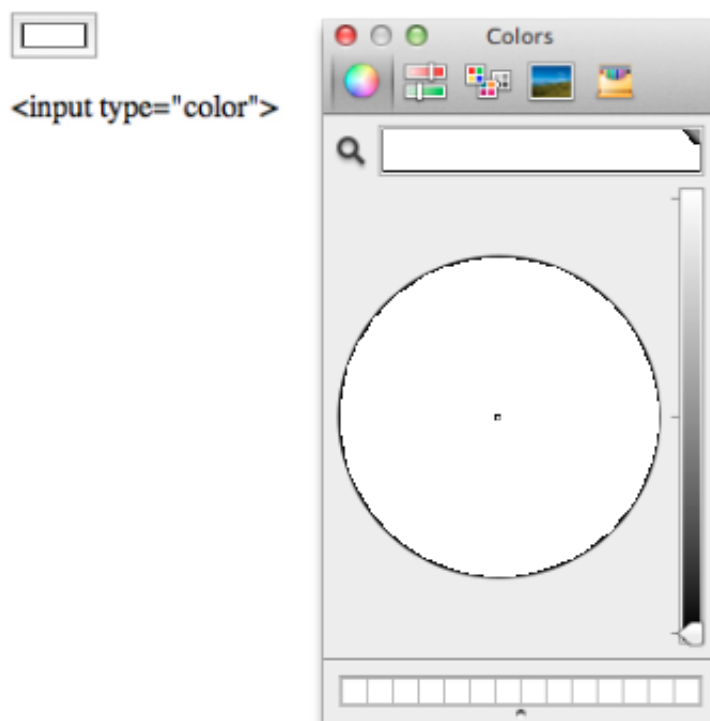
`<input type="date">`



color

`<input type="color" name="cor_olhos">`

O elemento input do tipo **color** permite que seja exibido um "color picker" para o preenchimento do seu valor. O Chrome no Mac, por exemplo, exibe o color picker padrão do sistema:



search

```
<input type="search" results="10">
```

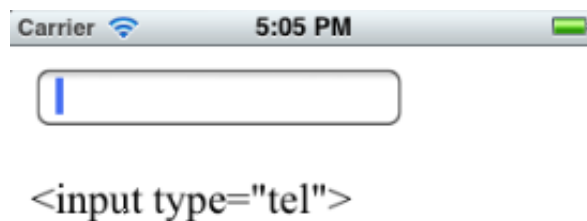
O input do tipo **search** exibe um campo específico para busca. O atributo "results" determina quantas últimas buscas serão armazenadas e lembradas, além de exibir uma lupa dentro do campo (Safari e Chrome).

tel

```
<input type="tel" name="telefone">
```

O input do tipo **tel** foi especificado para coletar um número de telefone.

Em dispositivos com teclados virtuais como smartphones e tablets, é comum o teclado ser adaptado para exibir apenas opções relevantes à entrada de números telefônicos, como no iPhone:



11.8 – NOVOS ATRIBUTOS HTML5 EM ELEMENTOS DE FORMULÁRIO

Na especificação do HTML5 estão definidos novos atributos para os elementos de formulário, visando implementar algumas necessidades comuns que antes não eram possíveis de serem atendidas puramente com a marcação do formulário.

autofocus

Sua presença indica que aquele campo deve iniciar com foco quando a página for carregada. O usuário já pode começar a digitar algo sem nenhum clique.

```
<input name="nome" autofocus>
```

placeholder

```
<input type="text" name="nome" placeholder="Insira seu nome">
```

O atributo **placeholder** exibe o texto contido em seu valor dentro do elemento do formulário caso o seu valor seja vazio.

autocomplete, list e datalist

É possível implementar uma funcionalidade de sugestão de valores com mais facilidade.

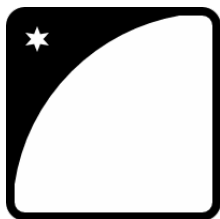
```
<input type="text" list="cidades" autocomplete="on">
<datalist id="cidades">
  <option value="Brasília">
  <option value="Rio de Janeiro">
  <option value="São Paulo">
</datalist>
```

A implementação de **autocomplete** sem o atributo **list** no campo, ligando-o a um **datalist**, vai utilizar os últimos valores utilizados em outros campos ou em outros formulários, dando prioridade a valores adicionados em inputs com o mesmo valor no atributo name.

