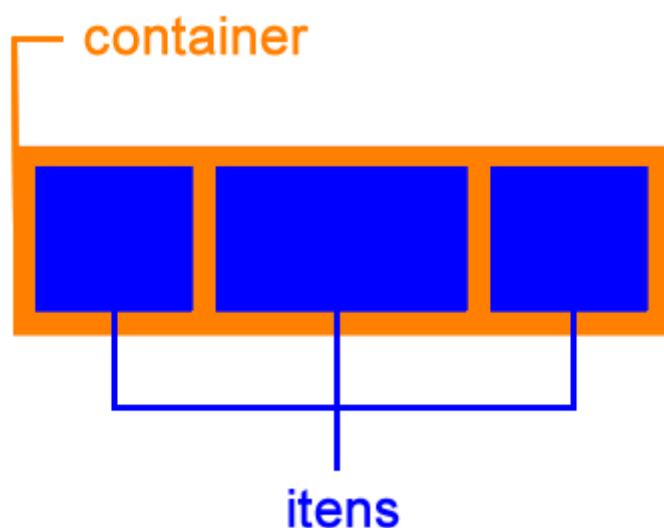


Introdução ao FLEXBOX

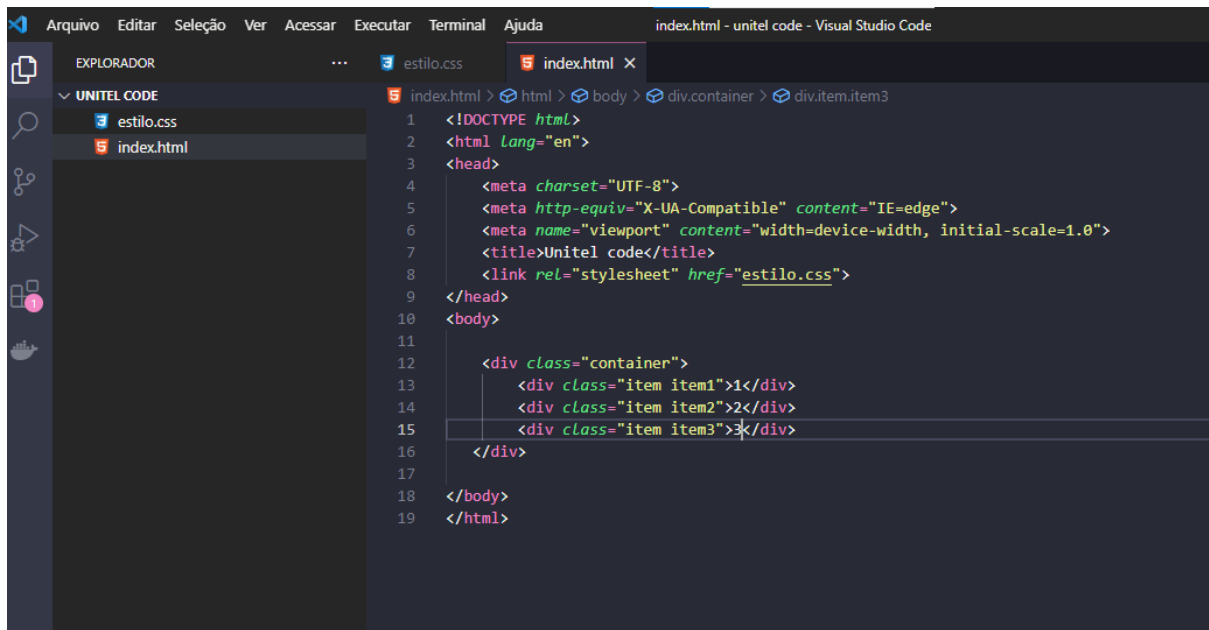
O **Flexbox** é um conceito do **CSS3** que visa organizar os elementos de uma página HTML dentro de seus containers de forma dinâmica. Ou seja, independente das suas dimensões eles sempre manterão um layout flexível dentro do seu elemento pai, reorganizando-se de acordo com a necessidade.

Conceitos

O **Flexbox** é um conjunto de propriedades que tem por objetivo organizar itens dentro de um elemento pai, normalmente chamado de container, conforme ilustra a **Figura 1**.



Portanto, para utilizar esse recurso é necessário ter no HTML ao menos um elemento (container) contendo outros (itens), como no código abaixo:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Unitel code</title>
8   <link rel="stylesheet" href="estilo.css">
9 </head>
10 <body>
11
12   <div class="container">
13     <div class="item item1">1</div>
14     <div class="item item2">2</div>
15     <div class="item item3">3</div>
16   </div>
17
18 </body>
19 </html>
```

Conforme veremos a seguir, algumas propriedades serão aplicadas ao container, enquanto outras serão aplicadas aos itens.

Display

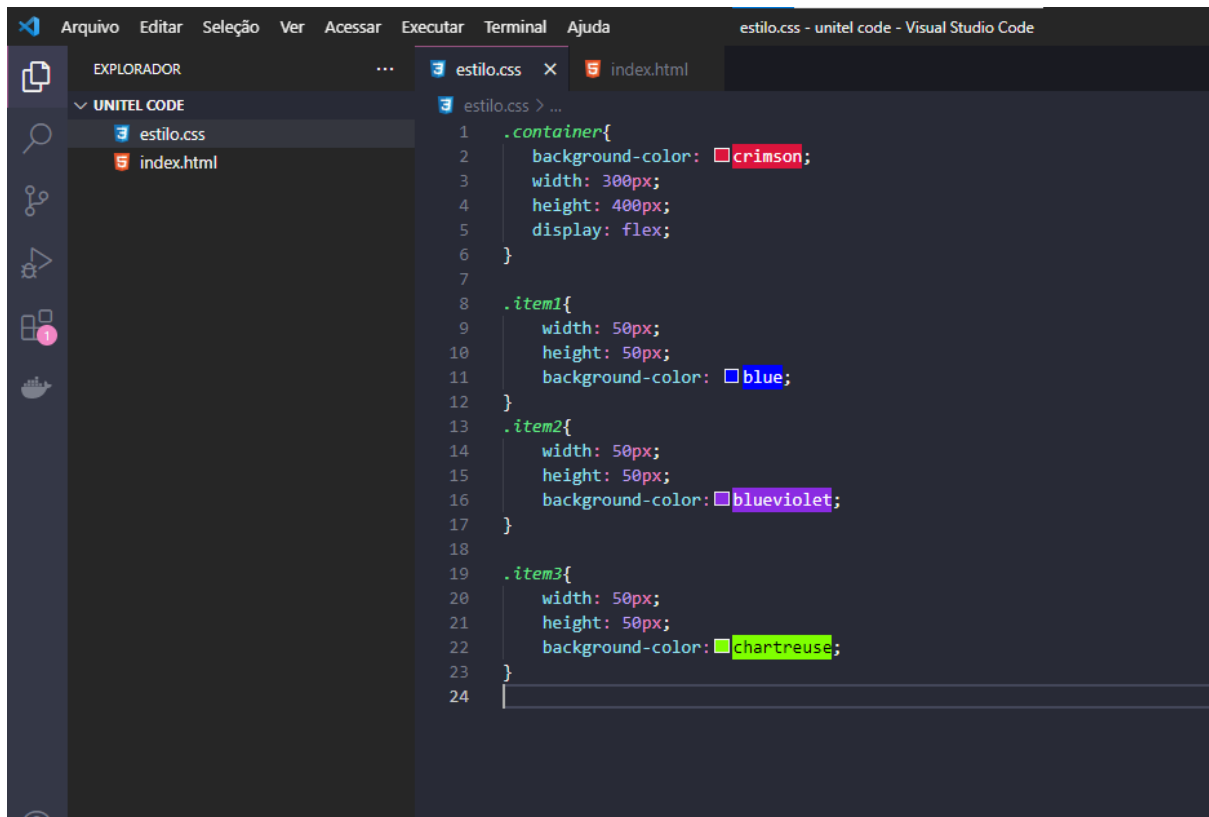
O primeiro passo para **utilizar o Flexbox** é definir a propriedade **display** do container com o valor **flex**. Isso é necessário para que as demais propriedades apresentem o resultado esperado.

