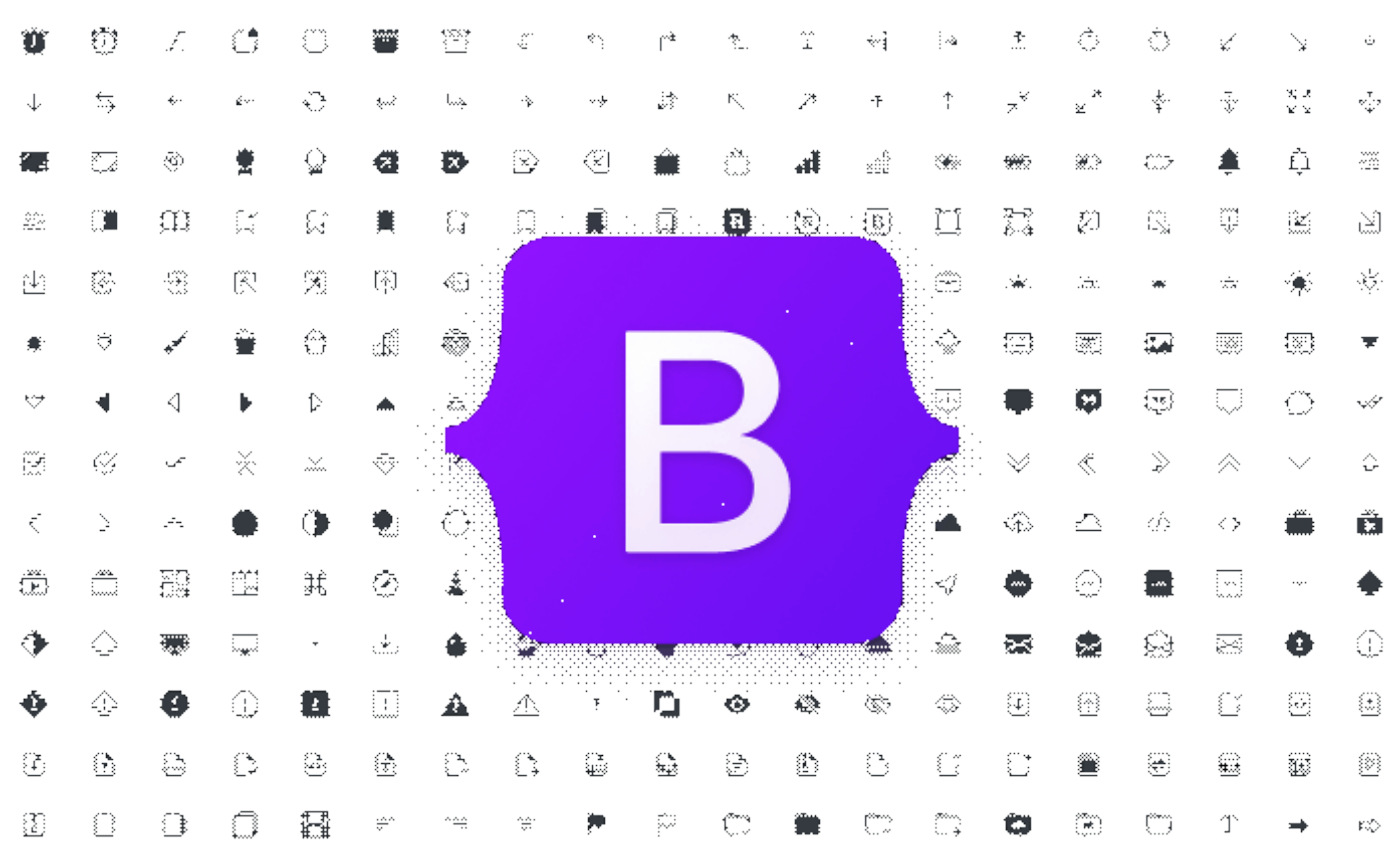


BOOTSTRAP 5 - Vinicius Ferraz

quarta-feira, 26 de abril de 2023 14:03



1.1 - Breakpoints

quarta-feira, 26 de abril de 2023 14:08

Para dominarmos design responsivo para bootstrap, temos que começar pelos breakpoints.