Existem diversas maneiras de utilizar os componentes de formulários, tanto os novos do HTML5 como os já existentes. Mesmo com a oportunidade de inovar e criar uma interação totalmente diferente do usuário com um formulário, é importante manter o mesmo método que adotamos anteriormente. A marcação correta do formulário facilita muito o uso dele em diversos navegadores e em outros tipos de clientes também, como por exemplo navegadores especiais para deficientes visuais.

Você não está nessa página a toa

Você chegou aqui porque a Caelum é referência nacional em cursos de Java, Ruby, Agile, Mobile, Web e .NET.



Faça curso com quem escreveu essa apostila.

[Consulte as vantagens do curso *Desenvolvimento Web com HTML, CSS e JavaScript*.](#)

11.9 – ÍCONES COM GLYPHICONS

O Bootstrap traz um conjunto de ícones prontos para uso chamado de **Glyphicons**. Esses ícones são disponibilizados através de uma **fonte de texto customizada**. Eles desenharam os ícones e exportaram como uma fonte normal.

Para usarmos com Bootstrap é bem simples:

```
<span class="glyphicon glyphicon-thumbs-up"></span>
```

São 180 ícones no total, das mais diversas razões. Basta olhar o nome da documentação e usar na página.

Available glyphs

Includes 180 glyphs in font format from the Glyphicon Halfings set. [Glyphicons](#) Halfings are normally not available for free, but their creator has made them available for Bootstrap free of cost. As a thank you, we only ask that you to include a link back to [Glyphicons](#) whenever possible.

















































 .glyphicon .glyphicon-adjust	 .glyphicon .glyphicon-align-center	 .glyphicon .glyphicon-align-justify	 .glyphicon .glyphicon-align-left	 .glyphicon .glyphicon-align-right	 .glyphicon .glyphicon-arrow-down	 .glyphicon .glyphicon-arrow-left	 .glyphicon .glyphicon-arrow-right
 .glyphicon .glyphicon-arrow-up	 .glyphicon .glyphicon-asterisk	 .glyphicon .glyphicon-backward	 .glyphicon .glyphicon-ban-circle	 .glyphicon .glyphicon-barcode	 .glyphicon .glyphicon-bell	 .glyphicon .glyphicon-bold	 .glyphicon .glyphicon-book
 .glyphicon .glyphicon-bookmark	 .glyphicon .glyphicon-briefcase	 .glyphicon .glyphicon-bullhorn	 .glyphicon .glyphicon-calendar	 .glyphicon .glyphicon-camera	 .glyphicon .glyphicon-certificate	 .glyphicon .glyphicon-check	 .glyphicon .glyphicon-chevron-down
 .glyphicon .glyphicon-chevron-left	 .glyphicon .glyphicon-chevron-right	 .glyphicon .glyphicon-chevron-up	 .glyphicon .glyphicon-circle-arrow-down	 .glyphicon .glyphicon-circle-arrow-left	 .glyphicon .glyphicon-circle-arrow-right	 .glyphicon .glyphicon-circle-arrow-up	 .glyphicon .glyphicon-cloud
 .glyphicon .glyphicon-cloud-download	 .glyphicon .glyphicon-cloud-upload	 .glyphicon .glyphicon-cog	 .glyphicon .glyphicon-collapse-down	 .glyphicon .glyphicon-collapse-up	 .glyphicon .glyphicon-comment	 .glyphicon .glyphicon-compressed	 .glyphicon .glyphicon-copyright-mark
 .glyphicon .glyphicon-credit-card	 .glyphicon .glyphicon-facsimile	 .glyphicon .glyphicon-film	 .glyphicon .glyphicon-flag	 .glyphicon .glyphicon-floppy-disk	 .glyphicon .glyphicon-globe	 .glyphicon .glyphicon-heart	 .glyphicon .glyphicon-home

Figura 11.9: Alguns dos ícones

<http://getbootstrap.com/components/#glyphicons>

A vantagem de se usar fontes para ícones é que o desenho fica escalável, como uma letra. Ele não perde qualidade em nenhum tamanho ou resolução por ser vetorial. E, assim como uma letra, podemos aplicar efeitos de texto como sombras e cores.

