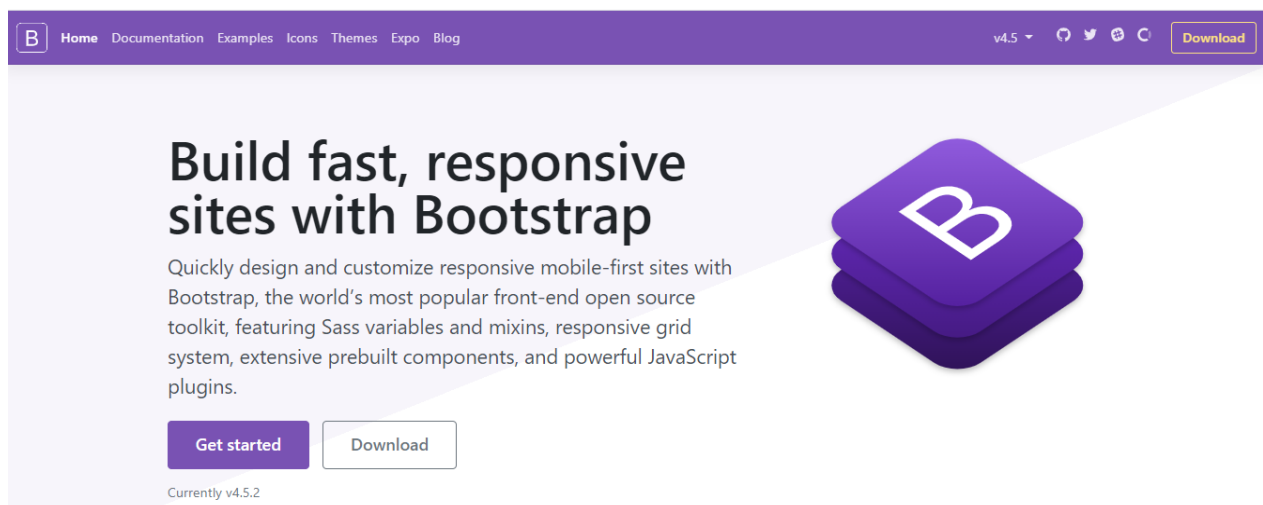


Bootstrap

O que é, pra que serve. como baixar?

Uma biblioteca feita inicialmente pelo twitter e depois incorporada por outras empresas, um projeto livre para download e utilização em qualquer site, incluindo site comercial. Basicamente é a biblioteca mais popular frondEnd hoje em dia. É uma biblioteca que permite, facilita criar varias coisa do dia-a-dia, como deixar um site responsivel, deixa pre pronto para deixar o site ser criado automaticamente responsivel. Como dividir conteúdos na página, criar menus dropdown, ou seja, infinitos componentes que pode ser feito download da internet e incorporar dentro do bootstrap. Foi criado para funcionar sem a necessidade do javascript para algumas funcionalidades, pelo menos as principais. Vamos conhecer as principais funcionalidades do bootstrap versão 4.5.2 a mais atual no momento, que permite fazer a separação de conteúdo o que chamamos de grid.

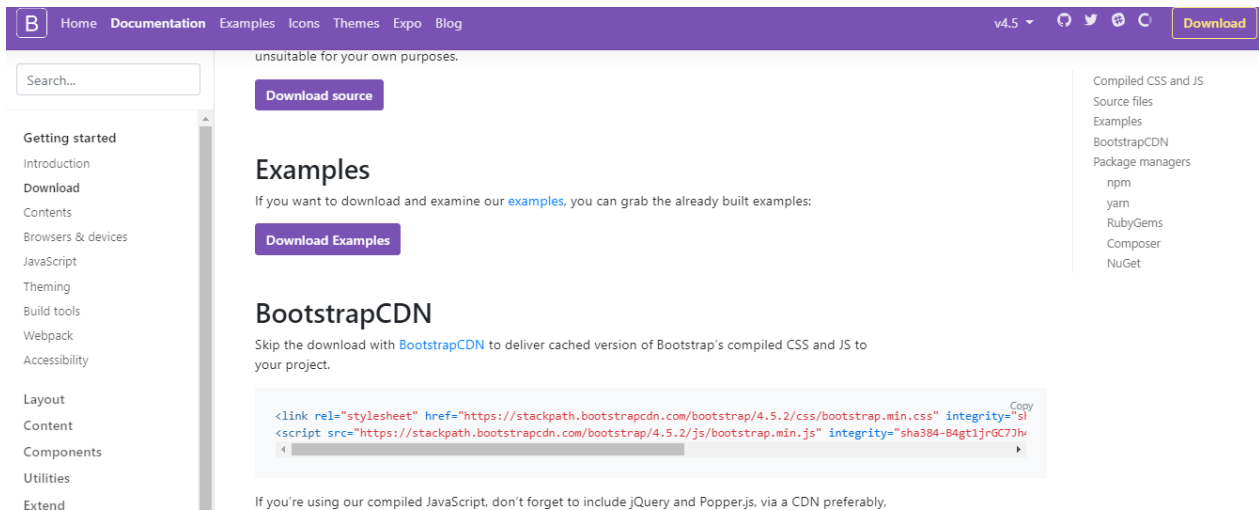
Primeiro vamos acessar o site **getbootstrap.com** ou digitar no google bootstrap download e com acesso ao site clicar em download para baixar um arquivo zip.



Se clicar no botão vai dar acesso a outra pasta. Existem duas formas de adicionar o bootstrap no seu projeto:

Primeiro baixando os arquivos do bootstrap, que faz o download de um arquivo zip que contém os arquivos.

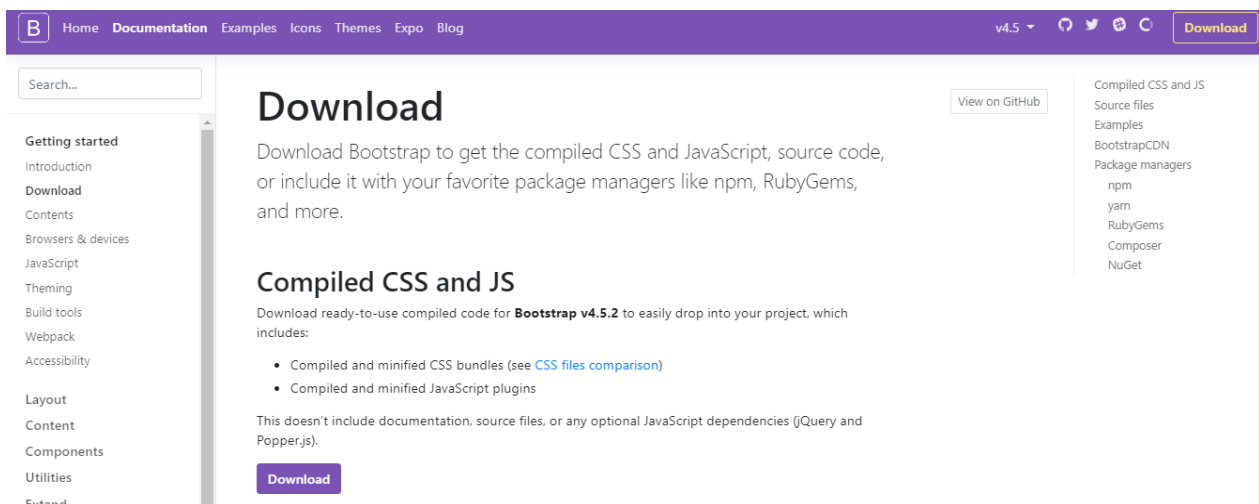
A segundo forma é o que chamamos de CDN. Que são links públicos, ou seja, os links que aparecem conforme mostra a imagem a seguir:



O acesso a esses links públicos se dá através do acesso do botão download na parte inferior mostrado na tela de download inicial. Esses links pode ser copiado e colado no seu projeto que automaticamente o bootstrap vai estar funcionando. Para teste e estudo pode ser feito assim, sem precisar baixar bibliotecas, porém ficamos dependentes desse site estar no ar. Ou seja, se criamos todo site utilizando CDN se eventualmente esse link seja trocado ou o site sai do ar, automaticamente o seu site também sairá do ar.

Então o recomendado é baixar toda a biblioteca para usar no projeto sem ficar dependente do CDN.

Vamos agora fazer o download do bootstrap:



Clicando no botão de download baixa um arquivo zipado com duas pastas, uma do CSS e outra do JS (javascript).

Sobre essas duas pastas vamos ver primeiro os arquivos da pasta CSS. Não vamos precisar de todos esse arquivos que ela contém. Na verdade vamos precisar de apenas um desses arquivos.

Nome	Tamanho	Tipo
bootstrap-grid.css	44 KB	CSS
bootstrap-grid.min.css	34 KB	CSS
bootstrap-reboot.css	5 KB	CSS
bootstrap-reboot.min.css	4 KB	CSS
bootstrap.css	178 KB	CSS
bootstrap.min.css	145 KB	CSS
bootstrap-grid.css.map	96 KB	Documento
bootstrap-grid.min.css.map	76 KB	Documento
bootstrap-reboot.css.map	58 KB	Documento
bootstrap-reboot.min.css.map	26 KB	Documento
bootstrap.css.map	412 KB	Documento
bootstrap.min.css.map	552 KB	Documento

Na pasta temos 3 versões do bootstrap, a versão `bootstrap.css` que inclui tudo o completo assim dizendo, podem observar que o tamanho dela é maior que dos outros arquivos na pasta.

Tem a versão `bootstrap.reboot.css` que é basicamente um conjunto de regras de css que padroniza a página em todos os navegadores. Isso quer dizer se usar esse no arquivo vai ficar padrão em todos os navegadores como, Google Chrome, Firefox, Edge e etc. Esse css tipo reseta, por isso o nome reboot. Dentro do css inteiro, ou seja, o arquivo *bootstrap.css* já vem ele, a não ser que queira separado.

O próximo é o `bootstrap.grid.css`, basicamente é a funcionalidade que vai ser mais usada em projetos. É a biblioteca responsável pela divisão de conteúdos na página, ou seja, a organização de menus, botões e documentos em uma página. Então essa é praticamente a única funcionalidade do grid. Se quer usa somente esse arquivo.

Agora se quer pegar todas as funcionalidades de uma vez usa-se o `bootstrap.css`.

Então tem o css completa ou seja o css legível e tem a versão mini ficada que é o arquivo *bootstrap.min.css* que basicamente é a mesma coisa do `bootstrap.css` só com o css todo comprimido, sem espaço, todos os códigos juntos, de forma que podem ver na imagem acima o arquivo fica com tamanho menor. Assim vale para os outros arquivos como podem ver na imagem acima.

Então, sempre que for utilizar usar a versão min, que vai deixar o projeto mais leve.

Vamos usar a versão min do css completa, ou seja, *bootstrap.min.css*, com isso vamos estar utilizando todas as funcionalidade do css.

Pegamos esse arquivo e colocamos onde está o projeto em que vamos utiliza-lo.

Agora vamos para a pasta JS, onde está o javascript:

Nome	Tamanho	Tipo
bootstrap.bundle.js.map	327 KB	Documento
bootstrap.bundle.min.js.map	274 KB	Documento
bootstrap.js.map	195 KB	Documento
bootstrap.min.js.map	162 KB	Documento
bootstrap.bundle.js	196 KB	JavaScript
bootstrap.bundle.min.js	68 KB	JavaScript
bootstrap.js	115 KB	JavaScript
bootstrap.min.js	49 KB	JavaScript

Percebam que é o mesmo esquema da pasta do css.

O bootstrap para rodar todas as funcionalidade, componentes precisa de 3 coisas em relação a javascript:

Primeiro o jQuery, tem que ter obrigatoriamente o jquery. Depois vai precisar o pop, que é outra biblioteca complementar que alguma funcionalidade específica do bootstrap precisa dessa biblioteca. Por exemplo, o dropdown (clica e aparece outros menus...).

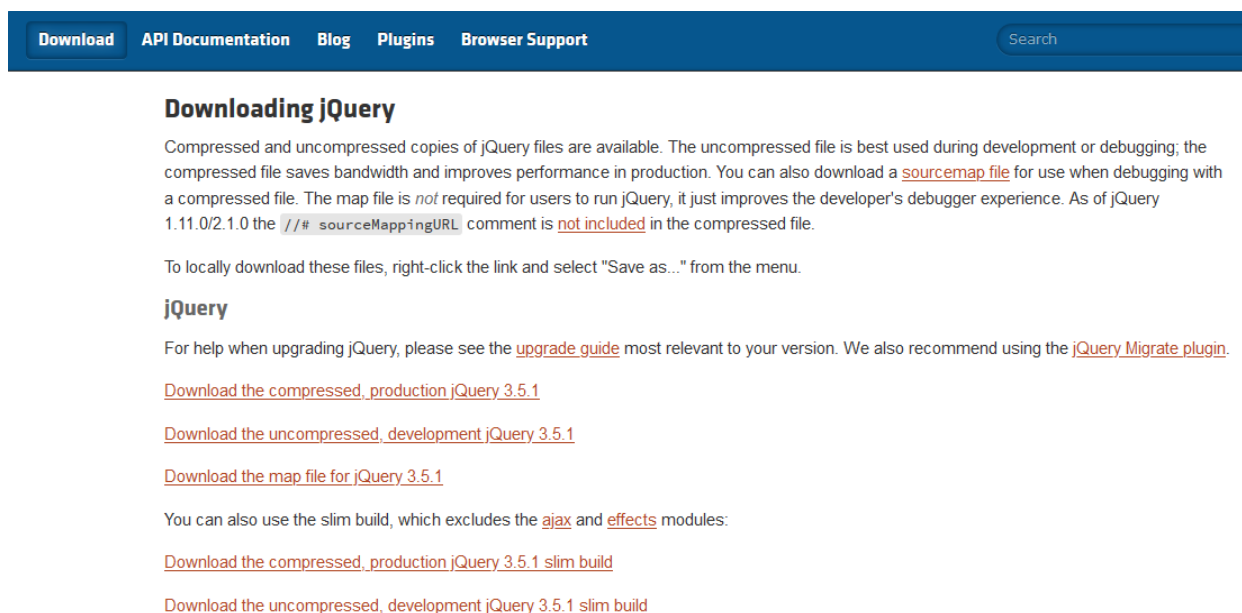
Se utilizarmos o bootstrap.js vai estar só o bootstrap, então teremos que baixar o jQuery e o pop. Agora, se pegarmos o arquivo *bootstrap.bundle.js* já vem o bootstrap e o pop juntos. Só que lembrando a versão min é mais leve, por isso utilizamos o arquivo *bootstrap.bundle.min.js*.