Breakpoints

Em Bootstrap, os breakpoints são pontos de interrupção que permitem definir a aparência de um site em diferentes tamanhos de tela, a fim de criar um design responsivo. O layout é adaptado para diferentes dispositivos, desde telas de computador desktop até smartp hones.





O Bootstrap tem cinco breakpoints que representam cinco tamanhos de tela comuns:





















- Extra small (xs) - Menor que 576 pixels
- Small (sm) - Igual ou maior que 576 pixels
- Medium (md) - Igual ou maior que 768 pixels
- Large (lg) - Igual ou maior que 992 pixels
- Extra large (xl) - Igual ou maior que 1200 pixels

Para cada breakpoint, o Bootstrap define um infixo de classe que é adicionado aos nomes de classe existentes para modificar seu comportamento. Aqui está uma tabela com os breakpoints, seus infixos de classe e dimensões:

Breakpoint	Infixo de classe	Dimensões
XS	Sem infixo	Menor que 576 pixels
SM	- sm	Igual ou maior que 576 pixels
MD	- md	Igual ou maior que 768 pixels
LG	- lg	Igual ou maior que 992 pixels
XL	- xl	Igual ou maior que 1200 pixels

Ao usar infixos de classe, você pode alterar o comportamento de uma classe Bootstrap com base no tamanho da tela, permitindo que você crie um layout responsivo e personalizado para seu site.

			
xs	sm	md	lg

	xs	sm	md	lg
				
				
				
				

Vamos analisar o conteúdo abaixo

Na classe <p>, colocamos 3 tipos de breakpoints, onde:

- Text-center irá por padrão me trazer o texto no centro da div pai.
- Text-sm-end me trará em todos os dispositivos de dimensões que enquadrem-se na classe sm (ou small) o texto alinhado mais ao fundo.
- E por fim, o text-md-center, nos dispositivos medium (md), me trará o texto alinhado ao fundo da tela.

```
<div class="container">
  <div class="row">
    <div class="col" style="border: 1px solid ■burlywood;">
      <p class="text-center text-sm-end text-md-center">Lorem ipsum dolor sit amet consectetur adipisicing
elit. Voluptates labore enim adipisci suscipit culpa illo, at eos neque, esse nam molestias
consequuntur! Nihil temporibus labore provident qui accusantium delectus ea. Ab fugit cupiditate at
fugiat nesciunt soluta iusto! Natus a, quos inventore animi ad iste delectus ipsum minima quis quasi
similique ut quae eligendi vero temporibus velit, commodi beatae eum!</p>
    </div>
  </div>
</div>
```

1.2 - Containers e Layouts em Grid

quarta-feira, 26 de abril de 2023 14:25

O container é um dos blocos de construção mais importantes para layouts responsivos usando bootstrap5.

No Bootstrap, o container é um elemento que ajuda a controlar o layout da página. Ele é usado para agrupar e centralizar o conteúdo da página dentro de um limite definido. É como se fosse uma caixa que contém o conteúdo da página, e que possui margens laterais que mantêm esse conteúdo centralizado na página.

O container é utilizado para criar um layout mais estruturado e organizado para a página, evitando que o conteúdo fique desalinhado ou bagunçado. Ele também ajuda a manter a consistência do design da página em diferentes dispositivos, desde desktops até smartphones.

Aqui está um exemplo de uma div sem o container:

```
<div class="bg-primary text-white p-4">
  <h1>Minha página</h1>
  <p>Este é o conteúdo da minha página.</p>
</div>
```

E aqui está o mesmo código, agora utilizando o container:

```
<div class="container">
  <div class="bg-primary text-white p-4">
    <h1>Minha página</h1>
    <p>Este é o conteúdo da minha página.</p>
  </div>
</div>
```

A principal diferença entre o container e o container-fluid é que o container tem uma largura fixa e predefinida em cada breakpoint, enquanto o container-fluid ocupa 100% da largura disponível da tela. O container é usado para criar um layout mais estruturado e organizado, enquanto o container-fluid é usado para criar um layout mais fluido e flexível.