A desvantagem é que cada ícone só pode ter um path no desenho e uma única cor. Não é possível usar ícones complexos com fontes.

11.10 – EXERCÍCIOS: FORMULÁRIOS

1. O formulário de compra possui campos para o cliente digitar informações pessoais e informações sobre o pagamento. Para melhor organização, vamos separar os campos em dois fieldsets.

Vamos criar o `<form>` logo depois do *panel*, e ainda **dentro** do container. Neste form, crie os dois fieldsets usando `<legend>` para identificar cada um. No final, um botão cuidará do envio dos dados (vamos usar um `btn-primary` do Bootstrap).

```
<form>
  <fieldset>
    <legend>Dados pessoais</legend>

  </fieldset>

  <fieldset>
    <legend>Cartão de crédito</legend>

  </fieldset>

  <button type="submit" class="btn btn-primary">
    Confirmar Pedido
  </button>
</form>
```

2. O primeiro fieldset, dos Dados Pessoais, deve conter os campos Nome, Email, CPF e um checkbox para o usuário optar ou não por receber spam.

Implemente os campos **dentro do primeiro fieldset**. Use as classes do Bootstrap para formulários. Use também um `input email` do HTML5.

```
<fieldset>
```

```

<legend>Dados pessoais</legend>

<div class="form-group">
  <label for="nome">Nome completo</label>
  <input type="text" class="form-control" id="nome" name="nome">
</div>

<div class="form-group">
  <label for="email">Email</label>
  <input type="email" class="form-control" id="email" name="email">
</div>

<div class="form-group">
  <label for="cpf">CPF</label>
  <input type="text" class="form-control" id="cpf" name="cpf">
</div>

<div class="checkbox">
  <label>
    <input type="checkbox" value="sim" name="spam" checked>
    Quero receber spam da Mirror Fashion
  </label>
</div>
</fieldset>

```

Repare que cada campo possui um input e um label. Para agrupá-los, usamos um div form-group do Bootstrap. Cada input deve ter uma classe form-control.

Teste a página e observe o estilo padrão que ganhamos apenas por usar o Bootstrap.

3. O fieldset de dados do cartão tem três campos: um com código do cartão, outro com a bandeira do cartão e outro com data de validade. Neste último, usaremos o *input month* do HTML5.

Implemente os campos dentro do segundo fieldset:

```

<fieldset>
  <legend>Cartão de crédito</legend>

  <div class="form-group">
    <label for="numero-cartao">Número - CVV</label>
    <input type="text" class="form-control"
      id="numero-cartao" name="numero-cartao">
  </div>

  <div class="form-group">
    <label for="bandeira-cartao">Bandeira</label>
    <select name="bandeira-cartao" id="bandeira-cartao"
      class="form-control">
      <option value="master">MasterCard</option>
      <option value="visa">VISA</option>
      <option value="amex">American Express</option>
    </select>
  </div>
</fieldset>

```

```

    </select>
  </div>

  <div class="form-group">
    <label for="validade-cartao">Validade</label>
    <input type="month" class="form-control"
      id="validade-cartao" name="validade-cartao">
  </div>
</fieldset>

```

4. **Adicione** o atributo **placeholder** do HTML5 nos campos email e CPF com dicas de preenchimento:

```

<input type="email" class="form-control" id="email" name="email"
  placeholder="email@exemplo.com">

...

<input type="text" class="form-control" id="cpf" name="cpf"
  placeholder="000.000.000-00">

```

Adicione o atributo **autofocus** do HTML5 no input nome:

```

<input type="text" class="form-control" id="nome" name="nome" autofocus>

```

5. Vamos incentivar o clique no botão de pedido com um ícone além do texto. Use os **glyphicons** do Bootstrap pra isso. Dentro do botão, apenas **adicione a linha** que declara o ícone:

```

<button type="submit" class="btn btn-primary">
  <span class="glyphicon glyphicon-thumbs-up"></span>
  Confirmar Pedido
</button>

```

Para saber mais sobre os ícones do Bootstrap:

<http://getbootstrap.com/components/#glyphicons>

6. Use outras classes do Bootstrap para ajustar mais detalhes. No botão, **adicione** a classe **btn-lg** para deixar o botão maior.