A sintaxe de uso dessa propriedade é a seguinte:



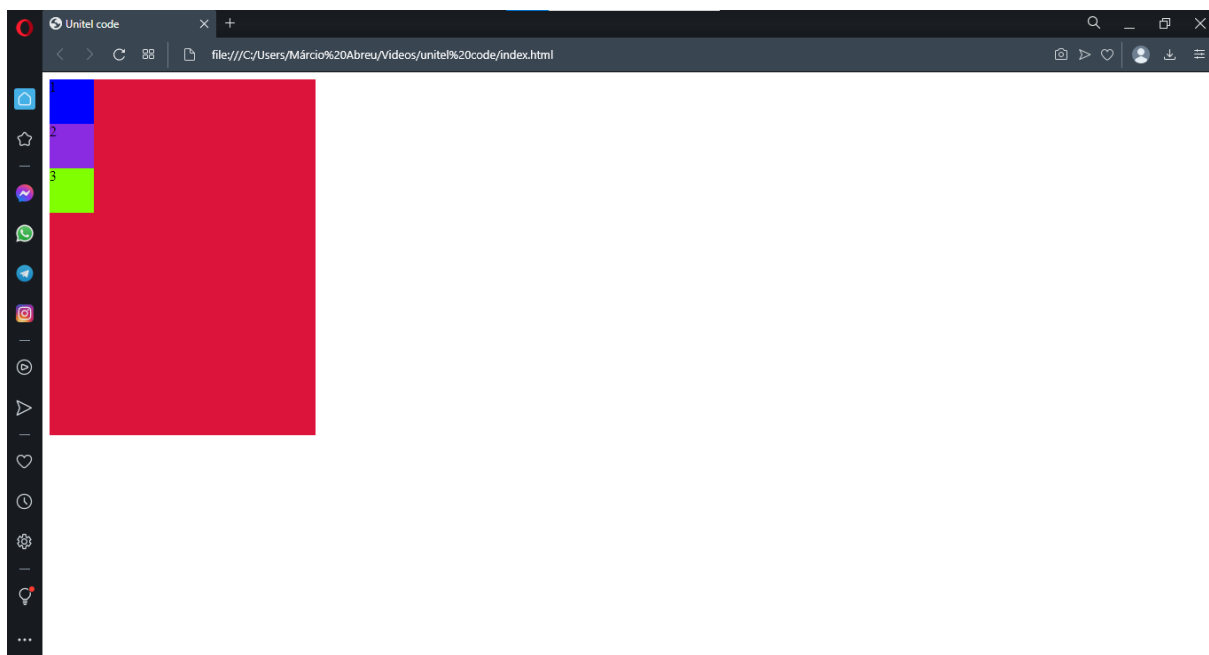
```
1
2 .container{
3   display: flex;
4 }
```

Para nós ver bem como está funcionando as propriedades do flexbox, vamos mexer nas classes filhas do container do html, que são os itens.



```
1 .container{
2   background-color: crimson;
3   width: 300px;
4   height: 400px;
5   display: flex;
6 }
7
8 .item1{
9   width: 50px;
10  height: 50px;
11  background-color: blue;
12 }
13 .item2{
14   width: 50px;
15   height: 50px;
16   background-color: blueviolet;
17 }
18
19 .item3{
20   width: 50px;
21   height: 50px;
22   background-color: chartreuse;
23 }
24
```

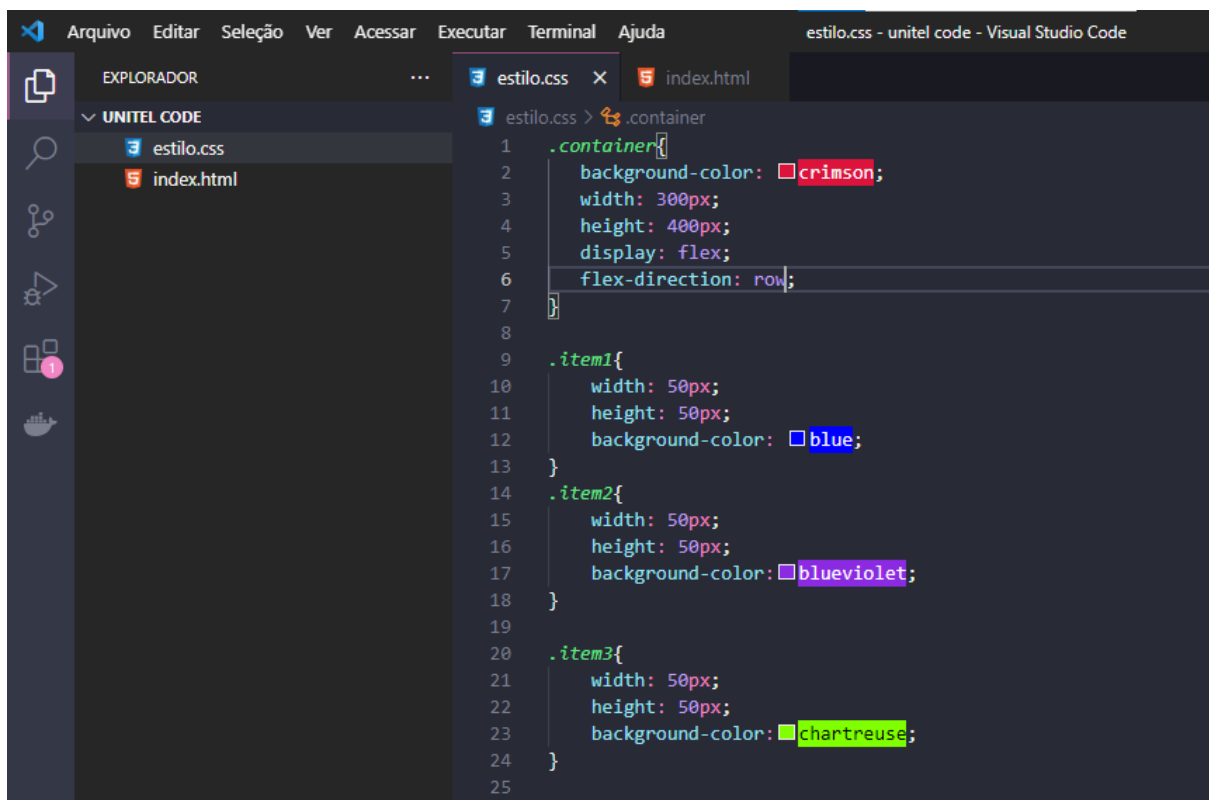
Nesse momento já demos um tamanho no elemento pai que é o container, colocamos fundo, e utilizamos a propriedade fundamental que é `display: flex`; de modo que o flexbox funciona.