Abaixo, temos uma tabela de Containers e suas classes.

Class	Extra small < 576px	Small ≥ 576px	Medium ≥ 768px	Large ≥ 992px	X-Large ≥ 1200px	XX-Large ≥ 1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

O sistema de grid do Bootstrap é construído usando as classes .row e .col-*, onde * é um valor que pode variar dependendo do tamanho da tela. A classe .row é usada para envolver um conjunto de colunas, enquanto a classe .col-* é usada para definir o tamanho das colunas em uma linha.

O sistema de grid é baseado em um grid de 12 colunas. As classes .col-* podem ser usadas para definir o número de colunas que uma determinada coluna deve ocupar em uma linha. Por exemplo, a classe .col-6 é usada para definir uma coluna que ocupa 6 das 12 colunas disponíveis (50% da largura da linha).

Um exemplo prático de como usar o sistema de grid em um container pode ser visto abaixo:

```

<div class="container">
  <div class="row">
    <div class="col-md-4">Coluna 1</div>
    <div class="col-md-4">Coluna 2</div>
    <div class="col-md-4">Coluna 3</div>
  </div>
</div>

```

Neste exemplo, o container envolve o conjunto de colunas e a classe .row é usada para definir uma nova linha.

As classes .col-md-4 são usadas para definir três colunas iguais, cada uma ocupando 4 das 12 colunas disponíveis.

É importante notar que o sistema de grid do Bootstrap é responsivo, o que significa que o número de colunas que uma coluna ocupa pode mudar dependendo do tamanho da tela. Por exemplo, uma coluna definida como .col-md-4 ocupará 4 das 12 colunas em telas médias, mas pode ocupar todas as 12 colunas em telas menores (como celulares).

Em resumo, o sistema de grid do Bootstrap usa as classes .row e .col-* para criar um sistema flexível e responsivo de colunas que podem ser usadas para construir layouts de página. O container é usado para envolver o conjunto de colunas e definir suas dimensões e alinhamento.



Agora vamos para o desafio do dia.

O Desafio do dia era criar uma page com uma única row somente.

E Foi isso mesmo, o resultado foi atingido.

Dentro de um container coloquei uma row, e dentro dessa row coloquei 4 col.

Para visualiza-las melhormente, deixei uma cor específica para cada uma, e também defini algumas alturas.

Obviamente que dava pra fazer de outras maneiras, mas é insano a idéia de que você pode criar um site somente com uma row.

De aprendizado, fica que, por padrão, todas as col, virão definidas com o valor de 12 colunas.

Se eu não especificar, ela fica com o valor que sobrar, e caso eu coloque uma outra col dentro da row, neste caso a col que está sem valor, ficará com o valor que sobrar.

Mas como cada uma ficou especificado sendo uma com 12, outra com 3, outra com 9 e a última com 12, foi possível concluir o desafio.

Lembrando que também era possível criar 1 row para cada parte que eu quisesse.

1 row pro cabeçalho (header)

1 row para o meio (nav e main)

1 row para o rodapé (footer)

Resultado Esperado:	Código utilizado:
---------------------	-------------------



```

<style>
  .h400 {
    height: 400px;
  }

  .h100 {
    height: 100px;
  }
</style>

<body>

  <!--Abaixo virá o Header-->

  <div class="container">
    <div class="row">
      <div class="col-12 bg-dark bg-opacity-75 h100">header</div>
      <div class="col-3 bg-dark bg-opacity-20 h400">nav</div>
      <div class="col-9 bg-dark bg-opacity-10 h400">main</div>
      <div class="col-12 bg-dark bg-opacity-75 h100">footer</div>
    </div>
  </div>