Ainda no botão, **acrescente** também a classe **pull-right** para deixá-lo alinhado à direita.

Veja mais opções de botões com Bootstrap: <http://getbootstrap.com/css/#buttons>

7. (opcional) O Bootstrap tem outros recursos para formulários, como os **input groups**. Teste trocando o código do campo email para isso:

```

<div class="form-group">
  <label for="email">Email</label>

```

```
<div class="input-group">
  <span class="input-group-addon">@</span>
  <input type="email" class="form-control"
    id="email" name="email">
</div>
</div>
```

Implemente também em outros campos, inclusive usando ícones do glyphsicons.

Veja mais opções do Bootstrap para formulários:

<http://getbootstrap.com/css/#forms>

11.11 – VALIDAÇÃO HTML5

Entre as muitas novidades de formulários que vimos no HTML5, há ainda toda uma parte de validação de dados com restrições expressas diretamente no código HTML.

required

Podemos indicar na marcação do formulário quando um campo é de preenchimento obrigatório.

```
<input type="text" name="nome" required>
```

Esse atributo permite uma validação *fraca* no lado do cliente.

pattern

Conseguimos também especificar um formato requerido através do atributo **pattern**, adicionando uma expressão regular como valor:

```
<input type="text" pattern="^\@\w{2,}" name="usuario_twitter">
```

O atributo **pattern** também permite uma validação *fraca* do campo.

Validação no CSS

A maioria dos novos componentes de formulário e os atributos que funcionam como validadores de campos na verdade somente aplicam uma pseudo-classe específica no campo que não está atendendo ao padrão ou requisito especificado.

Essa pseudo-classe é a **:invalid**, e pode ser utilizada para dar um retorno visual imediato caso o usuário não esteja atendendo aos requisitos dos campos do

formulário.

```
:invalid {  
  outline: 1px solid #cc0000;  
}
```

Essa validação é **fraca** pois de maneira direta não é possível impedir que o usuário envie as informações do formulário, mesmo que incompletas ou incorretas. É possível porém alterar o botão de submit e deixá-lo desabilitado caso seja possível selecionar algum elemento por essa pseudo-classe no formulário. Essa verificação e alteração do elemento submit pode ser feita por JavaScript e jQuery de maneira simples.

Seus livros de tecnologia parecem do século passado?



Conheça a **Casa do Código**, uma **nova** editora, com autores de destaque no mercado, foco em **ebooks** (PDF, epub, mobi), preços **imbatíveis** e assuntos **atuais**.

Com a curadoria da **Caelum** e excelentes autores, é uma abordagem **diferente** para livros de tecnologia no Brasil.

Conheça os títulos e a nova proposta, você vai gostar.

[Casa do Código, livros para o programador.](#)

11.12 – PARA SABER MAIS: CONTROLANDO AS VALIDAÇÕES HTML5

A ideia da nova validação do HTML5 é permitir que os navegadores já possuam uma forma simples de prover validações sem que os desenvolvedores precisem recorrer a complicadas bibliotecas JavaScript (algo comum em muitas páginas).

No entanto, muitas vezes, as opções padrão do navegador não são exatamente o que precisamos, e queremos mudar o comportamento da validação ou executar validações personalizadas e diferentes.

Podemos, então, usando JavaScript, desabilitar a validação padrão e fazer a nossa própria:

```
document.querySelector('form input').oninvalid = function(evt) {  
  // cancela comportamento padrão do browser  
  evt.preventDefault();  
}
```



```
// checa validade e mostra alert
if (!this.validity.valid) {
    alert("Nome obrigatório!");
}
};
```

Isso nos permite trocar, por exemplo, todo o visual e forma de apresentação dos erros. E, o melhor, caso o usuário esteja com JavaScript desabilitado, será executada a validação padrão sem problemas. Um ótimo fallback. (nas soluções tradicionais de validação dom jQuery, por exemplo, tudo se perde quando o usuário desabilita JavaScript).

Outra forma de desabilitar a validação, afetando o formulário inteiro, é colocando o atributo novalidate na tag <form>.