Percebam a diferença no tamanho do arquivo completo *bootstrap.bundle.js* tem 196 KB e o minificado *bootstrap.bundle.min.js* tem 68 KB.

Usando esse arquivo *bootstrap.bundle.min.js* ele não vem com o jQuery, portanto tem que usar o jQuery separadamente.

Como fizemos com o arquivo css, vamos pegar o arquivo *bootstrap.bundle.min.js* e colocamos onde está o projeto em que vamos utiliza-lo.

Agora vamos baixar o jQuery, para isso acessamos o site **jquery.com** e clicar em download. E ai vamos baixar a versão comprimida de produção, ou seja no link Download the compressed, production jQuery 3.5.1 conforme mostra a imagem abaixo:



The screenshot shows the jQuery website's download page. At the top, there is a navigation bar with links: Download, API Documentation, Blog, Plugins, and Browser Support. A search bar is also present. The main heading is "Downloading jQuery". Below it, a paragraph explains that compressed and uncompressed copies are available, with the compressed version being better for production. It mentions a "sourcemap file" for debugging. A note states that the "sourceMappingURL" comment is not included in the compressed file. Instructions are given to right-click the link and select "Save as...". The "jQuery" section follows, with a link to the "upgrade guide" and a recommendation to use the "jQuery Migrate plugin". Three download links are provided: "Download the compressed, production jQuery 3.5.1", "Download the uncompressed, development jQuery 3.5.1", and "Download the map file for jQuery 3.5.1". A section for the "slim build" is also shown, with links for "Download the compressed, production jQuery 3.5.1 slim build" and "Download the uncompressed, development jQuery 3.5.1 slim build".

Essa versão é comprimida do jQuery. Feito o download colocamos esse arquivo no projeto em que vamos utiliza-lo.

Se quiser nessa pasta que está todos esses arquivos podemos criar uma outra pasta chamada **assets** e dentro dessa pasta vamos ter o css, javascript, as imagens que vamos usar no nosso projeto. Isso só por questão de organização.

Então dentro dessa pasta chamada assets, vamos criar outra pasta que vamos chamar de css e outra pasta que vamos chamar de js e uma terceira que vamos chamar de images.

Agora vamos colocar os 3 arquivos feito download antes dentro dessas pastas que acabamos de criar.

Primeiro pegamos o arquivo *bootstrap.min.css* e colocamos dentro da pasta *css*;

Segundo passo, vamos colocar os dois arquivos *bootstrap.bundle.min.js* e *jquery-3.5.1.min.js* na pasta *js*.

Estruturando o projeto

A partir de agora vamos inserir o bootstrap no nosso projeto e começar a utilizá-lo.

Primeiro o bootstrap usa o HTML 5 e por isso devemos começar nosso projeto com a tag doctype.

```
<!doctype html>
```

Feito isso colocamos a famosa tag *html*

```
<html> </html>
```

E dentro dela colocamos a tag *<head>* que é o cabeçalho como já sabem disso.

Depois dentro dela colocamos a tag meta *<meta charset="UTF-8"/>*

E após essa tag devemos usar obrigatoriamente um viewport ou seja

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

O que é o viewport? Significa dizer para o navegador a forma de como renderizar o conteúdo na tela. Para isso vamos especificar algumas propriedades no atributo *content*.

A primeira propriedade que vamos especificar é a largura do site ou seja *width* vai ser igual a largura do dispositivo (computador, celular, TV, etc) que estiver acessando o site (*device-width*). Após isso colocamos vírgula e dizemos se pode ser feito zoom no site. No caso para site responsivo não se dá zoom, então colocamos *initial-scale=1* que significa o zoom inicial é 1 e em 1 que vai ficar funcionando. Depois colocamos mais uma vírgula para a última propriedade (*shrink-to-fit=no*) que diz pro computador que ele não tem que comprimir as coisas, os elementos para que ele fique na tela. Significa deixar para o bootstrap essa tarefa.

Depois o próximo passo ainda dentro da tag *head* vamos colocar o *css* do bootstrap que está dentro da pasta *assets - css - bootstrap.min.css* para isso colocamos:

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

O bootstrap leva *css* e também *javascript*. Para isso adicionamos o *javascript* agora, mas observem ele não vai dentro da tag *head*, por quê? Se colocarmos dentro do *head* atrasa o carregamento da página, como queremos que a página carregue mais rápido colocamos fora do *head*. Pois tudo que está dentro do *head* vai ser carregado antes da página aparecer na tela. Colocamos o *css* dentro do *head* para evitar a página aparecer toda quebrada, como se vê algumas vezes quando acessamos certos sites. Estando dentro do *head* o *css* já vai fazer aparecer todo arrumado o site. Então após o *css* aparecer que vai ser carregado o *javascript*. Para isso colocamos o *javascript* no final da página, ou seja, antes do final da tag *body*.

Qual a ordem de inserir as coisas, ou seja os *javascripts*?

Sempre o *jQuery* vai ser inserido por primeiro:

```
<script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>
```

Logo depois adicionamos o *bootstrap.bundle*:

```
<script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>
```

Com isso temos tudo para rodar o jQuery perfeitamente no nosso sistema.

A seguir o projeto que foi explicado até aqui como podem ver.

Arquivo index.php

```
<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>

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

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


    <title> Projeto Bootstrap 4 </title>

    </head>

    <body>

        <h1> Hello World </h1>

        <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

        <script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

    </body>

</html>
```

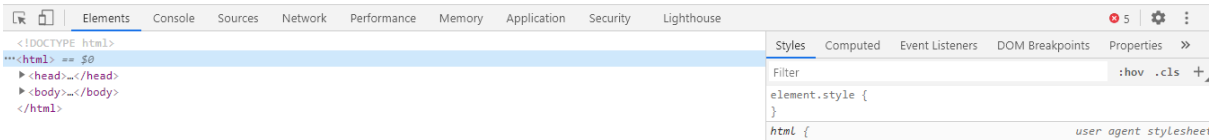
Usando o Console do Navegador

Durante todo o desenvolvimento do projeto vamos estar lidando com responsabilidade, como os componentes se entregam, como funcionam em grupo e individualmente. Pra isso, principalmente no Google Chrome e também no Firefox tem essa funcionalidade também de simular as dimensões um determinado dispositivo, como um celular, um tablet, enfim de varias coisas diferentes ou então até de nossa tela normal. E fizemos isso através de uma ferramenta que tem nos navegadores com poucas diferenças entre eles, mas o objetivo e a aparências são muitas semelhantes.

No chrome para ter acesso a essa ferramenta vamos com o botão direito e no botão inspecionar.

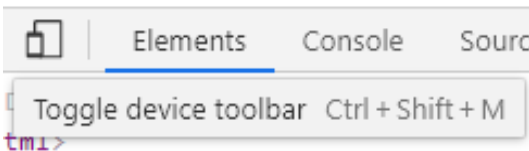
Vai aparecer algo parecido com isso:

Hello World

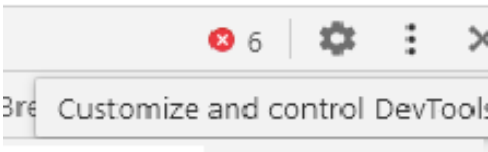


O que chamamos de console. Nesse console tem algumas tabs como a aba elements. Ela mostra o html que esta aparecendo na tela nesse momento. Não necessariamente o html do código fonte do seu site.

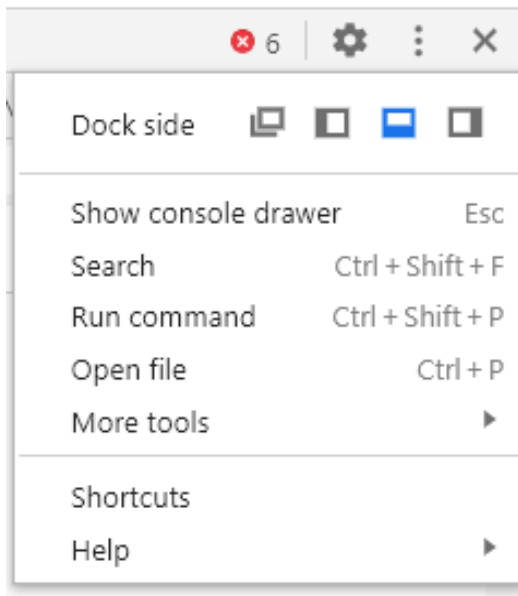
Existe esse botão o Toggle device toolbar nessa área do console:



Ele vai tornar a área de exibição personalizável, antes de fazermos isso observem a direita onde tem os 3 pontinhos:

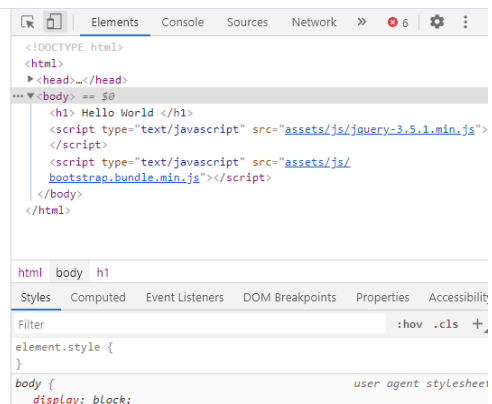


Clicando neles observem que aparece um menu igual a esse na imagem a seguir:



E observem na linha onde tem Dock side, cada imagens dos retângulos ao lado representa como você vai posicionar o console na tela. Normalmente a maioria dos programadores preferem o Dock to right, o último a esquerda.

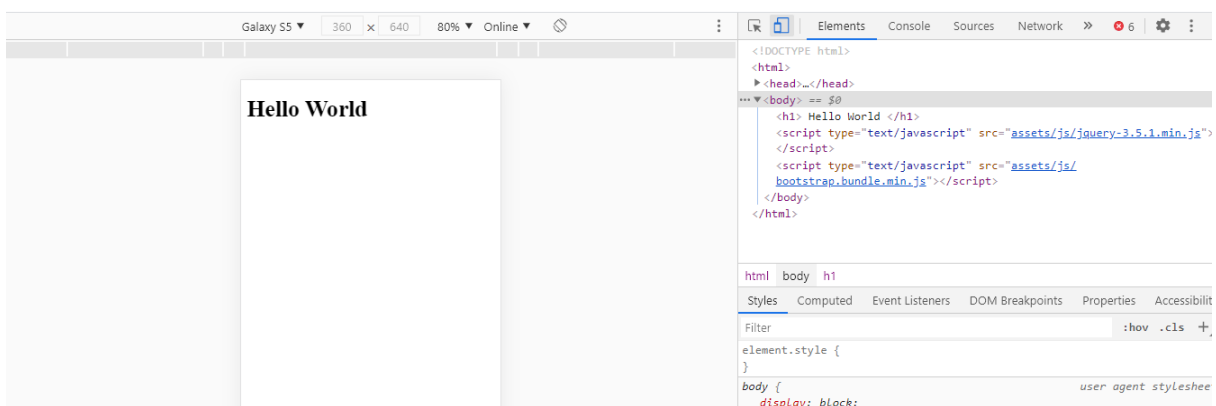
Hello World



Quando clicarmos no botão Toggle device toolbar



vai simular um celular como podem ver na imagem a seguir:



Assim podemos ver tanto a tela do dispositivo celular e os dados ao lado, por isso é mais usado esse formato do console no formato Dock to right. Favorecendo uma melhor visualização do dispositivo e dos dados.