```



As informações em vermelho e em verde limão, guarde isso sempre na sua cabeça!

```

<!--Abaixo virá o Header-->

<div class="container">
  <div class="row">
    <div class="col-12 bg-dark bg-opacity-75 h100">header</div>
    <div class="col-12 col-md-3 bg-dark bg-opacity-20 h400">nav</div>
    <div class="col-12 col-md-9 bg-dark bg-opacity-10 h400">main</div>
    <div class="col-12 bg-dark bg-opacity-75 h100">footer</div>
  </div>
</div>

<!--Abaixo estarão os scripts do bootstrap-->

```

Este "col-12" será aplicado do xs (que é o menor tamanho, até o último tamanho Até que eu defina um outro breakpoint como por exemplo col-md-3

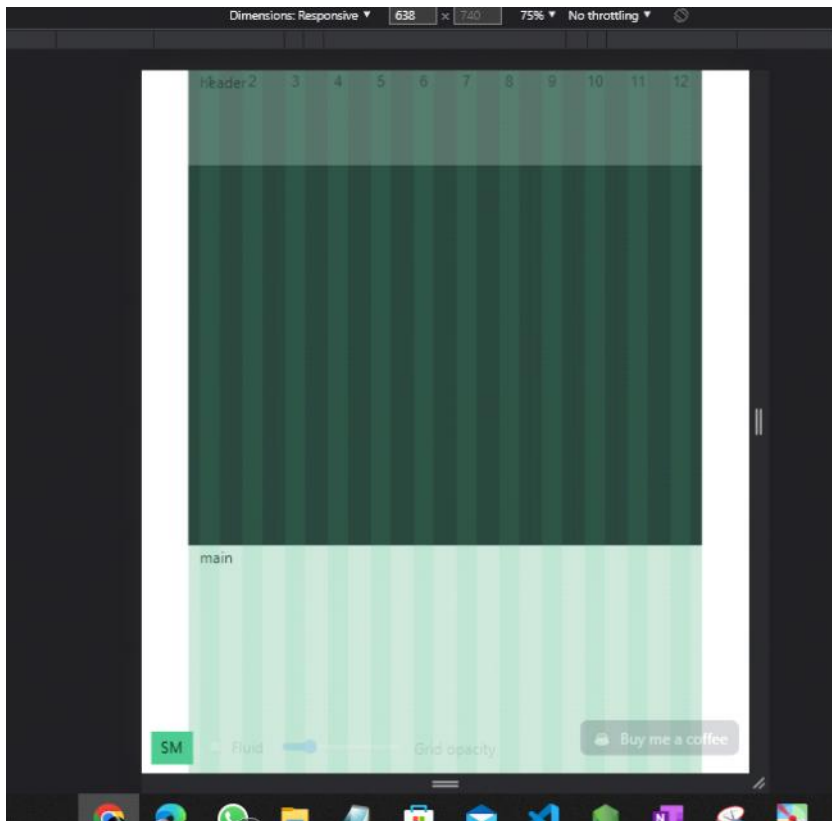
A partir do col-md todos os outros dispositivos maiores, terão essas medidas

Olha como ficou:

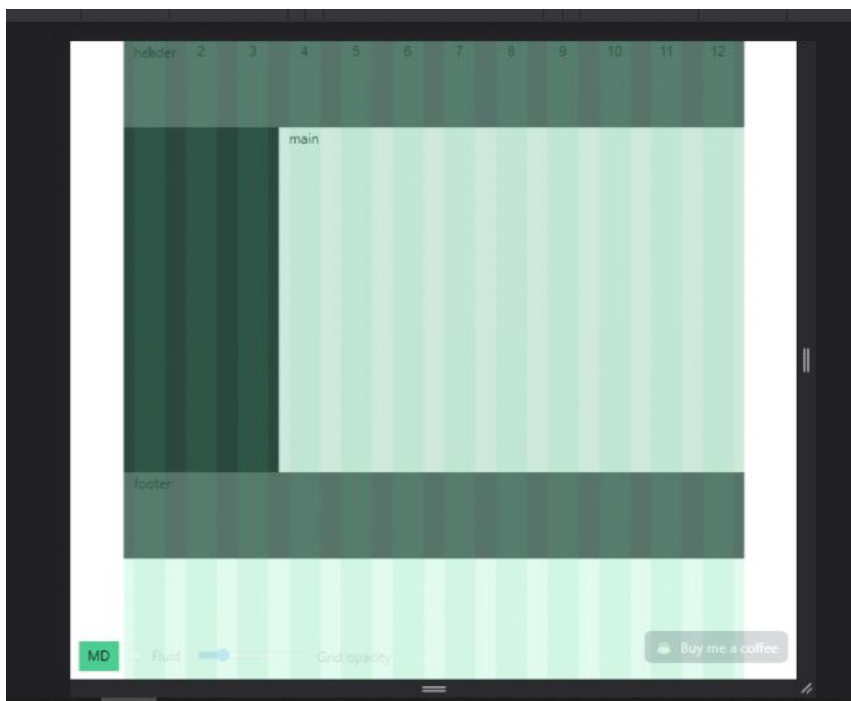
Estou utilizando uma extensão do chrome para verificar as divisões.

Veja que no col-12, eu havia deixado para o menor tamanho em diante **até que chegasse no meu breakpoint col-md**, os quais teriam outros tamanhos definidos.

Exemplo de layout **col-12** em um dispositivo **SM**



Agora o exemplo a partir do tamanho do breakpoint col-md. Resultado do Layout abaixo:

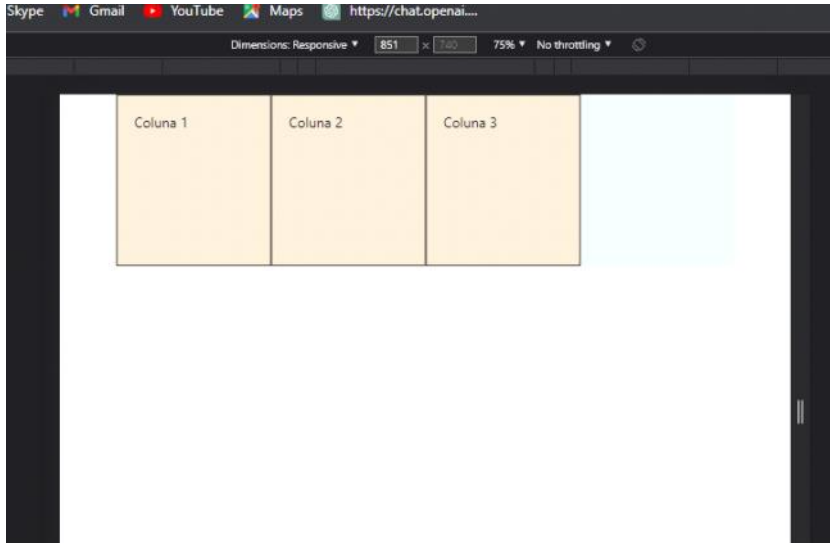


1.3 - Formatação de Colunas de Layout

quarta-feira, 26 de abril de 2023

16:08

Vamos observar o layout para estudarmos:



Temos uma `.row` com o fundo clarinho lá atrás, ocupando 12 colunas. Dentro dessa `.row` temos 3 `.col` as 3 com o mesmo tamanho.

Esse é o código.

```
<style>
  .row {height: 200px;
  background-color: #aliceblue;
}

.col-3 {background-color: #antiquewhite;
padding: 20px;
border: 1px solid black;}

</style>

<div class="container">
  <div class="row">
    <div class="col-3">Coluna 1</div>
    <div class="col-3">Coluna 2</div>
    <div class="col-3">Coluna 3</div>
  </div>
</div>
```

```
<div class="container">
  <div class="row">
    <div class="col-3">Coluna 1</div>
    <div class="col-3">Coluna 2</div>
    <div class="col-3">Coluna 3</div>
  </div>
</div>
```

Por padrão, essa "row" estará comportando todas as divs que vierem abaixo dela no caso abaixo, estamos usando col-3 para as 3, e a divisão pai, que no caso é .row terá 12 colunas