Além de desabilitar completamente a validação do navegador, podemos apenas trocar a mensagem de erro mas ainda usar o mecanismo e design padrão:

```
document.querySelector('input[type=email]').oninvalid = function() {

    // remove mensagens de erro antigas
    this.setCustomValidity("");

    // reexecuta validação
    if (!this.validity.valid) {

        // se inválido, coloca mensagem de erro
        this.setCustomValidity("Email inválido");
    }
};
```

Suporte nos navegadores

A validação HTML5 está implementada no Chrome, Firefox, Safari, Opera e IE10. Dos navegadores móveis, temos suporte em Chrome, Firefox, Opera, IE e Blackberry:

<http://caniuse.com/form-validation>

Se você quiser suportar navegadores mais antigos, recomendamos o uso de um polyfill:

<https://github.com/aFarkas/webshim>

11.13 – EXERCÍCIOS: VALIDAÇÃO COM HTML5

1. **Adicione** o atributo **required** nos campos Nome e CPF.

Teste submeter o formulário sem preencher esse campos.

2. Algumas validações já são implícitas apenas por usarmos o input type correto. Por exemplo, tente submeter o formulário preenchendo o Email com um valor inválido (com dois @ por exemplo).

3. Podemos estilizar no CSS quando um campo está inválido:

```
.form-control:invalid {  
    border: 1px solid #cc0000;  
}
```

4. (opcional avançado) Implemente uma mensagem customizada para erro de email de inválido usando a API JavaScript das validações HTML5.

```
document.querySelector('input[type=email]').oninvalid = function() {  
  
    // remove mensagens de erro antigas  
    this.setCustomValidity("");  
  
    // reexecuta validação  
    if (!this.validity.valid) {  
  
        // se inválido, coloca mensagem de erro  
        this.setCustomValidity("Email inválido");  
    }  
};
```

11.14 – GRID RESPONSIVO DO BOOTSTRAP

Um das dificuldades mais comuns de um projeto front-end é o posicionamento de elementos, sobretudo em designs multi coluna. A solução mais comum é uso de grids, uma ideia antiga que veio dos próprios designers.

Divide-se a tela em colunas e vamos encaixando os elementos dentro desse grid.

Todo framework CSS moderno traz um grid pronto para utilização. Todo código CSS necessário para correto posicionamento já foi escrito e só precisamos **usar as classes certas**. O Bootstrap tem um grid pronto e várias classes para usarmos.

O grid do Bootstrap trabalha com a ideia de **12 colunas** e podemos escrever nosso código escolhendo quantas colunas ocupar. Alguns exemplos:

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8									.col-md-4			
.col-md-4				.col-md-4				.col-md-4				
.col-md-6						.col-md-6						

Essas classes de coluna são as que definem o tamanho de cada elemento na página com base nas 12 partes do grid padrão. Em código:

```
<div class="row">
  <div class="col-md-4">
    ...
  </div>

  <div class="col-md-8">
    ...
  </div>
</div>
```

No código anterior, deixamos o primeiro DIV ocupando 4/12 da tela e o outro, 8/12. Repare que, para o grid funcionar, ao redor das colunas usamos um div com class **row**. Ele é necessário.

Podemos ainda criar grids dentro de grid, sempre obedecendo a divisão de 12 colunas em cada. Por exemplo:

```
<div class="row">
  <div class="col-md-4">
    ...
  </div>

  <div class="col-md-8">
    <div class="row">
      <div class="col-md-6">
        ...
      </div>
      <div class="col-md-6">
        ...
      </div>
    </div>
  </div>
</div>
```

Esse exemplo criou um segundo grid dentro da coluna da direita do primeiro. Nesse segundo grid há duas colunas ocupando metade cada uma (6/12). Mas como

um grid está dentro do outro, na prática, ele vai ocupar metade do tamanho do div que tem 8/12 de tamanho.

Responsivo

Um dos pontos mais interessantes dos grids é que eles são **responsivos**. Isso quer dizer que podemos aplicar diferentes layouts de colunas no nosso código ao mesmo tempo e cada um deles vai valer só em determinada situação.

Nos códigos anteriores, por exemplo, usamos classes como **col-md-6**. O **md** nessa classe significa que vamos ocupar 6 colunas do grid *apenas em telas maiores que 992px de largura*. Em telas menores, automaticamente nosso grid será de uma coluna só.