Esse é o resultado do código que acabamos de fazer em css.

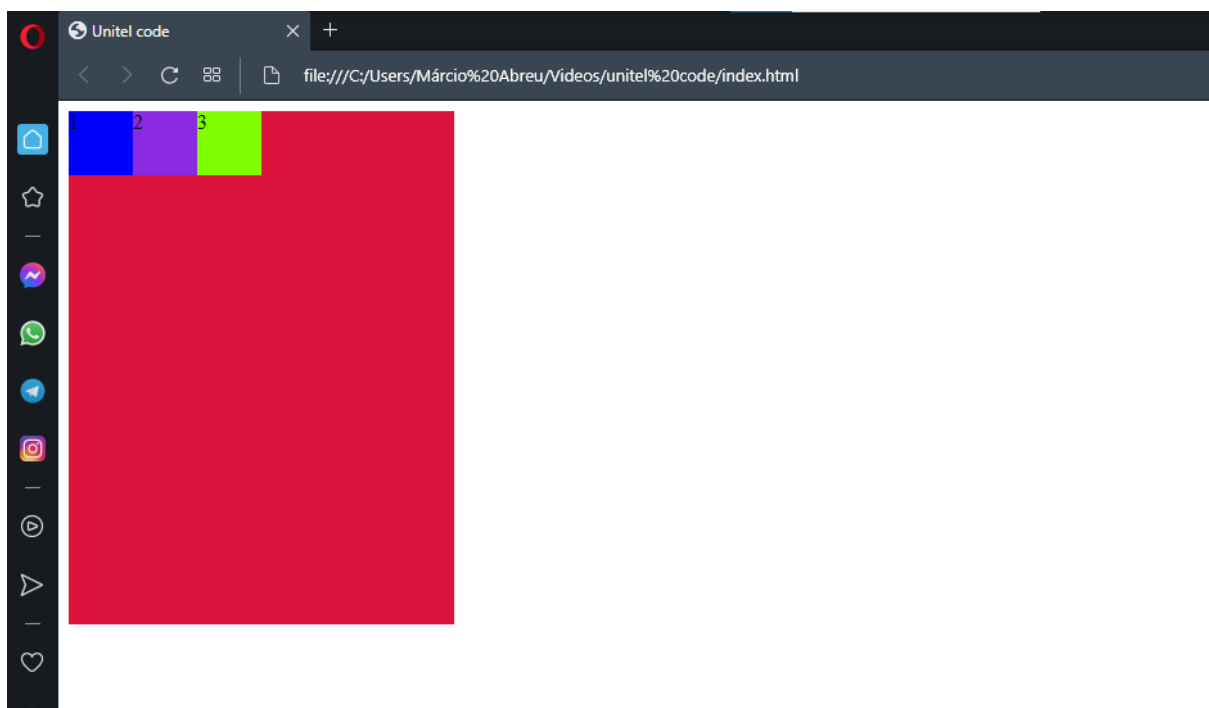
Flex-direction: A propriedade `flex-direction` deve ser aplicada ao container e define o eixo/fluxo de exibição em que os elementos serão

organizados. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:



```
estilo.css > .container
1  .container{
2      background-color: crimson;
3      width: 300px;
4      height: 400px;
5      display: flex;
6      flex-direction: row;
7  }
8
9  .item1{
10     width: 50px;
11     height: 50px;
12     background-color: blue;
13 }
14 .item2{
15     width: 50px;
16     height: 50px;
17     background-color: blueviolet;
18 }
19
20 .item3{
21     width: 50px;
22     height: 50px;
23     background-color: chartreuse;
24 }
25
```

usando o **flex-direction: row**; ele vai colocar todos os elementos filhos em forma de linha. vamos ver como está no navegador.



- **row** (padrão): Os itens são organizados em forma de linha da esquerda para a direita;
- **row-reverse**: Os itens são organizados em forma de linha da direita para a esquerda;
- **column**: Os itens são organizados em forma de coluna iniciando de cima para baixo;
- **column-reverse**: Os itens são organizados em forma de coluna iniciando de baixo para cima.

A **Figura 3** ilustra o funcionamento de cada valor.