Align-Items

Vejamos o código abaixo:


```

<div class="container">
  <div class="row align-items-start">
    <div class="col-3">Coluna 1</div>
    <div class="col-3">Coluna 2</div>
    <div class="col-3">Coluna 3</div>
  </div>
</div>

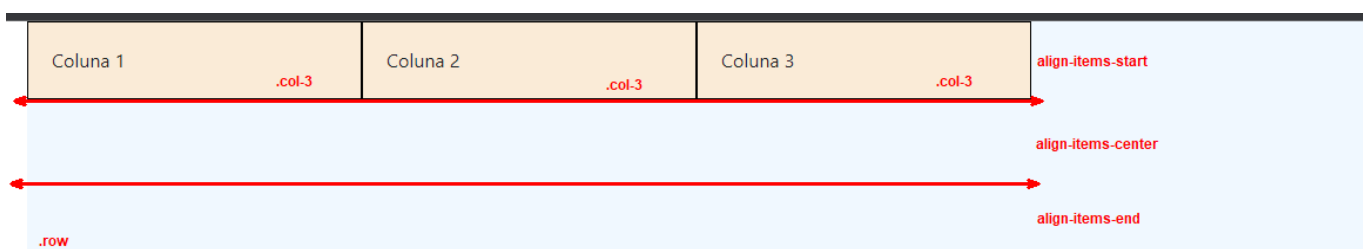
```

Alteramos as propriedades de row, com align-items-start, que é modificar sua posição do eixo vertical (y). Temos também outras posições.

Vale ressaltar que esse alinhamento vertical só se aplica quando o container tem uma altura superior a altura das colunas.

- **align-items-start:** alinha os itens no início do container flexível;
- **align-items-end:** alinha os itens no final do container flexível;
- **align-items-center:** centraliza os itens verticalmente dentro do container flexível;
- **align-items-baseline:** alinha os itens de acordo com a linha de base de seu conteúdo;
- **align-items-stretch:** estica os itens para preencher o container flexível (altura total).

Tivemos o seguinte resultado e de quebra já deixamos outros exemplos.



Align-Self-Start / Center / End

Vejam os abaixo:

```

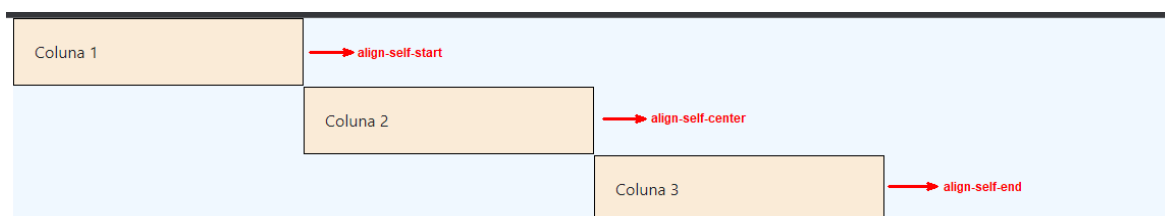
<div class="container">
  <div class="row">
    <div class="col-3 align-self-start">Coluna 1</div>
    <div class="col-3 align-self-center">Coluna 2</div>
    <div class="col-3 align-self-end">Coluna 3</div>
  </div>
</div>

```

Removemos o parametro align-items

Para mexer somente na coluna, utilizaremos o align-self

Resultado:



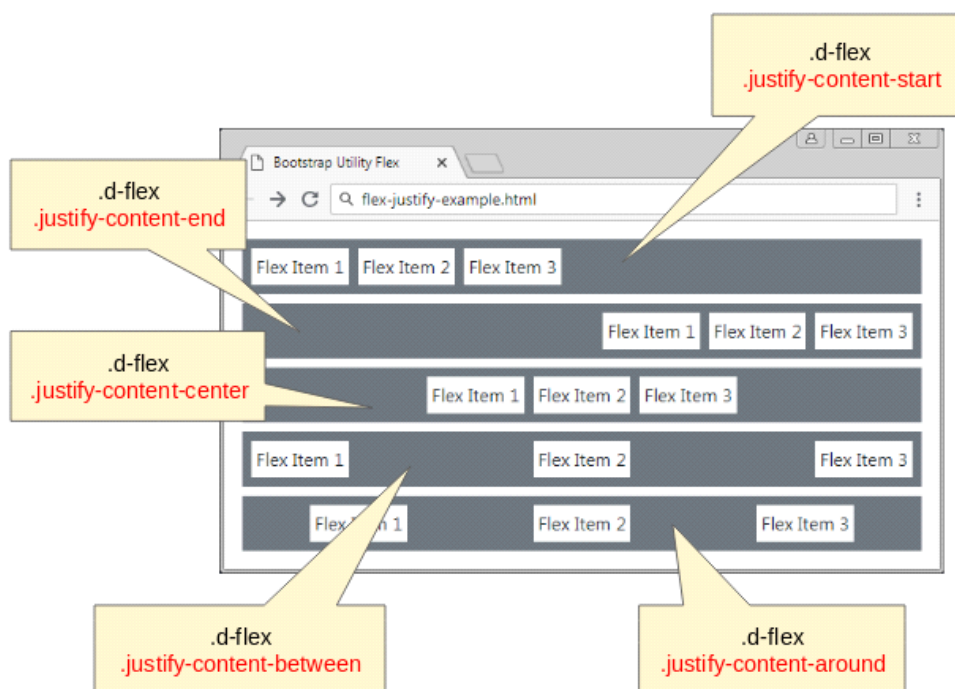
Uma observação importante a se fazer é que **todas essas classes tem suas variantes com os indicadores de breakpoints**, ou seja, é possível que tenhamos um alinhamento vertical diferente para uma coluna como por exemplo, em uma mostrada em tela de celular e uma coluna mostrada no desktop.

Justify-content-start/center/end

Existe uma série de parametros para justify-content.

- **justify-content-start:** alinha os itens no início do eixo principal do container flexível;
- **justify-content-end:** alinha os itens no final do eixo principal do container flexível;
- **justify-content-center:** centraliza os itens ao longo do eixo principal do container flexível;
- **justify-content-between:** distribui os itens ao longo do eixo principal do container flexível com espaçamento igual entre eles, mas não entre o primeiro e o último item;
- **justify-content-around:** distribui os itens ao longo do eixo principal do container flexível com espaçamento igual em torno deles;
- **justify-content-evenly:** distribui os itens ao longo do eixo principal do container flexível com espaçamento igual entre eles, inclusive entre o primeiro e o último item.

Vejamos abaixo na prática



Order-1/2/3...

Sobre podemos dizer que serve para mudar a ordem da disposição da coluna.
De maneira rápida, irei mostrar:

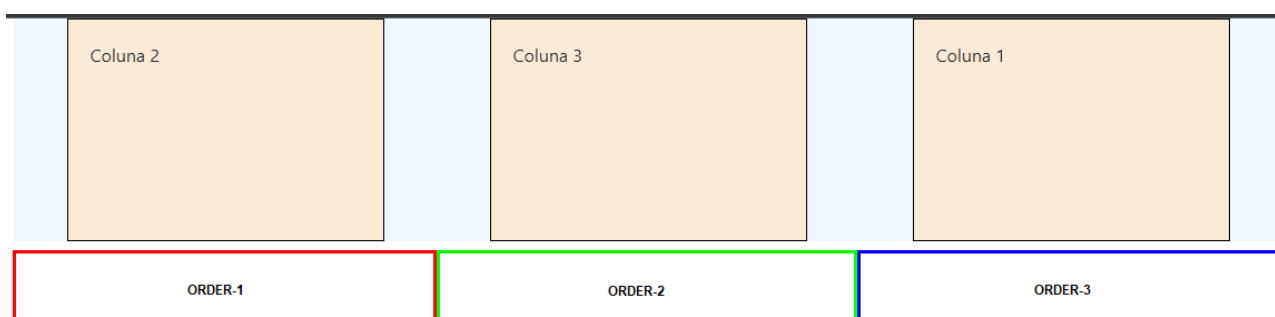
```
<div class="container">
  <div class="row justify-content-around">
    <div class="col-3 order-3">Coluna 1</div>
    <div class="col-3 order-1">Coluna 2</div>
    <div class="col-3 order-2">Coluna 3</div>
  </div>
</div>
```

Olhando aqui em cima, você pode ver que a coluna 2 está para aparecer na order-1.

A coluna 1, irá aparecer na order-3

A coluna 3, irá aparecer na order-2

Resultado abaixo.



offset-1/2/3...

Para deslocar uma div para o lado, podemos utilizar o parametro offset, o qual precede por um número que representa a quantidade de colunas.

Exemplo

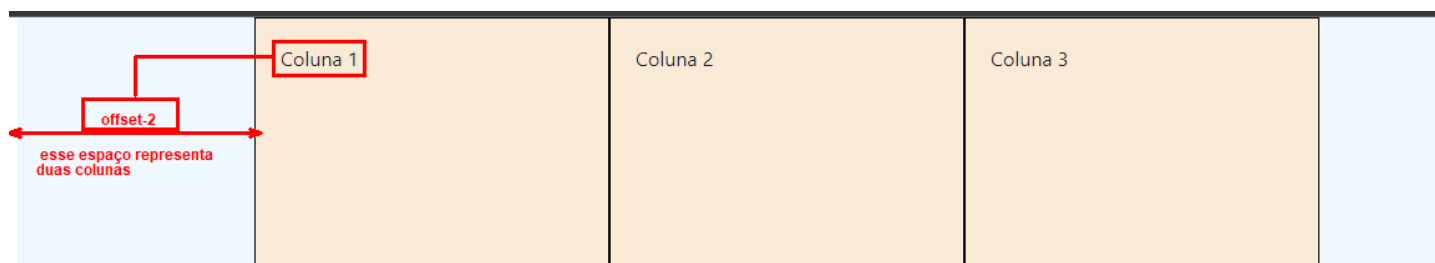
Offset-3 terá um deslocamento de 3 colunas.

Offset-2 terá um deslocamento de 2 colunas.

Código:

```
<div class="container">
  <div class="row">
    <div class="col-3 offset-2">Coluna 1</div>
    <div class="col-3">Coluna 2</div>
    <div class="col-3">Coluna 3</div>
  </div>
</div>
```

Resultado:



O bootstrap5 possui classes para colocação de margens automáticas.

Para tal, eu posso considerar em uma div

MS-AUTO (MS significa Margin Start)

Podemos também utilizar MX.

Mx-auto (mx significa eixo X, ou seja, eixo horizontal, esse recurso é bom para centralizar divs sem utilizar o offset)

As classes de **margin** começam com **m-** e são seguidas de um número de 1 a 5, representando diferentes tamanhos de margem.

As classes mx-auto e ms-auto são exemplos de classes de alinhamento disponíveis no Bootstrap. mx-auto centraliza horizontalmente um elemento dentro de seu container pai, enquanto ms-auto alinha o elemento à direita dentro de seu container pai.

Segue abaixo uma tabela:

Classe	Descrição
<code>`.m-1`</code>	Aplica uma margem de <code>`0.25rem`</code> em todos os lados do elemento.
<code>`.m-2`</code>	Aplica uma margem de <code>`0.5rem`</code> em todos os lados do elemento.
<code>`.m-3`</code>	Aplica uma margem de <code>`1rem`</code> em todos os lados do elemento.
<code>`.m-4`</code>	Aplica uma margem de <code>`1.5rem`</code> em todos os lados do elemento.
<code>`.m-5`</code>	Aplica uma margem de <code>`3rem`</code> em todos os lados do elemento.
<code>`.mx-auto`</code>	Centraliza horizontalmente o elemento dentro de seu container pai.
<code>`.ms-auto`</code>	Alinha o elemento à direita dentro de seu container pai.