E, claro, temos classes pra outros tamanhos de tela também. No Bootstrap temos essas famílias de classes de grids já prontas:

- **col-xs-** : Extra small. < 768px
- **col-sm-** : Small (tablets). >= 768px
- **col-md-** : Medium (Desktops). >= 992px
- **col-lg-** : Large (Desktops). >= 1200px

Podemos aplicar mais de uma classe ao mesmo tempo no mesmo elemento:

```
<div class="row">
  <div class="col-xs-6 col-sm-4">
    ...
  </div>

  <div class="col-xs-6 col-sm-8">
    ...
  </div>
</div>
```

Nesse exemplo, nosso grid divide no meio (6 pra cada lado) em telas muito pequenas mas depois divide em 4 e 8 pra telas um pouco maiores.

Agora é a melhor hora de aprender algo novo

Se você gosta de estudar essa apostila aberta da Caelum, certamente vai gostar dos novos **cursos online** que lançamos na plataforma **Alura**. Você estuda a qualquer momento com a **qualidade** Caelum.



11.15 – EXERCÍCIOS: GRIDS

1. Nosso design mobile-first funciona muito bem em telas pequenas. Mas conforme vamos aumentando o browser, notamos que tudo fica meio grande. O panel e o form esticam 100%, o que é um exagero em telas maiores.

Vamos usar grids do Bootstrap para transformar nosso design em 2 colunas em telas maiores. Por padrão, o Bootstrap já traz media queries para adaptação em 768px. A ideia é deixar o panel ocupar **4/12** e o form ocupar **8/12**.

São três alterações necessárias:

- Criar um div com classe `row` dentro do container;
- Criar um div com classe `col-sm-4` ao redor do panel;
- Aplicar a classe `col-sm-8` no formulário.

Faça essas alterações e **cuidado** com o resultado final e os milhões de divs misturados. O código deve ficar mais ou menos assim:

```
<div class="container">
  <div class="row">

    <div class="col-sm-4">
      <div class="panel panel-default">
        <!-- ... painel todo aqui ... -->
      </div>
    </div>

    <form class="col-sm-8">
      <!-- ... todos os campos aqui ... -->
    </form>

  </div>
</div>
```

Teste a página e redimensione para um tamanho em torno de 768px pra ver o resultado.

2. Repare que o Bootstrap ajusta várias coisas responsivamente pra gente de maneira automática. Além de aplicar as classes do grid, repare como os tamanhos e fontes aumentam de acordo com a resolução, sem precisarmos fazer nada.

Faça os testes.

3. Quando aumentamos bastante a tela, tudo ainda se ajusta na proporção de 4 pra 8 que definimos. Mas o formulário fica grande demais. Em telas maiores, talvez seja legal deixar o formulário em 2 colunas.

Vamos usar outras classes do grid do Bootstrap que se aplicam em layouts maiores que 992px. Vamos dividir o formulário em 2 partes iguais, ou seja 6/12 (lembre que o grid do Bootstrap tem 12 partes como base). Conseguimos isso tudo usando a classe **col-md-6**.

As mudanças necessárias são:

- Crie um div com classe **row** ao redor dos 2 fieldsets;
- Aplique a classe **col-md-6** em cada um dos fieldsets.

No final, a estrutura deve estar parecida com essa:

```
<form [....]>
  <div class="row">
    <fieldset class="col-md-6">
      ....
    </fieldset>
    <fieldset class="col-md-6">
      ....
    </fieldset>
  </div>
  <button [....]>
</form>
```

Teste a página e redimensione para um tamanho em torno de 992px pra ver o resultado.

4. (opcional) É possível usar mais de uma classe de grid ao mesmo tempo no mesmo elemento. Por exemplo: dividimos a tela em 4/12 para o painel e 8/12 para o formulário. Mas se, em telas maiores, você quiser mudar essa proporção para 3/12 e 9/12, basta adicionar as classes **col-lg-3** e **col-lg-9** em conjunto as que tínhamos antes.

Implemente essa mudança no projeto.

Exemplo:

```
<form class="col-sm-8 col-lg-9">
```

A série **col-lg-** aplica em resoluções acima de 1200px.

Para saber mais sobre os grids do Bootstrap: <http://getbootstrap.com/css/#grid>

5. (opcional) Além de alterar o grid nas diferentes resoluções, o Bootstrap também permite esconder/exibir certos elementos apenas em uma resolução específica.