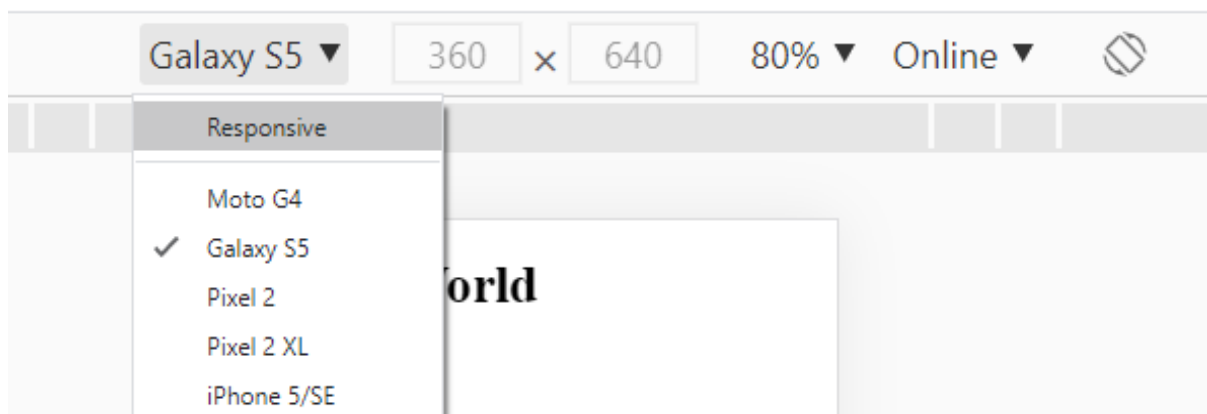
Observem que para aparecer na tela, toda a tela do celular geralmente o zoom fica por padrão no valor 80% ou 83%.



Se alterarmos, por exemplo, para 100% esse zoom percebam que altera a visualização do celular, mas altera somente o zoom, não o tamanho da tela do celular, ocasionando o surgimento de uma tela de rolagem. Então basta clicarmos em Fit to window que vai fazer com que seja visualizado por inteiro na tela o dispositivo.

Pode ser feita uma rotação do dispositivo e para isso basta clicar no botão Rotate a direita do botão do zoom. Mas normalmente se usa no formato do celular em pé, como aparece por padrão no momento de abrir o console.

Observem que tem varias opções de dispositivos já pré-definidos. Dependendo do tamanho ele altera o zoom para caber na tela o dispositivo. Se quiser deixar livre só colocar responsivel, que possibilita redirecionar. Mas normalmente escolhe um modelo de dispositivo.



Grid - Introdução

A partir de agora vamos colocar a mão na massa e aprender usar o bootstrap 4. Para inserir conteúdos dentro de uma estrutura feita com bootstrap podemos utilizar e é muito recomendável por dentro de container. Que é basicamente uma div que determina até onde seu site vai, por exemplo, vamos colocar o Hello World que temos no nosso projeto dentro de uma div.

Para isso criamos a div:

```
<div class="container">

    <h1>Hello World </h1>

</div>
```

Agora após salvar essa alteração e executar o arquivo na página, percebam Hello World teve um recuo para a direita.

Hello World

Vamos colocar uma cor de fundo no container só para poderem visualizar melhor essa div.

```
<div class="container" style="background-color:#FF0000">
```

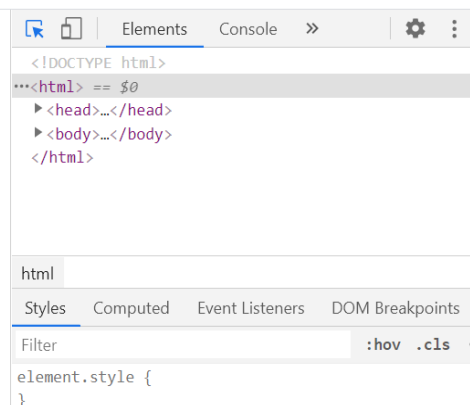
Atualizando a página percebam como ficou:

Hello World

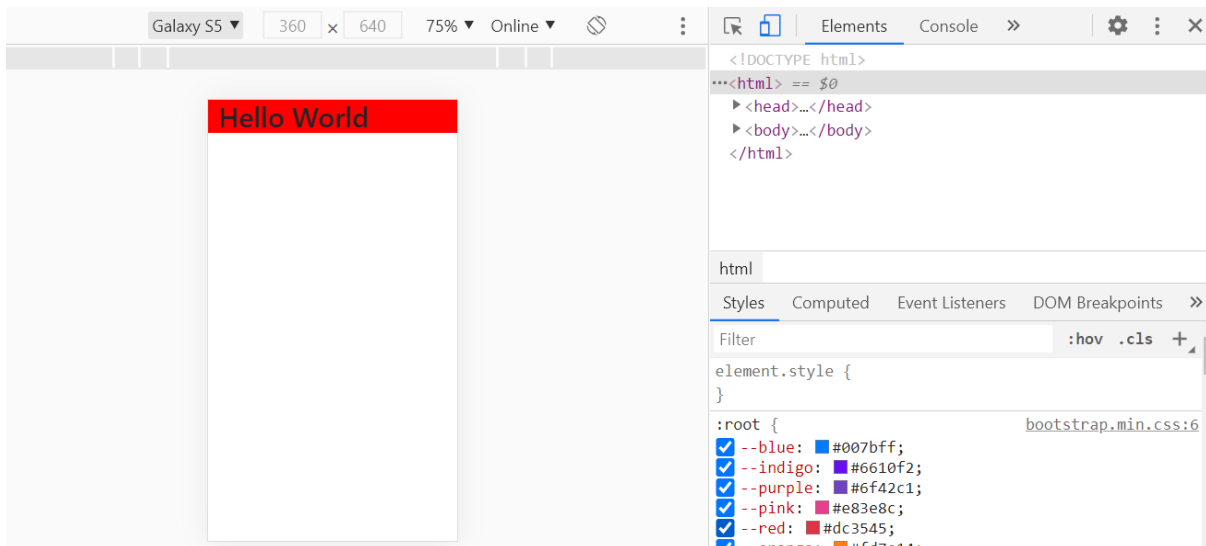
Basicamente o container estabeleceu um limite padrão para o site. por que essa limitação? Por causa de monitores menores. Essa limitação não é fixa, ou seja, se estiver num tablet, ou celular ela vai ser menor. Então vai existir momentos que não vamos ver esse limite em branco nos lados do container.

Por exemplo, com botão direito na tela clicamos em inspecionar e percebam que já diminui a área pois já está entendendo que pode ser a largura de um tablet:

Hello World



Agora vamos alterar para mobile, um Galaxy S5, percebam que a div container ficou de ponta a ponta, sem as laterais em branco. Por quê? Por que você já está num celular, tem a tela pequena, então já se adaptou a tela.



Agora que entenderam sobre container, precisam saber que existem 2 tipos de container no bootstrap.

O container normal que é esse que acabaram de ver, e tem também o container fluido. Para utiliza-lo vamos na div do container e alteramos o valor da classe para container-fluid, como podem ver a seguir:

```
<div class="container-fluid" style="background-color:#FF0000">
```

Após atualizamos a página e observem como fica:



O container fluid, a pesar dele funcionar como o container, ter todas as ajudas que um container tem, que vamos ver mais logo mais, ele tem a abrangência de ponta a ponta no site. Tem sites que usa esse tipo de estruturação e outros a exemplo do facebook não pega de ponta a ponta no site, pode até der um chat ao lado mas não pega de ponta a ponta, o site não se expande na tela, não usam o container-fluid por exemplo. O que não ocorre por exemplo em site de lojas virtuais, que quanto mais espaço tiver, são mais produtos para aparecer para o cliente. Então tem site que usa de uma forma e outros preferem da outra forma. Não existe uma certo e outro errado. Normalmente vamos usando o container normal, sem o container-fluid. Em alguns casos específicos, como um sistema, por exemplo, você não está fazendo um site, mas um sistema o que engloba toda a área da tela, é mais espaço para mostrar informações, e então usamos o container-fluid.

Veja como está o código do projeto até aqui:

Arquivo index.php

```
<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>
```

```

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

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


<title> Projeto Bootstrap 4 </title>

</head>

<body>

    <div class="container-fluid" style="background-color:#FF0000">

        <h1> Hello World </h1>

    </div>

    <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

    <script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>

```

Agora que aprenderam esse conceito de container, vamos começar com outro conceito que se chama grid que é um dos recursos principais do bootstrap.

O que são grids? Para entenderem melhor, façam uma associação visual com o Excel por exmplo, que são varias colunas e tudo mais, isso é basicamente um grid. Ou seja, são varias divisões de conteúdos. Por exemplo, no projeto que fizemos até aqui, podemos fazer um grid com 3 divisões, onde tenha conteúdo numa coluna a direita, centro e esquerda. Grid é basicamente isso, uma forma de dividir proporcionalmente todo o conteúdo do site ou sistema.

Vamos aprender fazer um grid com o bootstrap, mas mais infinitas possibilidades que tem.

Vamos iniciar com o código do projeto assim, tiramos a cor de fundo e deixamos só o container mesmo, mas podemos ter vários container no site sem problemas:

Arquivo index.php

```

<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>

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

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


    <title> Projeto Bootstrap 4 </title>

    </head>

    <body>

        <div class="container">


        </div>

        <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

```

```

<script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>

```

Vamos começar fazendo um grid básico. Para fazer um grid básico temos que definir 2 coisas. Primeiro definir a linha, definindo a linha depois vamos dizer quantas colunas vai ter dentro dessa linha. No bootstrap 4 utiliza a tecnologia flex, com isso veio a capacidade das divs se adaptar, de interpretar na quantidade de divs do conjunto. Vamos pra prática agora, dentro do container vamos colocar uma linha:

```

<div class="container">

    <div class="row">

    </div>

</div>

```

Row é linha, depois disso vamos colocar 3 colunas dentro dessa linha:

```

<div class="container">

    <div class="row">

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col">...</div>

    </div>

</div>

```

Percebam quando atualizar a página index.php que fica assim conforme imagem a seguir:

...

Parece estranho as colunas mas percebam que o conteúdo delas que no caso é os 3 pontinhos estão alinhados a esquerda de cada coluna.

Para terem uma visualização do espaço das colunas vamos colocar uma cor de fundo em uma das linhas:

```

<div class="container">

    <div class="row">

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col" style="background-color:#FF0000">...</div>

    </div>

</div>

```

Atualizando a página fica assim:



Vamos colocar a mesma configuração de cor também na primeira coluna:

```
<div class="container">
  <div class="row">
    <div class="col" style="background-color:#FF0000">...</div>
    <div class="col">...</div>
    <div class="col" style="background-color:#FF0000">...</div>
  </div>
</div>
```

Atualizando a página fica assim:



Vamos colocar uma cor diferente na coluna do meio:

```
<div class="container">
  <div class="row">
    <div class="col" style="background-color:#FF0000">...</div>
    <div class="col" style="background-color:#FFEE00">...</div>
    <div class="col" style="background-color:#FF0000">...</div>
  </div>
</div>
```

Atualizando a página fica assim:



Percebam que as 3 colunas ocuparam todo espaço disponível dentro do container e distribuíram de forma que todas tem o mesmo tamanho.

Se excluir uma das colunas, elas automaticamente iram ocupar espaço de forma que sejam do mesmo tamanho.

Atualizando a página fica assim:



Ou seja, foi excluído no exemplo a última coluna com cor de fundo em vermelho.

Vamos deixar as 3 colunas como antes no nosso código por enquanto:

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

```

<div class="row">

  <div class="col" style="background-color:#FF0000">...</div>

  <div class="col" style="background-color:#FFEE00">...</div>

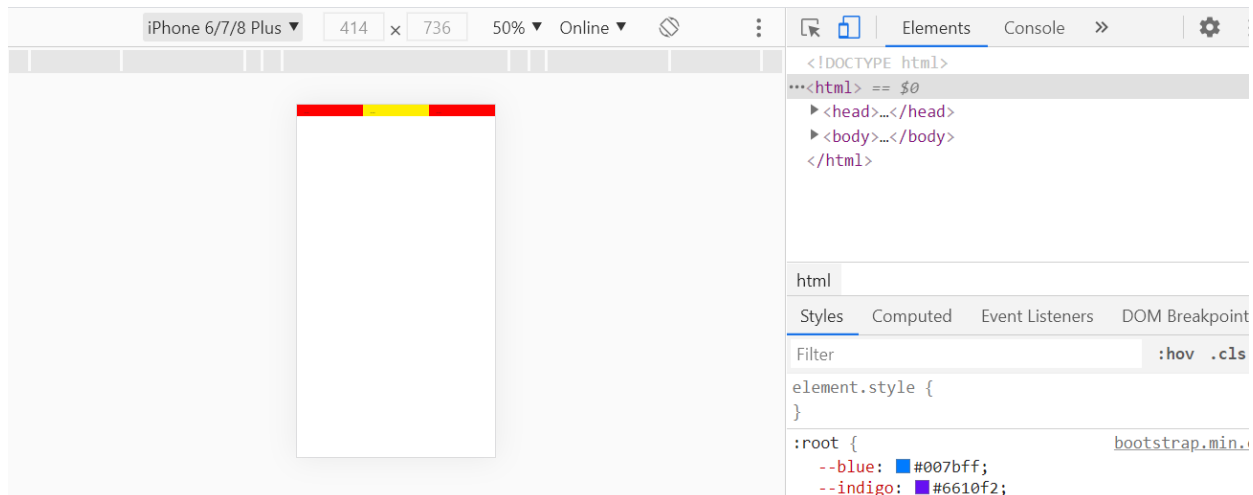
  <div class="col" style="background-color:#FF0000">...</div>

</div>

</div>

```

Agora vamos transformar ou seja, visualizar um celular. Botão direito, inspecionar, botão Togge Device toolbar (ctrl + shift + M).



Percebam que o conteúdo das 3 colunas diminuíram, mas para um celular fica estranho ter colunas desse tamanho. O normal em um celular é que as colunas fiquem uma abaixo da outra, é o natural que ocorre em celular. Mas então por que não esta acontecendo? Por que não especificamos um limite mínimo para essas colunas, e temos algumas formas de especificar. Vamos alterar na linha das colunas colocando **col-sm**, sm informa que é o limite de um celular.