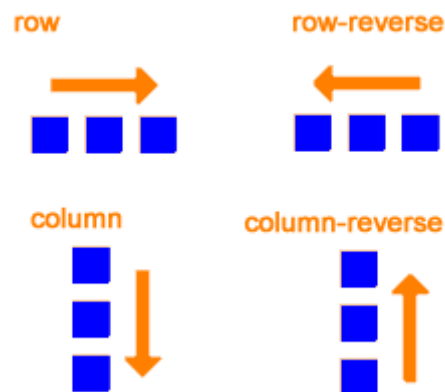
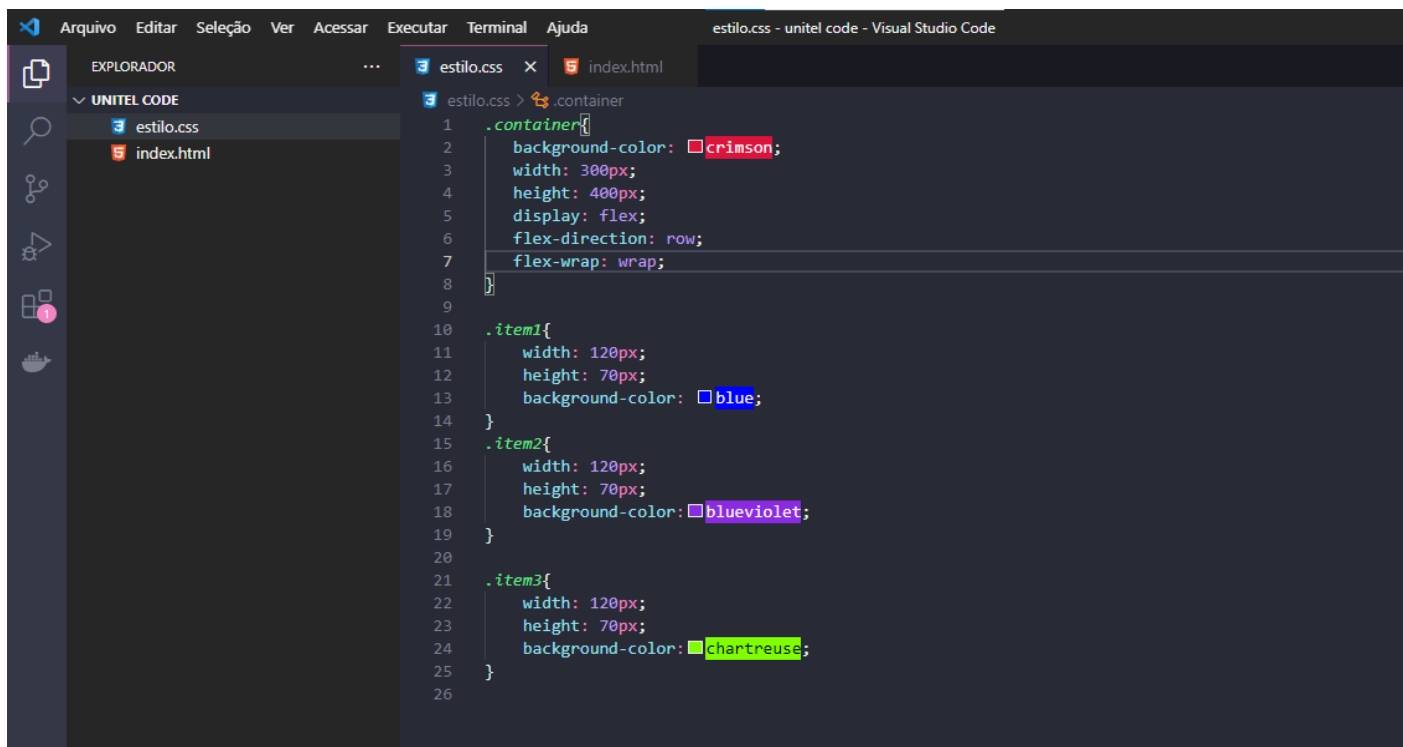


Figura 3. Funcionamento da propriedade flex-direction

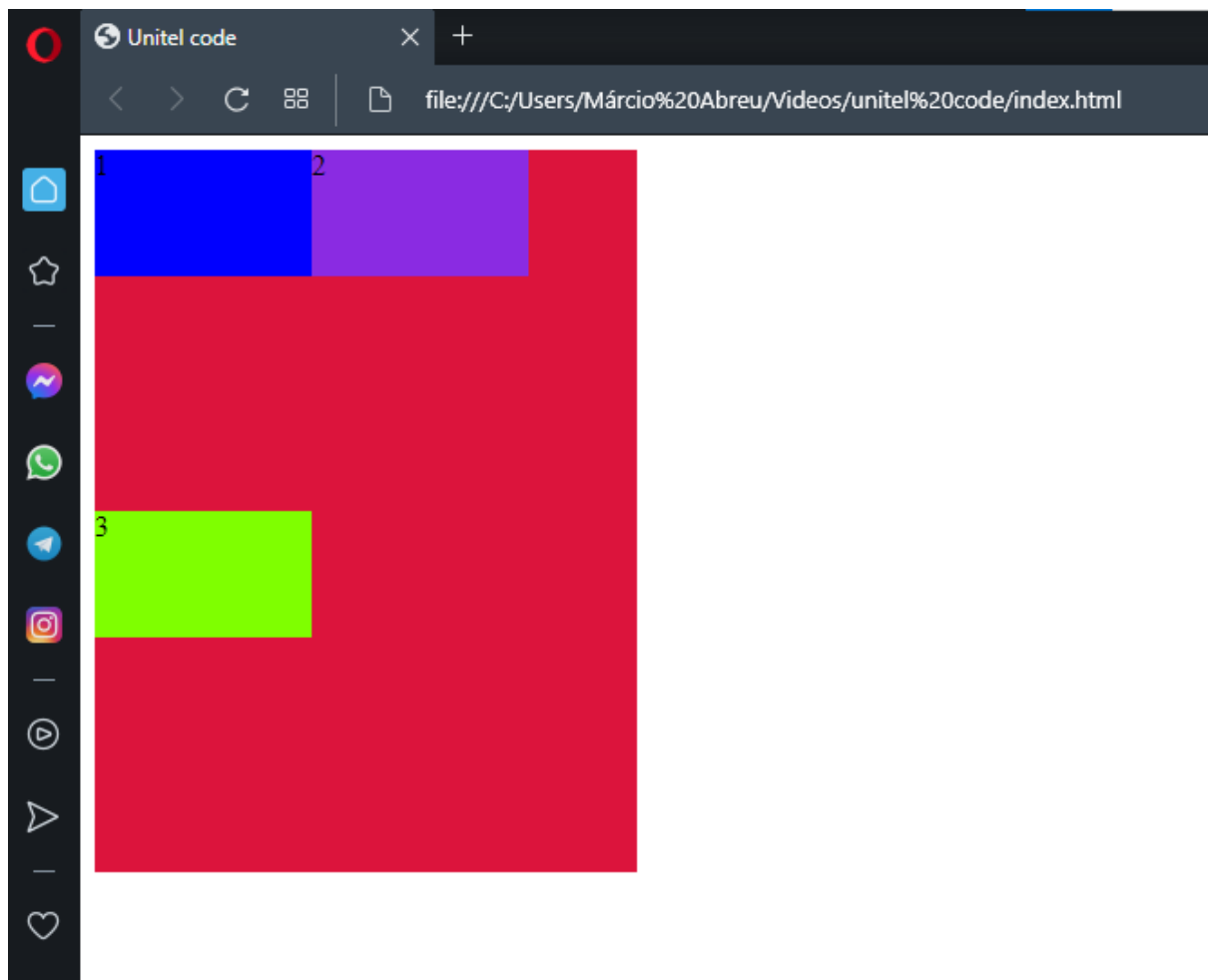
Flex-wrap: Por padrão os itens do container tentarão se ajustar em uma única linha dentro do container, mas para isso a sua largura original pode ser ajustada para que todos caibam na largura do elemento pai. Com a propriedade **flex-wrap** aplicada ao container podemos alterar esse comportamento, fazendo com que ocorra a “quebra de linha” nos itens.

A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

A screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar reads 'estilo.css - unitel code - Visual Studio Code'. On the left, the 'EXPLORADOR' sidebar shows a project named 'UNITEL CODE' with two files: 'estilo.css' and 'index.html'. The main editor area displays the 'estilo.css' file with the following CSS code:

```
1 .container{
2   background-color: crimson;
3   width: 300px;
4   height: 400px;
5   display: flex;
6   flex-direction: row;
7   flex-wrap: wrap;
8 }
9
10 .item1{
11   width: 120px;
12   height: 70px;
13   background-color: blue;
14 }
15 .item2{
16   width: 120px;
17   height: 70px;
18   background-color: blueviolet;
19 }
20
21 .item3{
22   width: 120px;
23   height: 70px;
24   background-color: chartreuse;
25 }
26
```

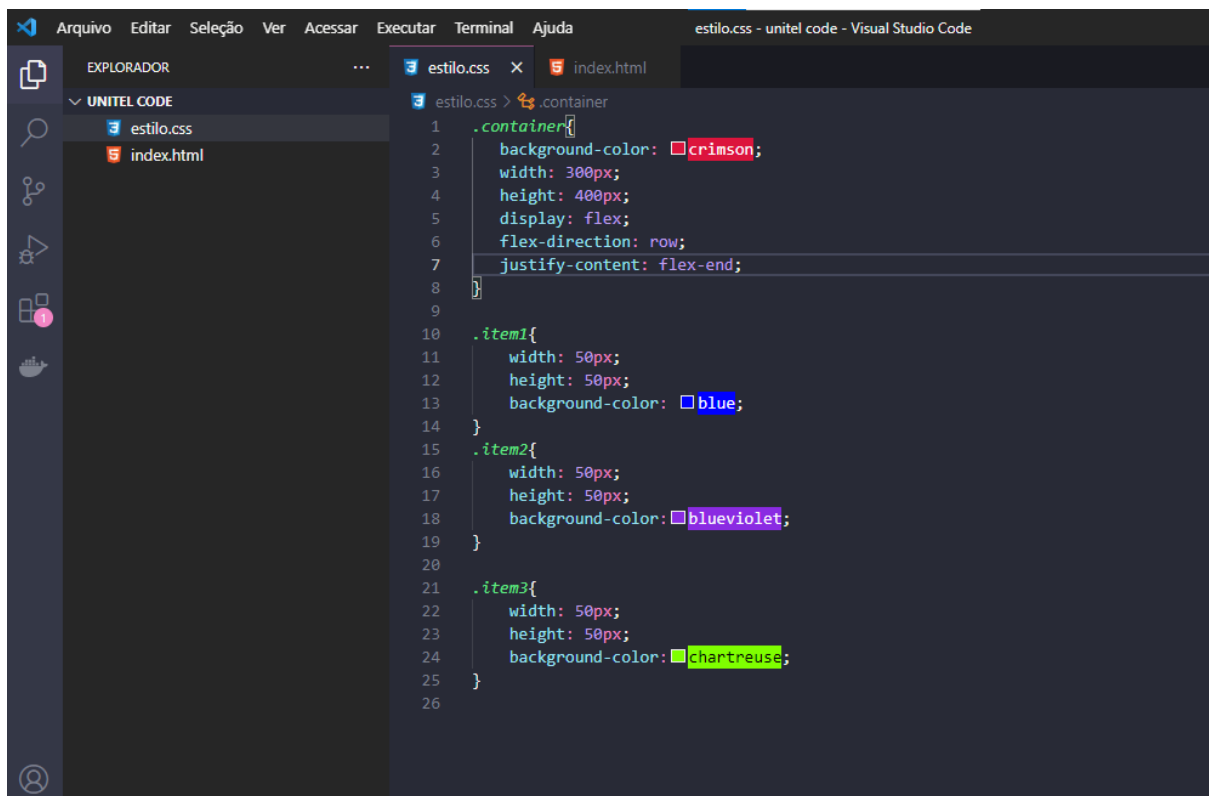
Flex-wrap só funciona quando os elementos filhos ficam com mais tamanho que o elemento pai. Quando isso acontece deve-se usar a propriedade **flex-wrap: wrap;** de modo a fazer quebra de linha. Porque ele passou do tamanho do seu pai e isso não pode acontecer.



Nesse momento houve uma quebra de linha. O item número 3 não pode ficar ai em cima, devido ao tamanho que ele tem.

- **nowrap** (padrão): Todos os itens serão dispostos em uma linha;
- **wrap**: Ocorrerá a quebra de linha e os itens mais à direita serão deslocados para a linha de baixo;
- **wrap-reverse**: Ocorrerá a quebra de linha e os itens mais à direita serão deslocados para a linha de cima;

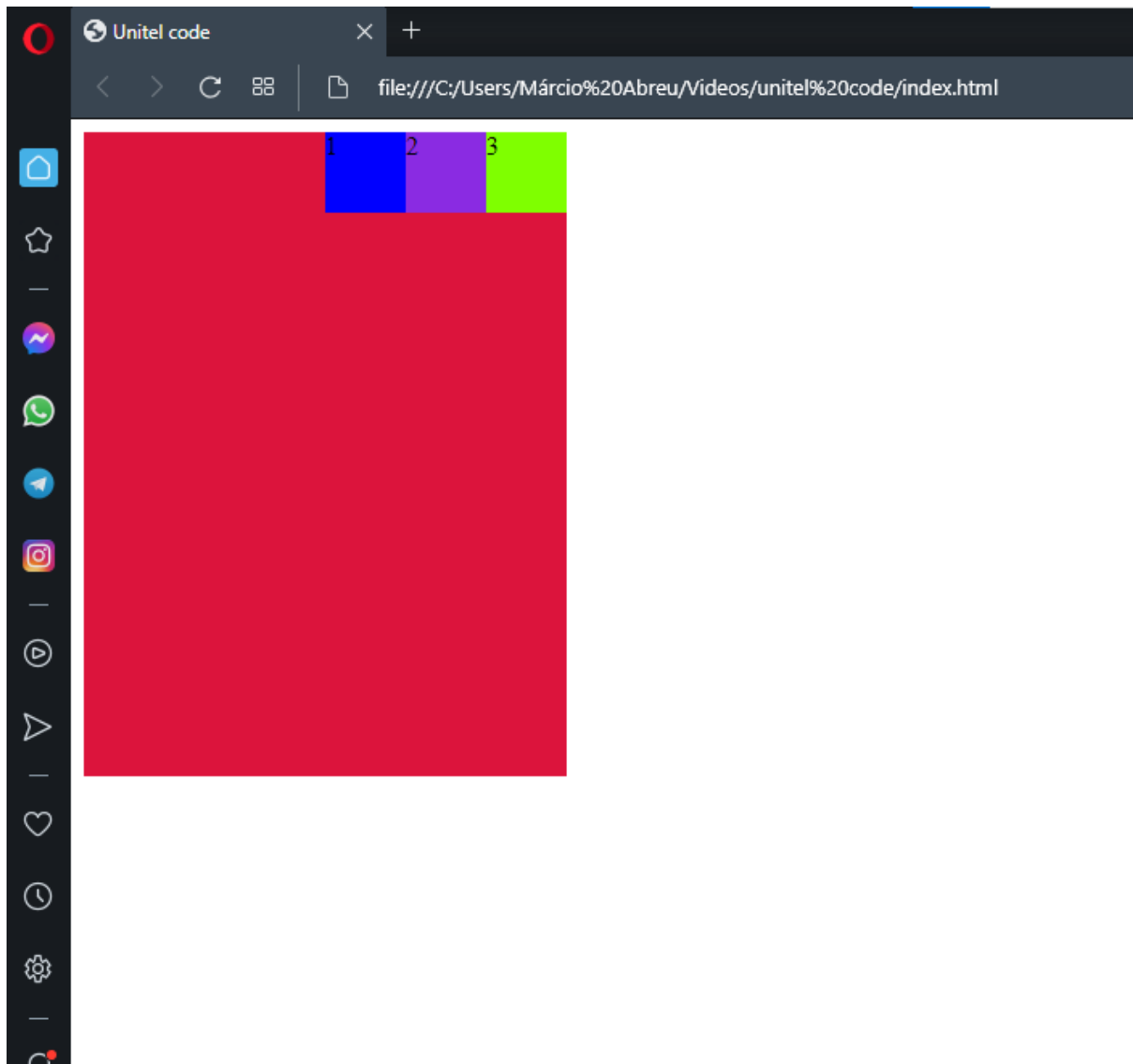
Justify-content: A propriedade **justify-content** define o alinhamento dos itens ao longo do eixo principal do container. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:



The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar indicates the active file is 'estilo.css - unitel code - Visual Studio Code'. On the left, the 'EXPLORADOR' sidebar shows a project named 'UNITEL CODE' containing two files: 'estilo.css' and 'index.html'. The main editor area displays the content of 'estilo.css'. The code defines a flex container and three items. The container has a crimson background, 300px width, 400px height, flex display, row direction, and justify-content set to flex-end. The three items have widths and heights of 50px each, with backgrounds of blue, blueviolet, and chartreuse respectively.

```
1  .container{
2    background-color: crimson;
3    width: 300px;
4    height: 400px;
5    display: flex;
6    flex-direction: row;
7    justify-content: flex-end;
8  }
9
10 .item1{
11   width: 50px;
12   height: 50px;
13   background-color: blue;
14 }
15 .item2{
16   width: 50px;
17   height: 50px;
18   background-color: blueviolet;
19 }
20
21 .item3{
22   width: 50px;
23   height: 50px;
24   background-color: chartreuse;
25 }
26
```

Aplicando essa propriedade **justify-content: flex-end;** ele coloca todos os elementos filhos à direita, temos que eliminar a propriedade quebra de linha. Agora vamos ver o resultado no navegador.



- **flex-start (padrão):** Os itens são alinhados a partir do início do eixo principal;
- **flex-end:** Os itens são alinhados a partir do fim do eixo principal;
- **center:** Os itens são alinhados ao centro do eixo principal;
- **space-between:** O primeiro item é deslocado para o início do eixo principal, o último é deslocado para o final do eixo principal e os demais são distribuídos uniformemente entre eles;
- **space-around:** Os itens são uniformemente distribuídos ao longo do eixo principal. Aqui, porém, são atribuídas margens iguais à esquerda e à direita (ou acima e abaixo, dependendo da direção do eixo principal). Por isso o primeiro e o último item não ficam “colados” nas bordas do container.

A **Figura 4** ilustra o funcionamento de cada valor:

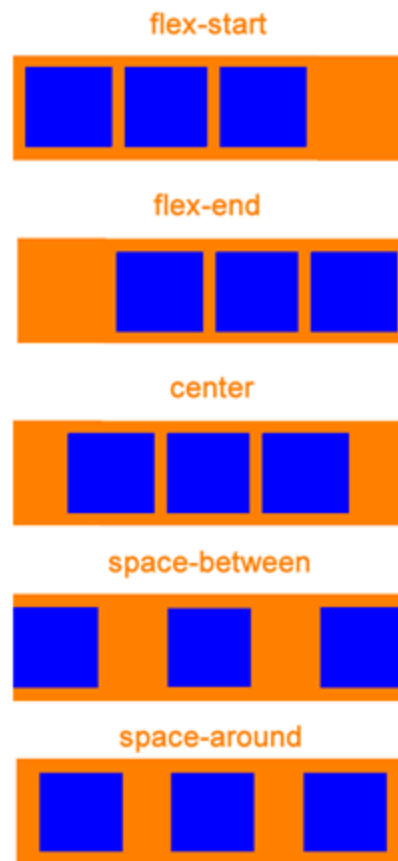
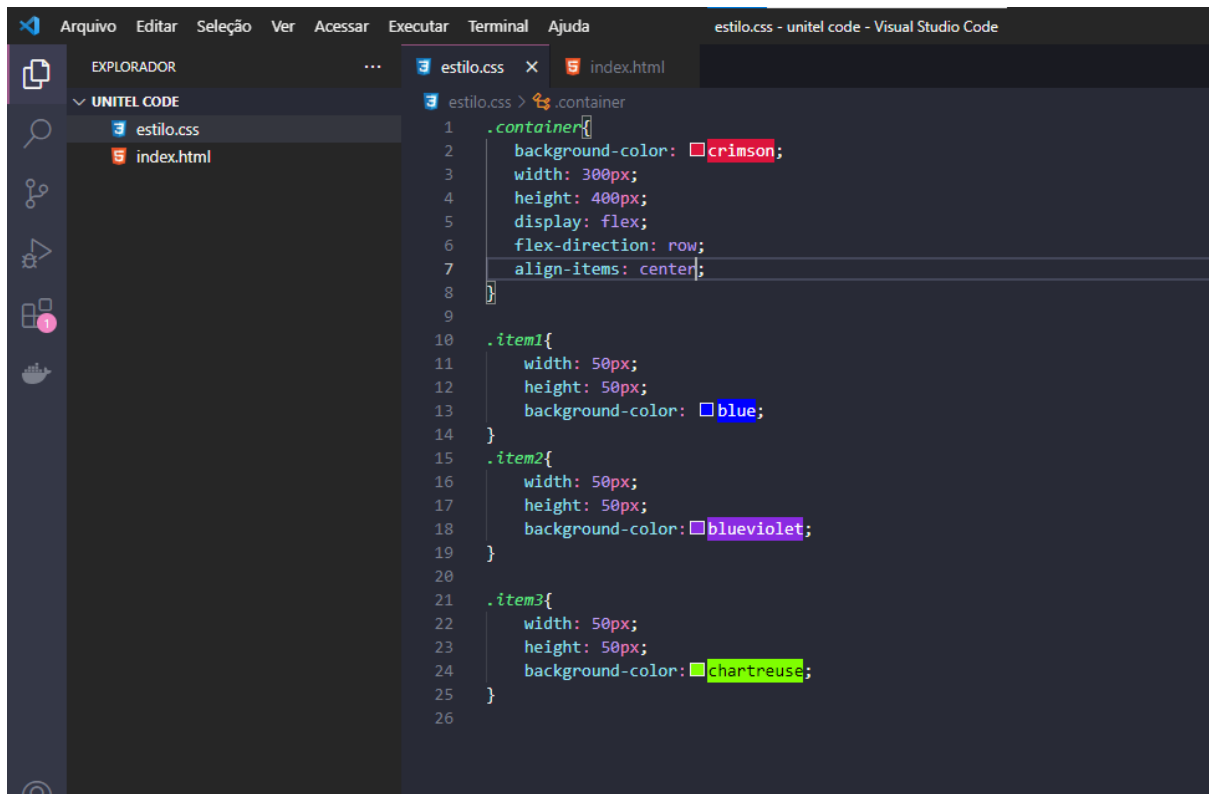


Figura 4 Representação da propriedade justify-content

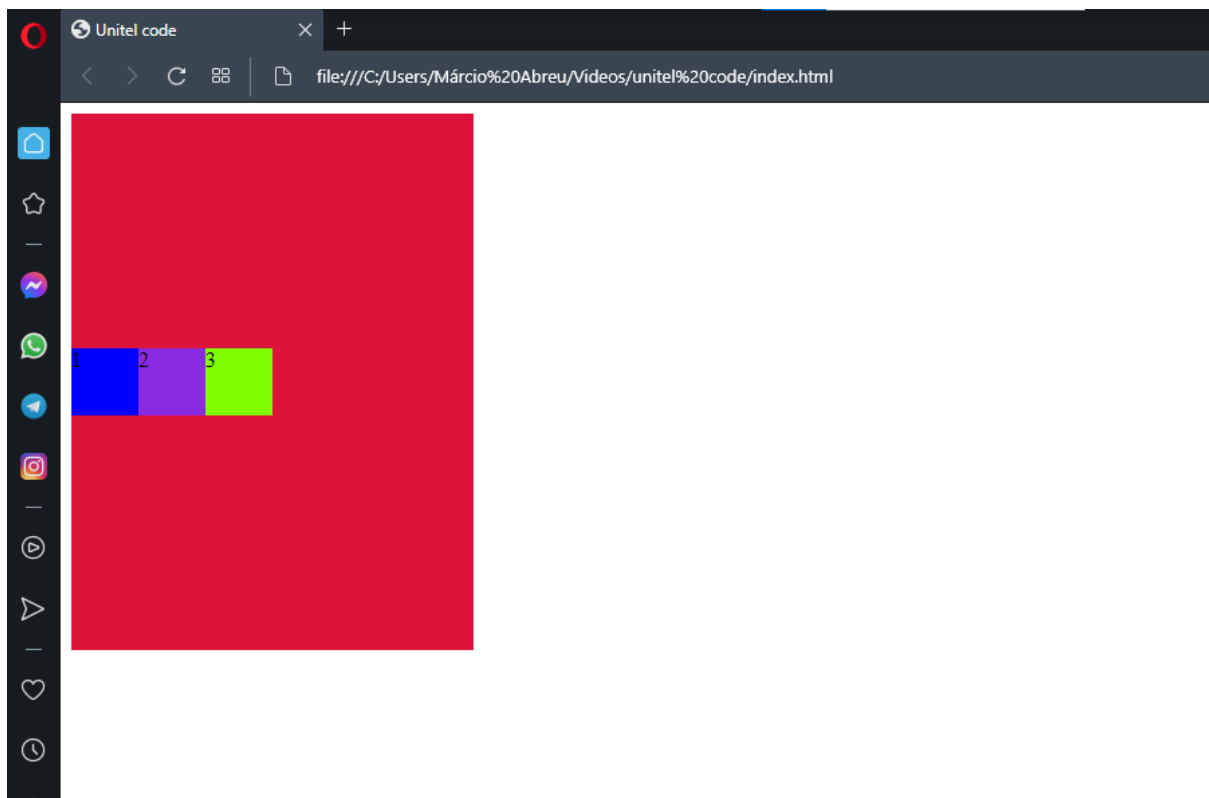
Align-items: Essa propriedade define como os itens são distribuídos ao longo do eixo transversal do container. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying 'estilo.css' and 'index.html' under the 'UNITEL CODE' folder. The main editor area shows the 'estilo.css' file with the following CSS code:

```
1 .container{
2   background-color: crimson;
3   width: 300px;
4   height: 400px;
5   display: flex;
6   flex-direction: row;
7   align-items: center;
8 }
9
10 .item1{
11   width: 50px;
12   height: 50px;
13   background-color: blue;
14 }
15 .item2{
16   width: 50px;
17   height: 50px;
18   background-color: blueviolet;
19 }
20
21 .item3{
22   width: 50px;
23   height: 50px;
24   background-color: chartreuse;
25 }
26
```

Aplicamos a propriedade **align-items:center;** de modo a deixar os elementos filhos no centro.



Nesse momento estamos a ver o resultado. os elementos filhos está no centro.

- **stretch** (padrão): Os itens serão esticados para preencher toda a dimensão do eixo transversal (altura ou largura);
- **flex-start**: Os itens são deslocadas para o início do eixo transversal;
- **flex-end**: Os itens são deslocadas para o final do eixo transversal;
- **center**: Os itens são centralizados no eixo transversal;
- **baseline**: Os itens são alinhados a partir da base da primeira linha de texto de cada um.

A **Figura 6** ilustra o funcionamento de cada valor:

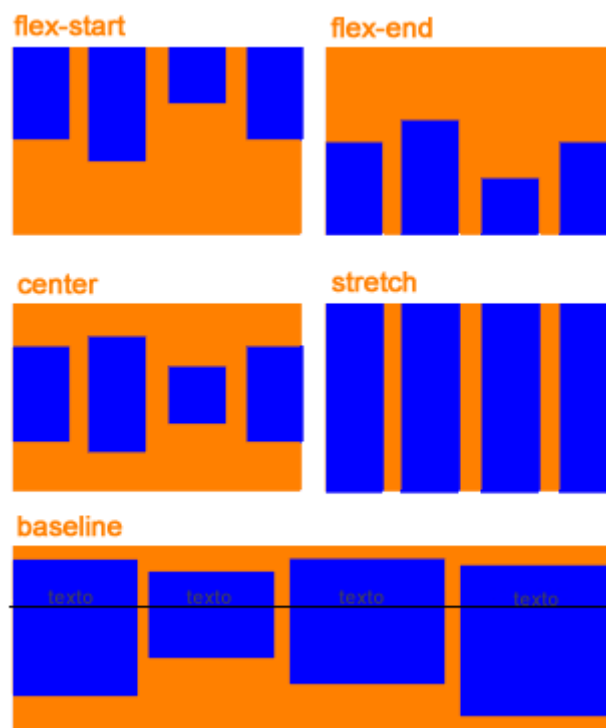
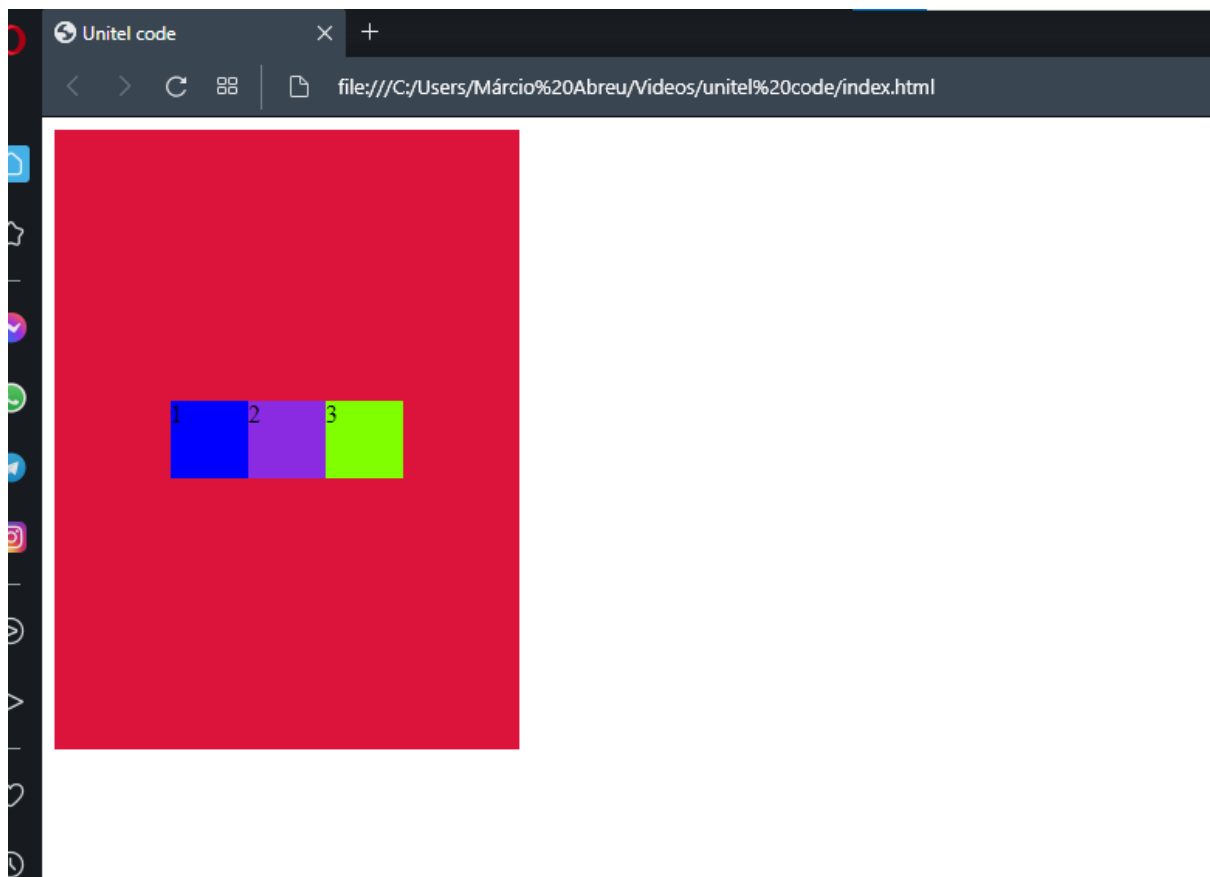


Figura 6. Representação da propriedade align-items

Exercício de Flexbox

1- Colocar os elementos filhos no centro do meio, como mostra a imagem.



formador Márcio D`Abreu: “ *Você só vai aprender a fazer site se você tiver foco*”

Links Úteis

Jogo Interativo Para Praticar Flexbox

<https://flexboxfroggy.com/>