Por exemplo: imagine que, para otimizar o espaço pequeno no design para smartphone, vamos esconder a imagem do produto. Podemos fazer isso **adicionando** a classe `hidden-xs` na ``. Isso vai esconder o elemento em resoluções menores que 768px.

Para saber mais sobre as classes auxiliares para responsivo do Bootstrap: <http://getbootstrap.com/css/#responsive-utilities>

11.16 – PARA SABER MAIS: COMPONENTES JS DO BOOTSTRAP

Além de componentes CSS puro do Bootstrap como `panel` e `jumbotron`, temos outros componentes mais avançados que envolvem interatividade e JavaScript.

Há muita coisa disponível por padrão no Bootstrap, pelo menos os componentes mais comuns como janela modal, galeria de imagens, dropdowns, menus de navegação e mais.

<http://getbootstrap.com/javascript/>

No exercício, vamos usar o menu superior (`navbar`).

11.17 – EXERCÍCIOS OPCIONAIS: NAVBAR E JAVASCRIPT

1. Um componente muito famoso do Bootstrap é seu menu superior, chamado de **navbar**. O HTML é um pouco mais complexo pois se trata de um menu completo, mas é relativamente fácil.

Implemente um `navbar` em nossa página acima do `jumbotron`, logo no topo da página:

```
<nav class="navbar navbar-default">
  <div class="navbar-header">
    <a class="navbar-brand" href="index.php">Mirror Fashion</a>
  </div>
  <ul class="nav navbar-nav">
    <li><a href="sobre.php">Sobre</a></li>
    <li><a href="#">Ajuda</a></li>
    <li><a href="#">Perguntas frequentes</a></li>
```

```
<li><a href="#">Entre em contato</a></li>
</ul>
</nav>
```

Teste o resultado no navegador.

2. Repare que o menu não gruda no jumbotron por ter uma margem por padrão. Sem problemas, com um CSS bem simples podemos customizar o componente.

Remova a margem da navbar **adicionando** esse CSS:

```
<style>
.navbar {
  margin: 0;
}
</style>
```

Além disso, **adicione** no <nav> a classe navbar-static-top.

Teste novamente.

3. Teste o menu em resoluções menores. Note que o Bootstrap ajusta automaticamente o navbar em telas menores. Por padrão, o comportamento é mudar o menu de horizontal para vertical em mobile.

Veja esse comportamento redimensionando o browser.

4. Uma outra solução para menus em telas pequenas é de juntar as opções em uma espécie de dropdown que só abre quando ativado. Isso é, criar um botão para ativar o menu (geralmente com o famoso ícone do sanduíche).

É bem simples fazer isso com Bootstrap, a funcionalidade está toda pronta.

Para fazer o menu colapsar em telas pequenas, basta **adicionar** 2 classes no : a **collapse** e a **navbar-collapse**.

```
<ul class="nav navbar-nav collapse navbar-collapse">
```

Se você testar agora, vai notar que o menu some nas telas menores. Para exibi-lo, precisamos fazer o próximo passo: criar o ícone que ativa o menu.

Dentro do navbar-header, logo abaixo do <a>, crie um botão de ativação. O botão é apenas o texto "menu" mas possui a classe **navbar-toggle**. Além disso, dois outros parâmetros configuram seu funcionamento com o collapse que usamos antes:


```
<button class="navbar-toggle" type="button"  
  data-target=".navbar-collapse" data-toggle="collapse">  
  menu  
</button>
```

Se testar agora, vai notar que o menu aparece mas não funciona quando clicado. É porque essa funcionalidade no Bootstrap é **implementada com JavaScript**. A boa notícia é que não precisamos escrever uma linha de código JS sequer, mas para tudo funcionar precisamos **adicionar o JavaScript do Bootstrap**.

No fim da página, logo antes de fechar o `</body>`, chame o arquivo do Bootstrap e do jQuery:

```
<script src="js/jquery.js"></script>  
<script src="js/bootstrap.js"></script>
```

Teste novamente e veja o plugin funcionando. Usamos o JavaScript do Bootstrap implicitamente.

Atributos customizados no HTML5

Até a versão 4 do HTML, não havia uma forma padronizada de colocar atributos customizados.