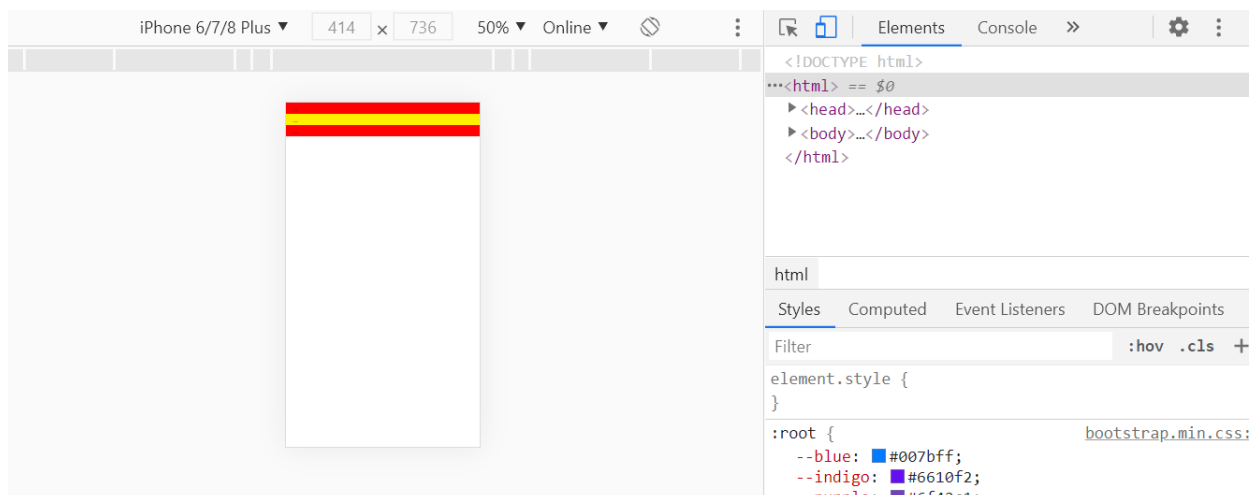
```

<div class="col-sm" style="background-color:#FF0000">...</div>

```

Isso colocamos nas 3 linhas de código das colunas.

Observem o momento que atualizar a página uma ficou abaixo da outra:



Agora se mudarmos para um iPad por exemplo, elas iram ficar lado a lado, por quê? Por que o tamanho dela suporta.

Existe o md e lg. O media device, para dispositivos um pouco maiores mas serve para celular e lg para monitores maiores. É bom vocês aproveitar e já testarem com esse código e redimensionar a tela estando no modo responsivo abre nas laterais para arrastar a tela.

Grid - Coluna com largura e nova linha

Vamos ver como colocar colunas específicas com tamanhos específicos e também fazer com que as colunas pulem de linha sem criar uma nova linha.

Primeiro, verifique o código do projeto que foi colocado um pequeno css para ficar melhor de visualizar com uma borda e cor uniforme.

Arquivo index.php

```
<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>

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

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

    <title> Projeto Bootstrap 4 </title>

    <style type="text/css">

        .row [class^=col-], .row .col{

            background-color:#DDD;

            border:1px solid #ccc;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="row">

            <div class="col">...</div>

            <div class="col">...</div>

            <div class="col">...</div>

        </div>

    </div>

</div>
```



```

<script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

<script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>

```

Ficando assim inicialmente a tela:



Assim como o css separado, não vai ser preciso mudar a cor das colunas na linha do código da coluna dentro do corpo da página.

Agora vamos colocar no código do arquivo mais 3 colunas:

```

<div class="container">

    <div class="row">

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col">...</div>

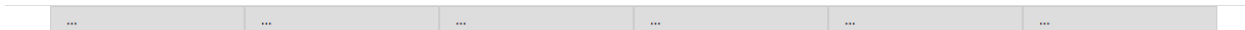
        <div class="col">...</div>

    </div>

</div>

```

Atualizando a página fica assim:



Só que agora não queremos mais que fique 6 colunas e sim 3 colunas, pule a linha e continue as outras 3 colunas. Para isso, após as 3 primeiras colunas colocamos outra div como `<div class="w-100">` :

Deixamos assim o código:

```

<div class="container">

    <div class="row">

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="col">...</div>

        <div class="w-100"></div>

        <div class="col">...</div>

        <div class="col">...</div>

    </div>

</div>

```

```
<div class="col">...</div>

</div>

</div>
```

Não colocamos conteúdo nessa div para quebra de linha. Essa é uma das formas de quebrar a linha.

Atualizando a página fica assim:

...
...

Uma outra forma de quebrar a linha é tirando essa div de quebra e colocando outra `<div class="row">` por exemplo:

```
<div class="container">

  <div class="row">

    <div class="col">...</div>

    <div class="col">...</div>

    <div class="col">...</div>

  </div>

  <div class="row">

    <div class="col">...</div>

    <div class="col">...</div>

    <div class="col">...</div>

  </div>

</div>
```

Atualizando a página fica assim:

...
...

O resultado será o mesmo, só que foi preciso criar mais uma row com 3 colunas. Existe situações que iremos dependendo criar de uma forma ou de outra dependendo do seu projeto.

Como definimos tamanhos específicos para cada coluna? Por exemplo, queremos que a coluna do meio tenha tamanho igual a 2, só que o bootstrap calcula as divisões das colunas por 12, se estamos dizendo que uma coluna tem tamanho 2 e relativo a 12 colunas. Vamos colocar `col-2` para tamanho equivalente a 2 contando que teria um espaço de 12 colunas. Veja aplicação no código:

```
<div class="row">

  <div class="col">...</div>

  <div class="col-2">...</div>
```

```

        <div class="col">...</div>
    </div>
    <div class="row">
        <div class="col">...</div>
        <div class="col">...</div>
        <div class="col">...</div>
    </div>

```

Atualizando a página fica assim:

...
...

Quando colocamos um número, ele automaticamente começa a fazer a matemática dividindo por 12.

Observem que ficou com espaço menor essa coluna como se coubessem 12 colunas nesse espaço.

Agora vamos colocar mais uma de tamanho 2 ou seja coluna também com tamanho 2:

Atualizando a página fica assim:

...
...

Agora vamos definir também tamanho 2 para terceira coluna.

Atualizando a página fica assim:

...
...

Percebam que não ocuparam todo espaço, por que agora elas tem tamanho pré-definido.

Agora se colocarmos tamanho 4 para a primeira coluna:

```

<div class="row">
    <div class="col-4">...</div>
    <div class="col-2">...</div>
    <div class="col-2">...</div>
</div>

<div class="row">
    <div class="col">...</div>
    <div class="col">...</div>
    <div class="col">...</div>
</div>

```

Atualizando a página fica assim:

...	
...

Ficou o equivalente ao preenchimento de 8 colunas e ficou faltando o preenchimento de algumas, no caso de 4 para preencher as 12 colunas. Temos no exemplo uma que equivale a 4, outra que equivale a 2 e mais uma que equivale a 2 e uma outra que fica faltando que equivale a 4.

Se colocarmos mais uma coluna nessa div row que equivale a 4, como mostra o código a seguir onde está em vermelho:

```
<div class="row">

    <div class="col-4">...</div>

    <div class="col-2">...</div>

    <div class="col-2">...</div>

    <div class="col-4">...</div>

</div>

<div class="row">

    <div class="col">...</div>

    <div class="col">...</div>

    <div class="col">...</div>

</div>
```

Atualizando a página fica assim:

...
...

Observem que preencheu agora os 12 espaços imaginários.

Então quando colocamos um número na coluna, automaticamente estamos fazendo com que ela fique com tamanho "fixo" isso digo entre aspas, por que é relativo ao espaço que tem no momento. Se o espaço for menor, se o dispositivo for um celular vai ficar tamanho 12 relativos ao tamanho do celular. Pega o tamanho do espaço disponível e divide por 12 e ai col-1 por exemplo vai ser 1 espaço imaginário de 12.

Se utiliza muito esse tipo de coisa, por exemplo uma coluna tem que ser um tamanho específico, fixa e outra depende do tamanho da tela do dispositivo do usuário, ou seja, utiliza muito esse tipo de formatação.

Podemos fazer da forma que quiser, por exemplo:

```
<div class="row">

    <div class="col">...</div>
```

```

        <div class="col-2">...</div>

        <div class="col-2">...</div>

</div>

<div class="row">

    <div class="col-6">...</div>

    <div class="col">...</div>

    <div class="col">...</div>

</div>

```

Atualizando a página fica assim:

...
...

Podemos fazer dessa forma e também podemos definir o limite mínimo para celular como sm:

```

<div class="container">

    <div class="row">

        <div class="col-sm">...</div>

        <div class="col-sm-2">...</div>

        <div class="col-sm-2">...</div>

    </div>

<div class="row">

    <div class="col-6">...</div>

    <div class="col">...</div>

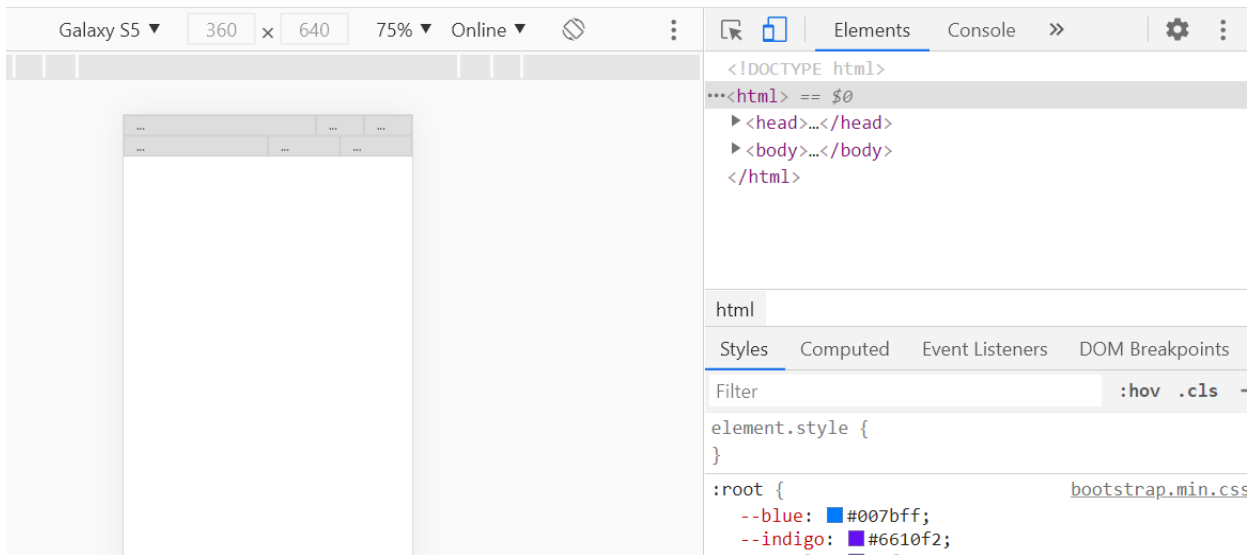
    <div class="col">...</div>

</div>

</div>

```

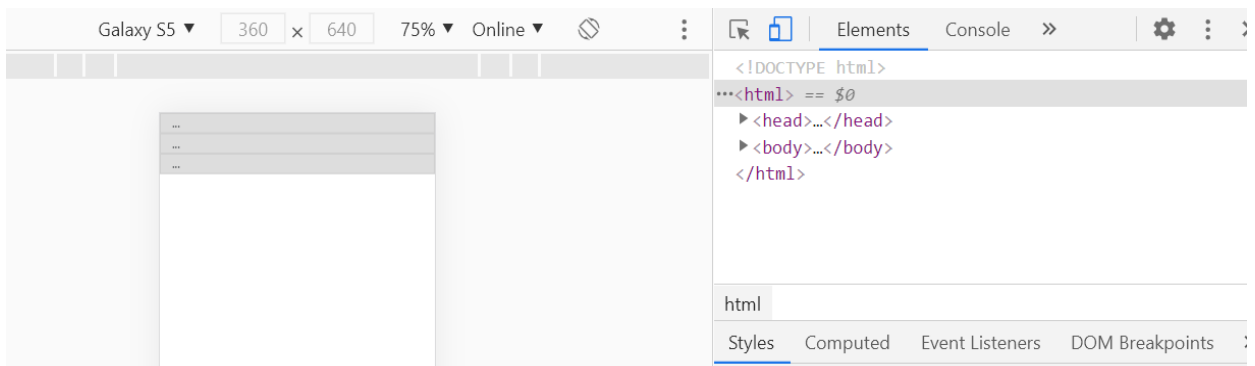
Atualizando a página fica a mesma coisa, mas para vermos como fica em um Celular, temos que ir em ***inspecionar*** e percebem que se ajusta perfeitamente também para celular:



Agora tiramos a segunda linha no código:

```
<div class="container">
  <div class="row">
    <div class="col-sm">...</div>
    <div class="col-sm-2">...</div>
    <div class="col-sm-2">...</div>
  </div>
</div>
```

Percebam como fica para celular o código acima:



Observem com limite mínimo elas quebram no celular.

Mudando novamente o código:

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

```

<div class="col-sm">...</div>

<div class="col-2">...</div>

<div class="col-2">...</div>

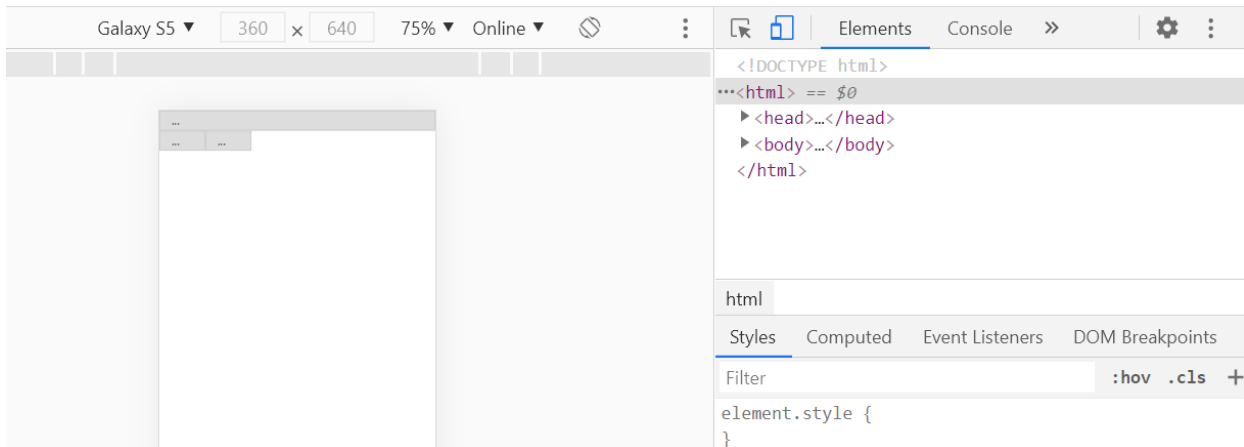
</div>

```

```
</div>
```

Percebam que especificamos **sm** apenas na primeira coluna.

Atualizando a página em inspecionar elemento em celular, fica assim:



Observem que a coluna com **sm** quebra e as outras duas com tamanho 2 fixo de acordo com o espaço disponível.

Alterando novamente o código:

```

<div class="container">

  <div class="row">

    <div class="col-sm">...</div>

    <div class="col">...</div>

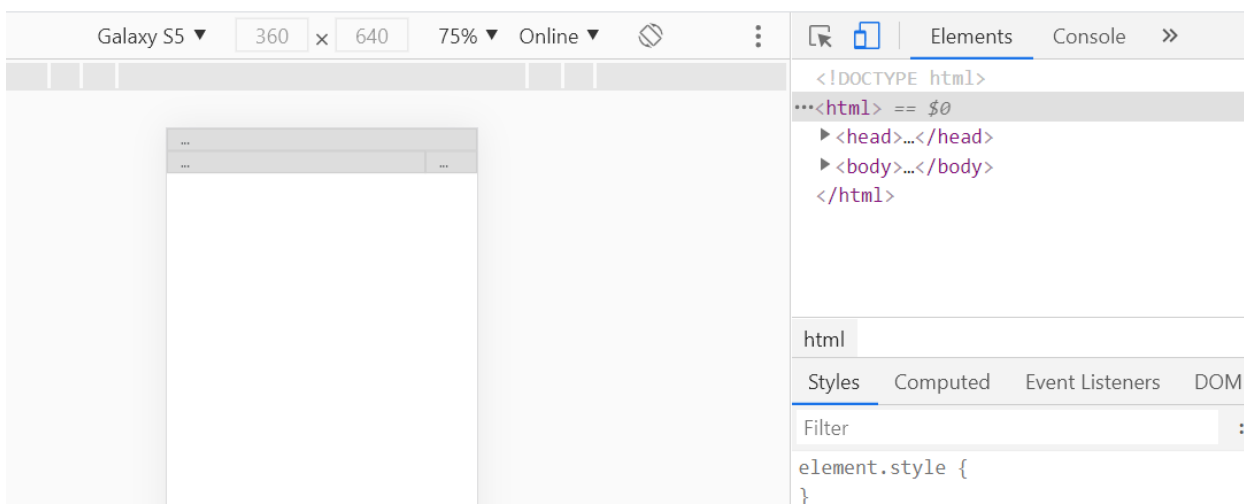
    <div class="col-2">...</div>

  </div>

</div>

```

```
</div>
```



Observem que a primeira coluna tem uma limitação mínima e segunda coluna está livre e a terceira, tamanho fixo.

Dessa forma se monta o layout como se quer que apareça um computador, tablet ou celular.

Uma DICA, para você descobrir a melhor forma de organizar o layout quando você estiver fazendo um sistema, um site ou um jogo é testando.

Grid - Ordenando Colunas

Ordenação de colunas é uma das novidades do bootstrap 4, que não tinha nas anteriores e funciona como uma mágica assim se dizendo. Se fossemos fazer isso com css estaríamos passando muito trabalho. Com bootstrap 4 conseguimos fazer tranquilamente.

Para iniciar vamos ordenar de forma diferente as colunas independentes da ordem que elas estão no código.

Arquivo index.php

```
<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>

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

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

    <title> Projeto Bootstrap 4 </title>

    <style type="text/css">

        .row [class^=col-], .row .col{

            background-color:#DDD;

            border:1px solid #ccc;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="row">

            <div class="col">Primeira</div>

            <div class="col">Segunda</div>

            <div class="col">Terceira</div>

        </div>

    </div>

</body>

</html>
```



```

    </div>

    <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

    <script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>

```

Atualizando a página fica assim:

Primeira	Segunda	Terceira
----------	---------	----------

Percebam que agora aparecem cada coluna com identificação no texto para melhor compreensão do que vamos fazer.

Para iniciar vamos ordenar de forma diferente as colunas independentes da ordem que elas estão no código.

Exemplo: Como fazemos para que a terceira coluna fique no começo da linha? Para fazer essa ordenação vamos colocar nessa terceira coluna uma classe chamada **order-first**. Veja no código a seguir onde está em vermelho:

```

<div class="container">

    <div class="row">

        <div class="col">Primeira</div>

        <div class="col">Segunda</div>

        <div class="col order-first">Terceira</div>

    </div>

</div>

```

Atualizando a página fica assim:

Terceira	Primeira	Segunda
----------	----------	---------

Agora altero o código novamente, para fazer com que a segunda coluna fique por último:

```

<div class="container">

    <div class="row">

        <div class="col">Primeira</div>

        <div class="col order-last">Segunda</div>

        <div class="col">Terceira</div>

    </div>

</div>

```

Atualizando a página fica assim:

Primeira	Terceira	Segunda
----------	----------	---------

Mas não é só dessa forma que podemos fazer ordenação. Vamos adicionar mais duas colunas no nosso código:

```
<div class="container">

  <div class="row">

    <div class="col">Primeira</div>

    <div class="col">Segunda</div>

    <div class="col">Terceira</div>

    <div class="col">Quarta</div>

    <div class="col">Quinta</div>

  </div>

</div>
```

Atualizando a página fica assim:

Primeira	Segunda	Terceira	Quarta	Quinta
----------	---------	----------	--------	--------

Vamos agora colocar uma ordem específica em uma das colunas, por exemplo na quarta coluna. Vou dizer que a ordem dela vai ser 1, ou seja, a primeira que vai aparecer. Quando colocamos isso vai fazer baseado nas outras colunas que estão aparecendo no sistema. Veja o código a seguir:

```
<div class="container">

  <div class="row">

    <div class="col">Primeira</div>

    <div class="col">Segunda</div>

    <div class="col">Terceira</div>

    <div class="col order-1">Quarta</div>

    <div class="col">Quinta</div>

  </div>

</div>
```

Atualizando a página fica assim:

Primeira	Segunda	Terceira	Quinta	Quarta
----------	---------	----------	--------	--------

Observem que a quarta coluna não ficou na posição 1, mas sim na última. Por que isso acontece? Por que as outras colunas não tem ordem específica e aí as faz virem por primeiro. As desordenadas vêm primeiras quando colocamos ordem com números e só depois vêm às ordenadas. Ou seja, para que isso funcione temos que colocar ordens em todas elas ou nas que queremos que sejam ordenadas.

se colocarmos assim:

```
<div class="container">

  <div class="row">
```

```

        <div class="col order-1">Primeira</div>

        <div class="col order-2">Segunda</div>

        <div class="col order-3">Terceira</div>

        <div class="col order-4">Quarta</div>

        <div class="col order-5">Quinta</div>

    </div>

</div>

```

Atualizando a página fica assim:

Primeira	Segunda	Terceira	Quarta	Quinta
----------	---------	----------	--------	--------

Agora se quisermos colocar a quarta coluna no começo fizemos assim:

```

<div class="container">

    <div class="row">

        <div class="col order-4">Primeira</div>

        <div class="col order-2">Segunda</div>

        <div class="col order-3">Terceira</div>

        <div class="col order-1">Quarta</div>

        <div class="col order-5">Quinta</div>

    </div>

</div>

```

Atualizando a página fica assim:

Quarta	Segunda	Terceira	Primeira	Quinta
--------	---------	----------	----------	--------

Ou seja, invertemos o processo a quarta foi para o lugar da primeira e a primeira para o lugar da segunda. Podemos mudar a ordem do jeito que quisermos independente de como elas estão no código.

Observem o exemplo a seguir:

```

<div class="container">

    <div class="row">

        <div class="col">Primeira</div>

        <div class="col">Segunda</div>

        <div class="col order-2">Terceira</div>

        <div class="col order-1">Quarta</div>

        <div class="col order-3">Quinta</div>

    </div>

</div>

```

Atualizando a página fica assim:

Primeira	Segunda	Quarta	Terceira	Quinta
----------	---------	--------	----------	--------

Percebam que as duas primeiras não tem ordem, e por isso vem por primeiro e depois vem a quarta que está ordenada em 1, seguida das outras duas conforme ordenação. se quer que todas fiquem ordenadas temos que especificar em todas elas.

Com esses exemplo acabamos de ver as duas formas de ordenar. Percebam que order-last ou order-first vai pra realmente onde especificamos , independente das outras não terem uma posição definida.

Essas são ótimas formas de ordenar colunas independente de como elas estão no código que é uma novidade do bootstrap 4.

Grid - JustifyContent e AlignItems

Vamos ver agora as duas formas de alinhar as colunas. Existe o alinhamento de acordo com a direção que estamos alinhando elas e existe o alinhamento inverso a elas. Se vocês já estudaram o flexbox em css, é a mesma coisa, só que aqui é com o bootstrap 4.

Vamos começar com 3 colunas:

Primeira	Segunda	Terceira
----------	---------	----------

E vamos diminuir o tamanho delas porque se alinharmos ela como estão tomando todo espaço da tela, automaticamente tomam o espaço todo da tela o que dificulta a visualização e dificulta o entendimento da explicação.

Vamos por primeiro, colocar um tamanho fixo nelas como 3/12, ou seja, três doze avos . Veja o código a seguir:

Arquivo index.php

```
<!doctype html>

<html>

    <head>

        <meta charset="UTF-8"/>

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

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


    <title> Projeto Bootstrap 4 </title>

    <style type="text/css">

        .row [class^=col-], .row .col{

            background-color:#DDD;

            border:1px solid #ccc;
```

```

    }

</style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-3">Primeira</div>
            <div class="col-3">Segunda</div>
            <div class="col-3">Terceira</div>
        </div>
    </div>

    <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>
    <script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>
</html>

```

Atualizando a página fica assim:

Primeira	Segunda	Terceira
----------	---------	----------

Percebam que diminuíram.

Agora no css da página vamos acrescentar:

```

.row {
    background-color:#FF9999;
    padding: 10px;
}

```

Atualizando a página fica assim:

Primeira	Segunda	Terceira
----------	---------	----------

Para melhor visualização total da área.

Agora, se quisermos que essas 3 colunas inicie do lado esquerdo, onde começa a exibição dos elementos como está, então colocamos na tag da linha justify-content-start. Por padrão já começa alinhamento da esquerda ou acima.

```

<div class="row justify-content-start">

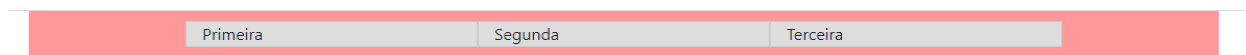
```

Atualizando a tela, percebam que nada muda, por que esse é o padrão de alinhamento da esquerda ou acima.

Se quisermos que inicie no meio da linha, colocamos:

```
<div class="row justify-content-center">
```

Atualizando a página fica assim:

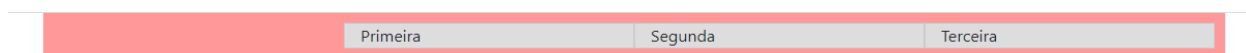


Como temos um item ao lado do outro na horizontal, o justify-content vai surgir efeito também na horizontal.

Se quisermos que inicie na direita, no fim da exibição, colocamos justify-content-end:

```
<div class="container">
  <div class="row justify-content-end">
    <div class="col-3">Primeira</div>
    <div class="col-3">Segunda</div>
    <div class="col-3">Terceira</div>
  </div>
</div>
```

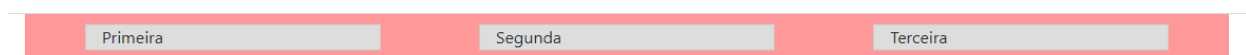
Atualizando a página fica assim:



Se quisermos que elas se espalhem de forma que o espaço entre uma e outra seja idêntico, colocamos justify-content-around. Around significa em volta.

```
<div class="row justify-content-around">
```

Atualizando a página fica assim:



Se quisermos espalhar eles não ao redor, mas entre um e outro colocamos justify-content-between.

```
<div class="row justify-content-between">
```

Atualizando a página fica assim:



Vamos agora fazer uma modificação no css, colocando uma altura na linha de 500px e na coluna 50px. E vamos tirar o deixar a linha sem o justify-content, ficando assim o código completo:

Arquivo index.php

```
<!doctype html>
<html>
```

```
<head>

<meta charset="UTF-8"/>

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

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


<title> Projeto Bootstrap 4 </title>

<style type="text/css">

    .row [class^=col-], .row .col{

        background-color:#DDD;

        border:1px solid #ccc;

        height:50px;

    }

    .row {

        background-color:#FF9999;

        padding: 10px;

        height: 500px;

    }

</style>

</head>

<body>

    <div class="container">

        <div class="row">

            <div class="col-3">Primeira</div>

            <div class="col-3">Segunda</div>

            <div class="col-3">Terceira</div>

        </div>

    </div>

    <script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

    <script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>
```

Atualizando a página fica assim:



Agora como fizemos para alinhar esses itens na vertical? Para o alinhamento na vertical usamos o `align-items`. Vamos colocar alinhados no começo, verticalmente da página como já estão conforme imagem acima. Para isso colocamos na linha

```
<div class="row align-items-start">
```

Vamos agora colocar no fim da página, para isso usamos `align-items-end`:

```
<div class="row align-items-end">
```

Atualizando a página fica assim:



Para colocar centralizado na vertical, colocamos `align-items-center`:

```
<div class="row align-items-center">
```

Atualizando a página fica assim:



Para alinhar tanto no centro na vertical quanto na horizontal fizemos assim:

```
<div class="row align-items-center justify-content-center">
```


A ordem aqui não importa dos atributos acima.

Atualizando a página fica assim:



E assim podemos fazer da forma que quisermos a colocação dos elementos na tela. Podemos combinar várias linhas de forma que uma fique no lado direito ou na esquerda e assim por diante, basta utilizar a imaginação.

Para treinar, sugiro que monte conteúdos com várias linhas e várias colunas, tentando montar estruturas com layout baseados no que aprenderam até agora.

Media Component

Esse recurso é utilizado quando vamos fazer um sistema de comentários. Precisamos fazer para isso uma estrutura que por exemplo apareça a foto da pessoa, um título que geralmente é o nome dessa pessoa e abaixo um comentário, listagens de coisas, esse tipo de estrutura que precisamos fazer. Inspirado no Facebook o Bootstrap criou uma estrutura pronta para utilizar em casos que precisamos desse tipo de procedimento. Vamos fazer agora no arquivo index.php e o nome desse recurso é media.

Arquivo index.php

```
<!doctype html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="UTF-8"/>
```

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

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

```
  <title> Projeto Bootstrap 4 </title>
```

```
<style type="text/css">
```

```
.avatar{  
    width:50px;  
    height:50px;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

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

```
    <div class="media">
```

```
        
```

```
        <div class="media-body">
```

```
            <h5>Algum tipo de título</h5>
```

<p>O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .media e a .media-body envolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento . </p>

```
        </div>
```

```
    </div>
```

```
</div>
```

```
<script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>
```

```
<script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>
```

```
</body>
```

```
</html>
```

Atualizando a página fica assim:



Algum tipo de título

O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .media e a .media-body envolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento .

Com isso foi feita uma estrutura mínima com a classe media, media-body e o conteúdo na tag p, que não envolvem muitas linhas para configurações de resultados desse tipo de estrutura.

Podemos também fazer estruturas desse tipo no formato de listas. Observem as alterações feitas no arquivo index.php

```
<!doctype html>
```

```
<html>
```

```
    <head>
```

```
        <meta charset="UTF-8"/>
```

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

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

```
        <title> Projeto Bootstrap 4 </title>
```

```
    <style type="text/css">
```

```
        .avatar{
```

```
            width:50px;
```

```
            height:50px;
```

```
        }
```

```
    </style>
```

```
    </head>
```

```
    <body>
```

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

```
            <ul class="list-unstyled">
```

```
                <li class="media">
```

```
                    
```

```
                    <div class="media-body">
```

```
                        <h5>Algum tipo de título</h5>
```

<p>O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento . </p>

</div>

<li class="media">

<div class="media-body">

<h5>Algum tipo de título</h5>

<p>O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento . </p>

</div>

</div>

<script type="text/javascript" src="assets/js/jquery-3.5.1.min.js"/></script>

<script type="text/javascript" src="assets/js/bootstrap.bundle.min.js"/></script>

</body>

</html>

Foi criado uma lista não ordenada com a classe list-unstyled, essa classe tira todos os estilos de listas. E após duplicada essa lista.

Atualizando a página fica assim:



Algum tipo de título

O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento .



Algum tipo de título

O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento .

Automaticamente já fica pronto para usar fazendo dessa forma. Também podemos colocar alinhamentos específicos nesse itens, por exemplo a imagem queremos que fique meio dos comentários, no sentido vertical, para isso basta ir na tag img e colocar na classe dessa imagem é digitar **align-self-center**, que significa efeitos que serão aplicados dentro desse elemento em foco que é a imagem.

```

```

Atualizando a página fica assim:

Algum tipo de título



O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento .



Algum tipo de título

O objeto de mídia ajuda a construir componentes complexos e repetitivos, onde alguma mídia é posicionada ao lado do conteúdo que não envolve essa mídia. Além disso, ele faz isso com apenas duas classes obrigatórias, graças ao flexbox. Abaixo está um exemplo de um único objeto de mídia. Apenas duas classes são necessárias - a envolvente .mediae a .media-bodyenvolvente do seu conteúdo. Preenchimento e margem opcionais podem ser controlados por meio de utilitários de espaçamento .

Se colocarmos align-self-end, fica parte inferior da base do comentário.

```

```

Se colocarmos align-self-start, fica alinhado na parte superior, ou seja, é a padrão se não especificar alinhamento nenhum.

```

```

Tipografia

Vamos ver alguns atalhos de tipografia que pode ser feito. Existe muitos, basta consultar a documentação do Bootstrap que tem várias opções. Então nessa aula vamos ver as principais. Por exemplo pode ser feitos títulos de páginas ou dar destaque a um determinado texto, como sendo um título, uma referência, algum cabeçalho de alguma coisa. Normalmente usamos as tags para títulos (h1...h6), pois em termos de busca, SO, Google, em termos de leitura da página por motores de busca as tags de título como o h1 é a primeira coisa que vai pesquisar, procurar, por que se refere ao título principal da página que está aberta. Só que por exemplo em outros locais da página podemos ter outros títulos de destaque, mas não de maior destaque do que foi usado no título que foi usado h1 por exemplo. Para isso podemos fazer de uma outra maneira, que é utilizando uma classe chamada h1 do próprio Bootstrap, veja exemplo:

```
<p class="h1"> Algum titulo qualquer </p>
```

```
<h1> Algum titulo qualquer </h1>
```

Atualizando a página fica assim:

Algum titulo qualquer

Algum titulo qualquer

No bootStrap tem a classe h1, h2, até h6. Esse é mais um meio quando queremos dar destaque a um titulo, mas sem afetar o processo a busca dos motores de reenderização na página.

Existe uma outra técnica também relacionada a titulos, cabeçalhos e textos em destaques, que é o seguinte, vamos supor que por exemplo tenho uma página de cadastro de usuários, ou seja, uma área de usuários e quero por dentro dessa área de usuários o texto um pouco maior ou um diferencial, tipo assim:

```
<h3>Usuário <small>Adicionar usuário</small> </h3>
```

Atualizando a página fica assim:

Usuário Adicionar usuário

Podemos perceber um diferencial nesse texto mas ainda se confunde um pouco. Então podemos utilizar a classe classe text-muted na tag small para realçar mais esse diferencial conforme código a seguir:

```
<h3>Usuário <small class="text-muted">Adicionar usuário</small> </h3>
```

Atualizando a página fica assim:

Usuário Adicionar usuário

Assim, podemos fazer uma diferenciação melhor. Para isso usamos esses recursos próprios do bootStrap que já tem pronto, para não precisarmos estar criando uma diferenciação de cores.

Outro recurso, por exemplo, temos um parágrafo um pouco maior:

```
<p>Documentação e exemplos para a tipografia Bootstrap, incluindo configurações globais, cabeçalhos, listas, texto do "<body>" e outros. O Bootstrap configura estilos básicos para display, tipografia e links, globalmente. Quando precisar de mais controle, confira as classes utilitárias de texto.</p>
```

E aí, queremos colocar um destaque a esse texto como se fosse um marcador em alguma parte desse texto. Para isso usamos por exemplo a tag span e dentro colocamos a classe mark, veja a seguir:

<p>Documentação e exemplos para a tipografia Bootstrap, incluindo ****configurações globais, cabeçalhos, listas, texto do "<body>" e outros. **** O Bootstrap configura estilos básicos para display, tipografia e links, globalmente. Quando precisar de mais controle, confira as classes utilitárias de texto.</p>

Atualizando a página fica assim:

Documentação e exemplos para a tipografia Bootstrap, incluindo configurações globais, cabeçalhos, listas, texto do "" e outros. O Bootstrap configura estilos básicos para display, tipografia e links, globalmente. Quando precisar de mais controle, confira as classes utilitárias de texto.

Essa classe realça com um fundo marcado.

Outro recurso importante por exemplo temos uma classe própria dentro de uma div e essa div vai ter um tamanho fixo:

<div class="teste">

Algum texto.

</div>

Atualizando a página fica assim:

Algum texto.

Se quisermos alinhar o texto dessa div a direita, usamos a a classe text-right, veja a seguir:

<div class="teste text-right">

Algum texto.

</div>

Atualizando a página fica assim:

Algum texto.

Se quisermos colocar esse texto no centro, usamos a classe text-center. São detalhes que podemos consultar a própria documentação do bootStrap e ver outros recurso que podemos utilizar desse tipo.

Temos outro recurso importante que é de listas, por exemplo quando queremos que itens fique um do lado do outro horizontalmente:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

Atualizando a página fica assim:

- Item 1
- Item 2
- Item 3

Para alinhar cada item um do lado do outro, usamos a classe list-inline-item e para retirar o simbolo de cada item a classe list-inline. Veja a seguir:

```
<ul class="list-inline">
  <li class="list-inline-item">Item 1</li>
  <li class="list-inline-item">Item 2</li>
  <li class="list-inline-item">Item 3</li>
</ul>
```

Atualizando a página fica assim:

Item 1 Item 2 Item 3

Com isso, usando próprios recursos do bootStrap já organizamos os elementos sem uso do CSS.

Imagens

Vamos ver alguns recursos do bootStrap em relação as imagens, recursos esses muito utilizados na criação de sistema de sites. Vamos começar com o recurso que é com certeza mais utilizado que é transformar a imagem adaptável, responsiva. Então como fazer com que uma imagem fique responsivel ao local onde está sendo aberta? Para isso utilizamos a classe img-fluid.

```

```


Com essa classe a imagem se adapta ao tamanho do espaço em que estiver. Sempre é bom observar o resultado na página antes sem e após do uso dessa classe para perceberem bem o recurso esse do bootStrap.

Se por exemplo criarmos uma div e colocar essa imagem dentro, e nessa div colocar um estilo css local, e colocarmos nesse estilo uma largura padrão fixa de 500px conforme exemplo abaixo:

```
<div style="width:500px">  
      
</div>
```

Essa imagem vai ficar da largura que especificamos nesse estilo css local de 500px. Essa classe img-fluid faz esse ajuste de acordo com esse espaço.

Agora, se temos uma imagem de tamanho menor, e queremos criar uma imagem no formato mais pra miniaturas(thumbnail) tipo foto 3 x 4 que geralmente tem uma borda em volta dela, o bootStrap tem uma classe chamada img-thumbnail que faz isso, sem precisarmos usar um estilo css.

```

```

São recursos que nos ajudam muito na hora de fazermos efeitos desse tipo.

Se quisermos deixar essa foto com as bordas no formato arredondado, usamos a classe rounded.

```

```

Podemos também misturar, além da classe rounded podemos deixar tipo fluid:

```

```

Desse jeito a imagem além de responsável ficou com as bordas arredondadas. Além disso podemos com uma imagem menor, colocá-la a direita:

```

```

Além desses recursos como rounded, thumbnail, img-fluid, temos o chamado figure, que é quando temos uma imagem e uma legenda embaixo, com css temos varias formas de fazer, mas o bootStrap já tem uma estrutura pre pronta que nos possibilita fazer sem uso do css. Para isso usamos a tag figure e dentro dela fizemos as configurações na tag img de acordo como vamos querer o efeito nessa imagem e após com a tag figcaption para aplicação da nossa legenda na imagem:

```
<figure class="figure">  
      
    <figcaption class="figure-caption"> Leão Africano</figcaption>  
</figure>
```

Atualizando a página fica assim:



Leão Africano

Observem na legenda a letra já fica com um formato de tom diferente, fazendo parte dessa estrutura toda.

Para colocar a legenda a direita só acrescentamos a classe na tag figure, a classe text-right:

```
<figcaption class="figure-caption text-right"> Leão Africano </figcaption>
```

O padrão dessa estrutura toda na tag figure seria esse:

```
<figure class="figure">
    
    <figcaption class="figure-caption"> Leão Africano </figcaption>
</figure>
```

Essas foram as principais classe para imagens do bootStrap, muito utilizada para formatações do dia-a-dia. E com certeza serão muito utilizada na questão de transformar as imagens em responsivas.

Tabelas

Vamos ver os recursos que o bootStrap tem para as tabelas. Primeiro, vamos fazer a tabela conforme mostra o código a seguir:

```
<div class="container">
    <table>
        <tr>
            <th>#</th>
```

```

        <th>Nome</th>
        <th>Sobrenome</th>
        <th>Idade</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Sandra</td>
        <td>Amorim</td>
        <td>60</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Carla</td>
        <td>Santos</td>
        <td>10</td>
    </tr>
    <tr>
        <td>3</td>
        <td>Marilda</td>
        <td>Fonseca</td>
        <td>80</td>
    </tr>
</table>
</div>

```

Atualizando a página fica assim:

Nome Sobrenome Idade

1 Sandra Amorim 80

2 Carla Santos 80

3 Marilda Fonseca 80

Podem ver que por enquanto não colocamos nenhuma formatação ainda nessa tabela.

Vamos agora começar usar os recursos para tabela, começando pela classe table na tag table:

```
<table class="table">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Observem o efeito que fez na tabela somente com uso da classe table na tag, só isso já faz a tabela ficar organizada. Vamos ver outros detalhes que o bootStrap nos possibilita.

Vamos agora deixar a tabela escura, ou seja, inverter as cores dessa tabela. Vamos colocar na tag table, além da classe table a classe table-dark:

```
<table class="table table-dark">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Observem, com a classe table-dark a tabela ficou invertida. As letras ficaram brancas e o fundo preto.

Outro recurso importante é o table-striped , que se utiliza muito em formulários, em listagens que se tem muitos itens, assim ajuda na identificação dos itens com uma alternância de cores:

```
<table class="table table-striped">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Observem que automaticamente coloca cor sim cor não em toda a tabela.

Outro recurso também é o table-bordered:

```
<table class="table table-bordered">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Table-bordered, aplica borda em toda a tabela, linhas e colunas. Podemos também combinar essas classes como:

```
<table class="table table-bordered table-striped">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Ou ainda:

```
<table class="table table-bordered table-striped table-dark">
```

Combinar todas as classe.

Podemos ainda usar um recurso que no momento que o usuário do sistema passar o mouse em cima da linha fica destacada. Para que isso ocorra usamos a classe table-hover:

```
<table class="table table-hover">
```

No momento que usuário passar o mouse em cima de uma determinada linha ela se diferencia com um fundo de cor diferente.

Podemos também usar um recurso que possibilita fazermos um diferencial no cabeçalho dessa tabela, para isso criamos um tag thead envolvendo o cabeçalho e outra tag chamada tbody envolvendo o corpo da tabela. Então tudo que tiver dentro da tag tbody vai ser o conteúdo dessa tabela e dentro da tag thead vai ser o cabeçalho.

Feito isso se executarmos a página pouca coisa muda na tabela, apenas uma linha com grossa na parte inferior do cabeçalho.

Para diferenciar o cabeçalho do corpo da página usamos a classe thead-dark na tag thead.

```
<table class="table">
```

```
  <thead class="thead-dark">
```

```
    <tr>
```

```
      <th>#</th>
```

```
      <th>Nome</th>
```

```
      <th>Sobrenome</th>
```

```

        <th>Idade</th>

    </tr>

</thead>

```

...

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Agora sim, ficou destacado o cabeçalho. ainda podemos colocar assim:

```
thead class="thead-light">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Colocando essas classes podemos ver melhor a divisão do cabeçalho com o conteúdo da tabela.

Podemos usar na tag table a classe table-hover que dará uma destaque melhor quando o usuário passar o mouse em cima da tabela toda, mas ficando o cabeçalho com diferencial.

```

<table class="table table-hover">

    <thead class="thead-light">

        <tr>

            <th>#</th>

            <th>Nome</th>

            <th>Sobrenome</th>

            <th>Idade</th>

        </tr>

    </thead>

```

...

Se trocarmos essa classe por table-striped, colocamos cor sim cor não:

```
<table class="table table-striped">
```

Se quisermos fazer com que a tabela apareça mais compacta, colocamos a classe table-sm (sm de small), fazendo com que diminua os espaços entre os conteúdos da tabela.

```
<table class="table table-striped table-sm">
```

Atualizando a página fica assim:

#	Nome	Sobrenome	Idade
1	Sandra	Amorim	60
2	Carla	Santos	10
3	Marilda	Fonseca	80

Outro recurso, que é tornar a tabela responsiva, pois geralmente quando se tem uma tabela com muitas colunas, ela num determinado momento vai ser condençada de forma tornar ilegível. Então podemos tornar essa tabela responsivel de modo quando chegar num determinado mínimo ela crie uma barra de rolagem horizontal só nela mesma, não na página inteira. Que dessa forma podemos ter acesso as informações sem prejudicar as próprias informações. Como podemos fazer com que fique responsiva nossa tabela? Antes da tag table, onde começa a tabela na página, colocamos uma div e nela colocamos uma classe chamada table-responsive e no final dela obviamente fechamos essa div:

```
<div class="table-responsive">

    <table class="table table-striped table-sm">

        ...

    </table>

</div>
```

Com isso tornamos nossa tabela responsiva. Quando abriremos a tabela num dispositivo tablet, celular podemos perceber melhor, ainda mais se tiver muitas colunas a nossa tabela.

Alertas ou Avisos

Outro recurso que vamos usar muito é o sistema de alerta ou aviso. Por exemplo, estamos fazendo um sistema de login, e o usuário errou o login e então podemos mostrar um aviso que o usuário errou por exemplo e-mail e senha. Ou estamos inserindo uma determinada coisa e retorna mensagem de inserido com sucesso tipo assim. Mostrando para o usuário o que ele pretendia fazer foi realizado ou não foi possível ser feita. Para fazer esse tipo de aviso com o bootStrap é muito fácil e prático.

Para fazer esse tipo de recurso criamos uma div e nela colocamos a classe alert. Podemos usar só alert ou junto o tipo de alerta que se quer. Por exemplo, vamos colocar junto a essa classe o tipo padrão que é alert-primary e a seguir a ela colocamos outro recurso que é role="alert" que diz para o JS do bootStrap que isso se trata de um alerta.

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

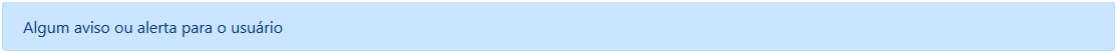
```
<div class="alert alert-primary" role="alert">
```

Algum aviso ou alerta para o usuário

```
</div>
```

```
</div>
```

Atualizando a página fica assim:

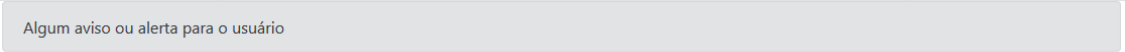


Algum aviso ou alerta para o usuário

Existe vários outros tipos de tematização desses alertas. No exemplo acima foi usado o padrão, mas temos um outro tipo que é o alert-secondary:

```
<div class="alert alert-secondary" role="alert">
```

Atualizando a página fica assim:



Algum aviso ou alerta para o usuário

Temos também o alert-success:

```
<div class="alert alert-success" role="alert">
```

Atualizando a página fica assim:




Algum aviso ou alerta para o usuário

Do tipo quando recebemos um tipo de aviso que alguma coisa deu certo.

Existem também o alert-danger:

```
<div class="alert alert-danger" role="alert">
```

Atualizando a página fica assim:



Algum aviso ou alerta para o usuário

Que é o vermelho, normalmente quando alguma coisa deu errado.

Existe também o alert-warning:

```
<div class="alert alert-warning" role="alert">
```

Atualizando a página fica assim:

Algum aviso ou alerta para o usuário

Alerta que não impede de ter acontecido e ao mesmo tempo tomar cuidado, observar, para prestar atenção.

Existe também o alert-info:

```
<div class="alert alert-info" role="alert">
```

Algum aviso ou alerta para o usuário

Fica em azul tipo primary, um pouco mais claro.

Existe o alert-light e o contrário desse que é o alert-dark.

```
<div class="alert alert-light" role="alert">
```

Algum aviso ou alerta para o usuário

Existe o alert-dark.

```
<div class="alert alert-dark" role="alert">
```

Algum aviso ou alerta para o usuário

Existe vários tipo, mas o que mais vamos usar vai ser o alert-success e o alert-danger, ou seja, o que deu certo ou errado.

Podemos também colocar titulos nesses aviso, o que torna um pouco maior e chamar a atenção.

Por exemplo:

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

```
  <div class="alert alert-danger" role="alert">
```

```
    <h4 class="alert-heading"> Deu errado! </h4>
```

```
    E-mail e/ou senha incorretos
```

```
  </div>
```

```
</div>
```

Ou seja, colocamos na div o alert-danger e a tag h4 a classe alert-heading, que é o cabeçalho do aviso.

Deu errado!

E-mail e/ou senha incorretos

O mesmo vale para todos os tipo de alerta.

Podemos também colocar um " x " nesses alerta para o usuário poder clicar nele e fechar esse alerta.

Exemplo:

```
<div class="alert alert-danger alert-dismissible" role="alert">  
    E-mail e/ou senha incorretos  
    <button class="close" data-dismiss="alert" aria-label="Fechar">  
        <span aria-hidden="true">&times;</span>  
    </button>  
</div>
```

Usamos mais uma classe chamada alert-dismissible na div (dismissible = significa estar passivo de desaparecer) que o alerta pode desaparecer caso aconteça alguma coisa, e após a mensagem colocamos a tag button com a classe close, que vai ajudar na formatação desse botão o data-dismiss="alert" vai dizer para o bootStrap que tipo de ação vai ocorrer, ou seja, o que ele vai fechar quando clicar nesse botão. E dentro da tag span o × é o x que o bootStrap imprime no aviso.



Para não fechar esse aviso quando o usuário clicar no x, de modo seco, assim dizendo, podemos adicionar mais uma classe nessa div chamada fade show, percebe um fechamento mais suave.

```
<div class="alert alert-danger alert-dismissible fade show" role="alert">
```

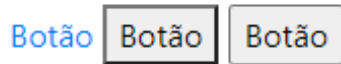
As classe fade show e aria-label="Fechar" não são de uso obrigatórios.

Botões e Grupo de Botões

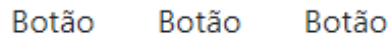
Um dos recurso que mais vamos utilizar com certeza, fora os grids são os botões. Existe infinitas formas de fazermos botões e grupos de botões, o que vamos ver a partir de agora. O bootStrap tem uma classe que basicamente vai padronizar a criação de botões. Então podemos ter botões que são links e temos botões que podem ser a tag button ou botões que são uma tag input type="submit". Então independente do tipo de tag que utilizamos, essa classe do bootStrap vai deixar todos identicos. Observem a aplicação dessas 3 formas a seguir:

```
<div class="container">  
  
</br>  
    <a href="#"> Botão </a>  
    <button> Botão </button>  
    <input type="submit" value="Botão"/>  
  
</div>
```

Executando o código temos um link normal e os dois últimos botões o button e o submit no mesmo formato.



Agora se colocarmos a classe chamada btn em cada um desses botões, eles todos ficaram esteticamente iguais:



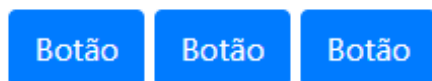
Se colocarmos mais uma classe em cada um, especificando o tipo visual de botão, por exemplo existe a classe primary que é o padrão.

```
<a class="btn btn-primary" href="#"> Botão </a>
```

```
<button class="btn btn-primary"> Botão </button>
```

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

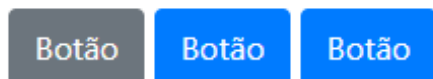
Atualizando a página percebem que todos ficaram iguais.



Percebem que agora olhando cada um desses botões não identificamos se é um link ou botão, todos com aparência idêntica, graças ao tipo de estilização, nesse colocamos o primary, mas existe outros por exemplo o secondary:

```
<a class="btn btn-secondary" href="#"> Botão </a>
```

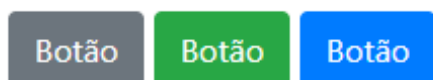
fica mais acinzentados:



Existe também o success que é de cor verde:

```
<button class="btn btn-success"> Botão </button>
```

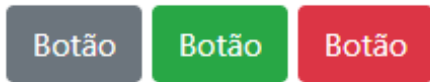
que fica assim:



Temos o danger com a cor vermelha:

```
<input class= "btn btn-danger" type="submit" value="Botão"/>
```

que fica assim:



Temos o warning, o info, light, o dark.

Mas se queremos imitar um link temos o link:

```
<input class= "btn btn-link" type="submit" value="Botão"/>
```

que fica assim:



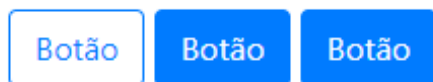
Observem no último desses botões, quando o usuário passar o mouse sobre ele fica igual ao um link.

Com isso percebem que independente do tipo de tag, basta colocar uma dessas especificações para terem um mesmo aspecto visual.

Existe também o btn-outline-primary que é botão só com bordas, sem fundo. Só fica com cor de fundo quando o usuário passar o mouse sobre ele.

```
<a class="btn btn-outline-primary" href="#"> Botão </a>
```

Fica assim:

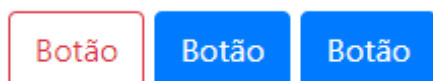


Observem o primeiro botão com esse recurso.

Tem também o danger, com cor vermelha:

```
<a class="btn btn-outline-danger" href="#"> Botão </a>
```

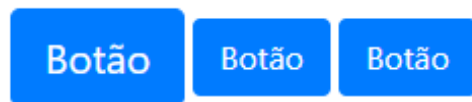
Fica assim:



Podemos alterar o tamanho dos botões, colocando btn-lg. Que é um botão maior:

```
<a class="btn btn-primary btn-lg" href="#"> Botão </a>
```

Fica assim:

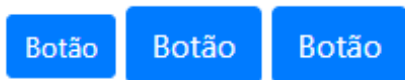


Observem o primeiro botão com essa estilização.

Temos também o btn-sm que é menor que o padrão:

```
<a class="btn btn-primary btn-sm" href="#"> Botão </a>
```

Fica assim:

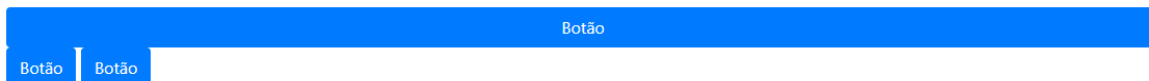


Observem esse efeito no primeiro botão, os outros dois estão no padrão que é o primary.

Temos também o btn-block, que torna o botão estendida a toda área do objeto. Abrange todo o espaço disponível.

```
<a class="btn btn-primary btn-block" href="#"> Botão </a>
```

Fica assim:

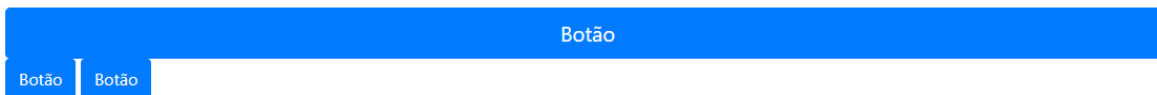


Observem como ficou o primeiro botão com essa estilização. Abrangeu todo o espaço disponível do container.

Podemos combinar essas classes, por exemplo vamos tornar esse botão no formato block e grande, aí colocamos o btn-lg:

```
<a class="btn btn-primary btn-block btn-lg" href="#"> Botão </a>
```

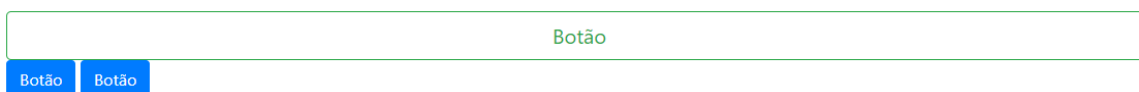
Fica assim:



Se quer menor, então btn-sm e assim sucessivamente. Queremos o outline:

```
<a class="btn btn-outline-success btn-block btn-lg" href="#"> Botão </a>
```

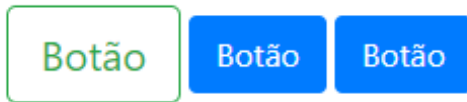
Fica assim:



Por exemplo, se tirarmos o btn-block:

```
<a class="btn btn-outline-success btn-lg" href="#"> Botão </a>
```

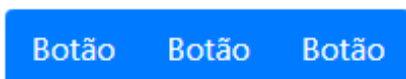
fica assim:



Vamos ver agora como agrupar botões. Os botões iram ficar todos juntos, colados um no outro, mas com uma borda indicando que são um grupo, para isso vamos envolver eles em uma div, e nessa div colocamos uma classe chamada btn-group, indicando que é um grupo de botões.

```
<div class="btn-group">  
  <a class="btn btn-primary" href="#"> Botão </a>  
  <button class="btn btn-primary"> Botão </button>  
  <input class="btn btn-primary" type="submit" value="Botão"/>  
</div>
```

Ficando assim:



Percebemos melhor quando colocado o mouse sobre eles. Podemos colocar quantos botões forem necessários e no primeiro e último botão coloca automaticamente a borda neles.

Por exemplo, podemos fazer mais de um grupo de botões, para isso duplicamos essa div da classe de grupos.

Ficando assim:

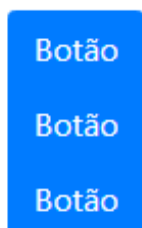


Assim que se faz as barras de ferramentas por exemplo, classificados por grupos.

Esses grupos eles não precisam ser só na horizontal, que é a padrão, mais utilizados, podemos deixar também na vertical. Para isso basta ir na classe da div e acrescentar a palavra vertical:

```
div class="btn-group-vertical">
```

Ficando assim:



Ficando um botão abaixo do outro. Assim podemos fazer um menu por exemplo.

Carousel (slideshow)

Com o bootstrap é um processo simples de se fazer. Para isso primeiro precisamos criar uma div para comportar todo nosso slideshow. Feito isso colocamos nessa div uma classe com indicando primeiro que vai ser um slide e depois um carousel. O bootstrap chama de carousel, o que pra nós é slideshow. O slide é o tipo de processo, transição que vai fazer. Dentro dessa div colocamos os próprios itens que vai ter nesse carousel. Então criamos uma div que vai englobar esses itens do carousel, e essa vai ter uma classe chamada carousel-inner, ou seja, os itens internos do carousel iram ser esses. Dentro dessa div, criamos outra div com a classe chamada carousel-item e dentro dela colocamos a tag img. E para essa foto pegar toda área do slideshow, colocamos uma classe nela chamada w-100, ou seja, 100%. Para exemplo vamos criar um carousel onde temos 4 fotos para nosso slideshow. Agora antes vamos definir qual a foto que vai ficar ativa assim que o carousel carregar na página. Para isso na tag img da foto que escolhermos que vai ser ativa criamos outra classe chamada de active.

Veja como fica o código completo do carousel:

```
<div class="slide carousel">

  <div class="carousel-inner">

    <div class="carousel-item active">

    </div>

    <div class="carousel-item">

    </div>

    <div class="carousel-item">

    </div>

    <div class="carousel-item">

    </div>

  </div>

</div>
```

Se executar o código acima, percebam que aparece a primeira foto que colocamos com a classe active, mas não temos o controle, ou seja, aquela seta que nos permite movimentar pro lado. Então para adicionar esse controle, depois da div com a classe carousel-inner, logo abaixo dela vamos colocar os controles, que são dois, um do lado esquerdo e outro do lado direito com a tag "a" e referenciar esse carousel, para isso colocamos na div do carousel uma id e atribuímos um nome para essa id:

```

<div class="container">

  <div id="slideshowExemplo" class="slide carousel">

    <div class="carousel-inner">

      <div class="carousel-item active">

      </div>

      <div class="carousel-item">

      </div>

      <div class="carousel-item">

      </div>

      <div class="carousel-item">

      </div>

    </div>

    //seta para esquerda

    <a class="carousel-control-prev" href="#slideshowExemplo" data-slide="prev">

      <span class="carousel-control-prev-icon"></span>

    </a>

    //seta para a direita

    <a class="carousel-control-next" href="#slideshowExemplo" data-slide="next">

      <span class="carousel-control-next-icon"></span>

    </a>

  </div>

</div>

```

Se executar o código já visualiza a seta para a esquerda e direita o que permite a visualização das 4 fotos nesse slideshow.

data-slide="prev" quer dizer a anterior, subentende que é o ícone do lado esquerdo que estamos referenciando. E dentro dessa tag a, colocamos a tag span que vai mostrar a seta para esquerda.

data-slide="nex" quer dizer o próximo

Se quisermos que apareça na parte inferior qual é a foto atual que está mostrando, aqueles quadradinhos ou bolinhas tipo botões de radio na parte inferior, fizemos os indicadores:

Acrescentamos no código o seguinte:

```
<div id="slideshowExemplo" class="slide carousel">

    <!-- Indicadores -->

    <ol class="carousel-indicators">

        <li data-target="#slideshowExemplo" data-slide-to="0" class="active"></li>

        <li data-target="#slideshowExemplo" data-slide-to="1"></li>

        <li data-target="#slideshowExemplo" data-slide-to="2"></li>

        <li data-target="#slideshowExemplo" data-slide-to="3"></li>

    </ol>

    <div class="carousel-inner">

        ...
```

Onde temos data-slide-to="0" e assim por diante, significa a posição da primeira foto, ou seja, vetor da posição zero e assim por diante.

class="active" que é a primeira foto ativa ao iniciar a página.

Para acessar outras funcionalidades sobre o carousel, basta acessar a documentação do bootstrap.

Formulários

Como exemplo abaixo temos um exemplo de formulário:

E-mail:

Senha:

Logar

Conforme código:

```
<div class="container">
  <form method="POST">
    E-mail: <br/><br/>
    <input type="email" name="email"/> <br/><br/>
    Senha: <br/><br/>
    <input type="password" name="senha"/> <br/><br/>
    <input type="submit" value="Logar">
  </form>

  <hr/>
</div>
```

Com recursos do bootstrap para formulários podemos deixar mais agradável visualmente esse formulário.

Para vermos a diferença dos recursos do bootstrap, vamos dividir com uma linha horizontal no código dessa página que contem esse formulário, e na parte inferior dessa linha vamos fazer o mesmo formulário, só usando técnicas diferentes do bootstrap.

Vamos agrupar o nome e o campo, o que chamamos de form-group, que é grupo de formulários.

```
<hr/>
<form method="POST">
<div class="form-group">
  <label for="email"> E-mail: </label>
```

```

        <input type="email" name="email" id="email" class="form-control" placeholder="E-mail"/>
    </div>
    <div class="form-group">
        <label for="senha">Senha: </label>
        <input type="password" name="senha" id="senha" class="form-control"
placeholder="Senha"/>
    </div>
    <div class="form-group">
        <input type="submit" value="Logar" class="btn btn-primary">
    </div>
</form>

```

Fica assim:

E-mail:

Senha:

Logar

E-mail:

E-mail

Senha:

Senha

Logar

Percebam a diferença do segundo formulário estilizado com o bootstrap. Aplicando cada nome e seu respectivo campo com a div com classe form-group, eliminamos as tags de quebra de linha, ou seja, tag br., não foi preciso utilizá-las. A tag label foi usada para tornar a campo acessível, por exemplo o usuário basta clicar no nome, ou rótulo do campo e seleciona o campo para digitar. Mas se não quiser usar a tag label, pode ser feita sem problemas. O próprio formulário já está responsivo.

Podemos também aumentar ou diminuir o tamanho dos elemento no formulário.

Por exemplo se queremos que o campo email fique maior, mais largo na altura, basta colocar a classe form-control-lg

```

<input type="email" name="email" id="email" class="form-control form-control-lg" placeholder="E-mail"/>

```

Agora queremos diminuir o campo senha, basta colocar a classe form-control-sm

```
<input type="password" name="senha" id="senha" class="form-control form-control-sm"
placeholder="Senha"/>
```

Fica assim:

E-mail:

Senha:

Agora vamos tirar essas duas classe que aumentou os dois campo para vermos outro recurso que é deixar o campo apenas leitura:

```
<input type="email" name="email" id="email" class="form-control" value="sandra-
famorim@gmail.com" readonly/>
```

Com a propriedade readonly, tornou o campo email não editável, no máximo o usuário só pode selecionar o campo.

Fica assim:

E-mail:

Senha:

Podemos também tornar esse campo como apenas um texto comum, para isso substituímos a classe form-control por form-control-plaintext

```
input type="email" name="email" id="email" class="form-control-plaintext" value="sandra-
famorim@gmail.com" readonly/>
```

Fica assim o campo email parecendo um texto comum.

E-mail:

sandra-famorim@gmail.com

Senha:

Se tirarmos o readonly, ele fica editável, mas quando tiramos o foco do campo ele fica parecendo um texto comum.

Podemos deixar os elementos do formulário inline, ou seja, uma ao lado do outro, para isso é necessário o formulário não ter tags label.

Para deixar os elementos ordenados lado a lados, basta irmos na tag form colocarmos a classe form-inline.

```
<form method="POST" class="form-inline">
```

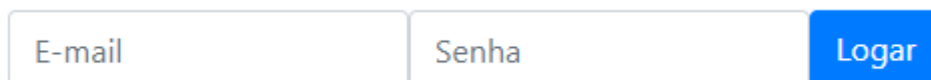
```
<div class="form-group">
```

```

        <input type="email" name="email" class="form-control" placeholder="E-mail"/>
    </div>
    <div class="form-group">
        <input type="password" name="senha" class="form-control" placeholder="Senha"/>
    </div>
    <div class="form-group">
        <input type="submit" value="Logar" class="btn btn-primary">
    </div>
</form>

```

Fica assim:



Nesse exemplo, foi tirado o rótulos dos campos, somente para ficar mais estilizados visualmente.

Combinado a isso podemos utilizar os grids, para deixar os campos cada um numa coluna por exemplo.

```

<form method="POST">
    <div class="form-row"> <!-- row = linha -->
        <div class="col"> <!-- col = coluna -->
            <div class="form-group">
                <input type="email" name="email" class="form-control" placeholder="E-mail"/>
            </div>
        </div>
        <div class="col"> <!-- col = coluna -->
            <div class="form-group">
                <input type="password" name="senha" class="form-control" placeholder="Senha"/>
            </div>
        </div>
    </div> <!-- Fim da primeira linha -->

```

```
<div class="form-row"> <!-- Inicio da segunda linha-->

  <div class="col">

    <div class="form-group">

      <input type="submit" value="Logar" class="btn btn-primary"/>

    </div>

  </div>

</div>

</form>
```

fica assim:

E-mail	Senha
<button>Logar</button>	