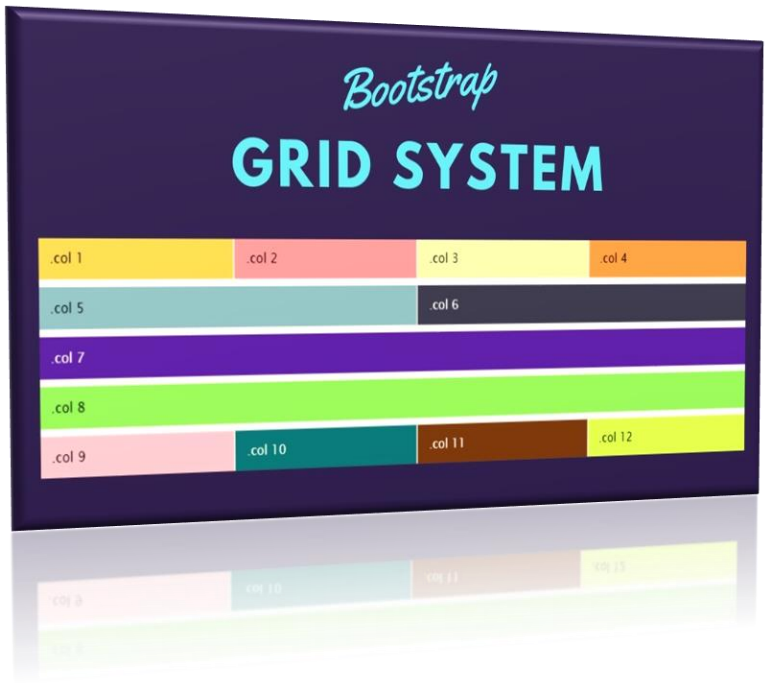
Autor: João vitor Quirino Sarti

GitHub: NULLBYTE-RGH (github.com)

Bootstrap 5: Grid system



Grid System:

O Grid nada mais é doque uma biblioteca que encapsula o tipo de (display Flex). Sendo assim o Bootstrap nada mais vai fazer do que colocar uma camada de abstração ao display, facilitando chamadas e encapsulando várias propriedades em um único comando.

Uma forma de testar e comparar ao código feito na lista 6 (display flex.), é utilizando a propriedade (container) do Bootstrap:

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

Essa propriedade nada mais é do que fazer: (em css puro)

```
.container{
display: flex;
justify-content: center;
}
```

Só que o Bootstrap traz a vantagem de várias propriedades de responsividade pré-definidas.

Para fazer com que elementos internos se comportem como (display flex) e (inline block) se faz uso do comando:

- Faz com que todo conteúdo filho seja em uma linha, respeitando o container Flex.

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

- Faz com que todo conteúdo filho seja parte de uma coluna.

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

Column	Column	Column

Código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Bootstrap 5 : Grid system</title>
    <!-- Incluindo o Bootstrap 5 via CDN-->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
    <!-- alguns recursos Bootstrap 5 precisam de codigo JS para
funcionar, como menus interativos -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundl
e.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
</head>

<body>
    <div class="container">
        <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col ">
            Column
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col ">
            Column
```

```
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col  ">
            Column
          </div>
        </div>
      </div>
</body>
</html>
```

O Bootstarp tem 2 formas de ser utilizado: via CDN, com uso dos links:

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundl
e.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
```

E podendo ser baixado a biblioteca do próprio site oficial:

https://getbootstrap.com/

Vantagem do CDN: não é necessário baixar, mas a cada vez que a página é carregada ela tem que ir até o link e obter o conteúdo

Vantagem do download: permite desenvolvimento offline e modificação das bibliotecas antes de fazer o envio ao cliente.

Tamanhos:

O Grid faz uso de 6 unidades de medidas padrões:

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

Isso quer dizer que se pode fazer todo o conteúdo ser responsivo a diferentes tamanhos de telas:

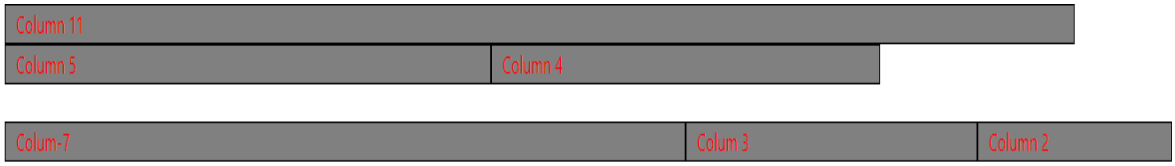
- **Xs** <576px
- **Sm** ≥576px
- **Md** ≥768px
- **Lg** ≥992px
- **Xl** ≥1200px
- **Xxl** ≥1400px

O uso dessas métricas se faz através d0 (-) após a classe:

```
col-sm col-md col-lg col-xl col-xxl
```

Junto ao tamanho mínimo de tele, poder se fazer alinhamentos com tamanhos pré-definidos de containers, no caso de (COL) pode-se variar de 1 a 12:

```
col-sm-1 col-sm-2 col-sm-5

ou apenas

col-3 col-6 col-12
```



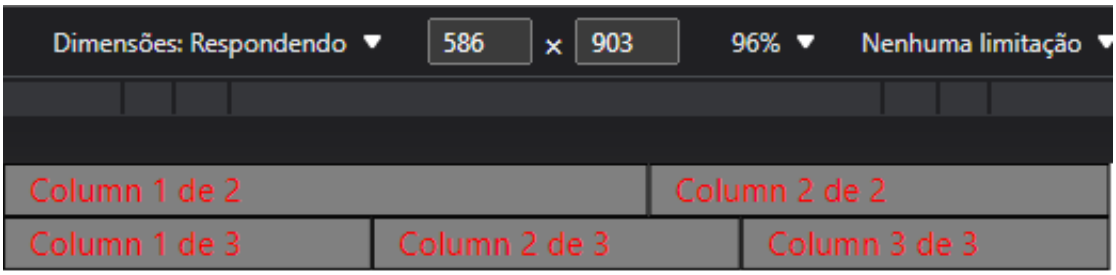

```
<body>
    <div class="container">
        <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-11">
            Column 11
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-5 ">
            Column 5
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-4  ">
            Column 4
          </div>
        </div>
      </div>
      <br>
      <div class="container">
        <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-7 ">
            Colum-7
          </div>
```

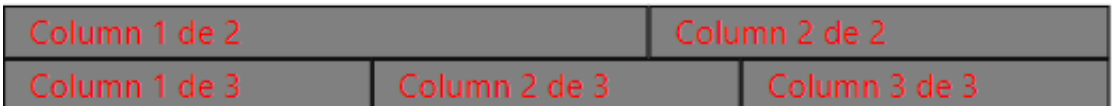
```
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-3">
            Colum 3
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-2 ">
            Column 2
          </div>
        </div>
      </div>

</body>
```

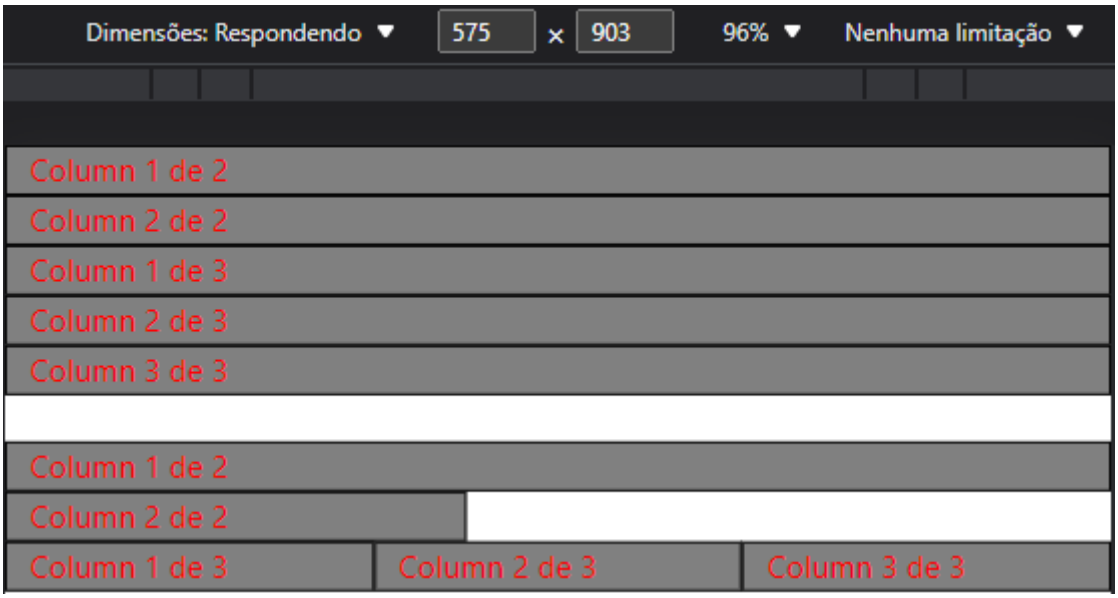
Tamanho variável e empilhamento:

Assim como no display flex, o tamanho dos componentes sempre tenta ser respeitados, e assim, mantando o maior sempre maior que os elementos menores, independentemente do tamanho da tela. Que no caso do CSS puro era aplicado com o Flex-grow ou Flex-shrink.

Com o Grid System, isso ocorre de forma automática:







Com esse código, ao se redimensionar a tela até 576 px de largura a proporção se mantem, já se for menor as TAGS que possuem o tamanho em (SM) alteram seu comportamento perdendo a proporção estimada para o limite de 576 px (SM). E teriam que ser inserida outra propriedade para o mesmo conjunto de classes dizendo a proporção para dispositivos menores que SM, que no caso seria XS. Sendo assim, pode-se definir as proporções para cada tamanho de tela, de forma fácil em apenas uma instanciação de classe.

```
col-sm-7 col-md-2
```

Isso significaria que até 768px ele terá o comprimento de 2 unidades de 12, já ao atingir 576px ele ocupara 7 unidades de 12, assim consequentemente redimensionando o componente ao seu lado.

Caso o tamanho limite seja atingido e nenhuma proporção seja definhada os elementos assumem o tamanho de 12 e cada um ocupa uma linha inteira para si. Dessa forma eles acabam se empilhando.

Código:

```
<body>
    <div class="container">
        <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-7">
            Column 1 de 2
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-5 ">
            Column 2 de 2
          </div>
        </div>
          <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-4  ">
            Column 1 de 3
```

```
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-4  ">
            Column 2 de 3
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-4  ">
            Column 3 de 3
          </div>
        </div>
      </div>

      <br>

      <div class="container">
        <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-sm-7">
            Column 1 de 2
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-5 ">
            Column 2 de 2
          </div>
        </div>
          <div class="row">
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-4  ">
            Column 1 de 3
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-4  ">
            Column 2 de 3
          </div>
          <div style="color:red; border:1px solid black;background-
color:grey;" class="col-4  ">
            Column 3 de 3
          </div>
        </div>
      </div>
</body>
```