A partir do HTML5, atributos começando com `data-` em qualquer tag são considerados atributos customizados e não quebram a validade do nosso código HTML. Esses atributos são bastante úteis para passar informação para um código Javascript, como fizemos agora, passando informação para o código do Bootstrap.

5. Há muitas opções possíveis para o navbar. Por exemplo, podemos inverter as cores e usar um esquema mais escuro apenas trocando a classe `navbar-default` pela classe `navbar-inverse`.

Para saber mais sobre o navbar: <http://getbootstrap.com/components/#navbar>

6. Por falar em customizações, uma grande vantagem do Bootstrap é seu imenso suporte na comunidade. Isso se traduz em muitas ferramentas e complementos desenvolvidos pra ele, como **novos temas**.

Deixamos no projeto um tema chamado **flatly**, open source. Para usá-lo, basta trocar o `bootstrap.css` pelo arquivo dele no head:

```
<link rel="stylesheet" href="css/bootstrap-flatly.css">
```

7. (opcional) Use ícones do glyphsicons no menu. Basta colocá-los dentro dos itens que quiser. Algumas sugestões:

```
<span class="glyphicon glyphicon-home"></span>
<span class="glyphicon glyphicon-question-sign"></span>
<span class="glyphicon glyphicon-list-alt"></span>
<span class="glyphicon glyphicon-bullhorn"></span>
```

Outra sugestão é trocar a palavra "menu" que usamos no navbar colapsado pelo ícone do sanduíche:

```
<span class="glyphicon glyphicon-align-justify"></span>
```

Se quiser mudar a cor do ícone, basta usar CSS e a propriedade color:

```
.navbar .glyphicon {
  color: yellow;
}
```

Consulte outros na documentação:

<http://getbootstrap.com/components/#glyphicons>

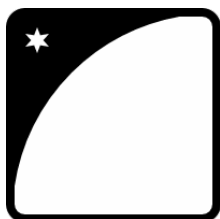
8. (opcional) Troque a classe navbar-static-top pela **navbar-fixed-top**. Repare que o menu fica fixo no topo mesmo com scroll.

Você talvez queira aplicar um padding no body pro conteúdo não subir:

```
body { padding-top: 70px; }
```

9. (opcional) No navbar-header, use um logo da Mirror Fashion em vez de texto.

Você pode também fazer o curso WD-43 dessa apostila na Caelum



Querendo aprender ainda mais sobre HTML, CSS e JavaScript? Esclarecer dúvidas dos exercícios? Ouvir explicações detalhadas com um instrutor?

A Caelum oferece o **curso WD-43** presencial nas cidades de São Paulo, Rio de Janeiro e Brasília, além de turmas

incompany.

[Consulte as vantagens do curso *Desenvolvimento Web com HTML, CSS e JavaScript*.](#)

11.18 – PARA SABER MAIS: OUTROS FRAMEWORKS CSS

O Bootstrap não é o único framework CSS do mercado. É talvez o mais famoso e com mais usuários, mas há muitas outras opções que às vezes podem ser até melhores para seu caso.

Três opções famosas:

- **Foundation:** Da Zurb, fortemente baseado em mobile e responsivo. <http://foundation.zurb.com/>
- **Semantic UI:** tem nomes de classes mais simples e semânticos que os outros. <http://semantic-ui.com/>
- **Pure:** Do Yahoo, outra alternativa, mais recente. <http://purecss.io/>

De maneira geral, esses frameworks permitem fazer as mesmas coisas, mas cada um com seu estilo. Um botão principal por exemplo:

```
<!-- Bootstrap -->
<button class="btn btn-primary btn-lg">Clique aqui</button>

<!-- Foundation -->
<button class="large button">Clique aqui</button>

<!-- Semantic UI -->
<button class="large ui button">Clique aqui</button>

<!-- Pure -->
<button class="pure-button pure-button-primary pure-button-large">
  Clique aqui
</button>
```

11.19 – DISCUSSÃO EM AULA: OS PROBLEMAS DO BOOTSTRAP E QUANDO NÃO USÁ-LO

CAPÍTULO ANTERIOR:

[PHP: parâmetros e bancos de dados](#)

PRÓXIMO CAPÍTULO:

[jQuery](#)

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter