

Desenvolvimento Web

Position, Flexbox e Grid Layout

position

Definindo o posicionamento de elementos

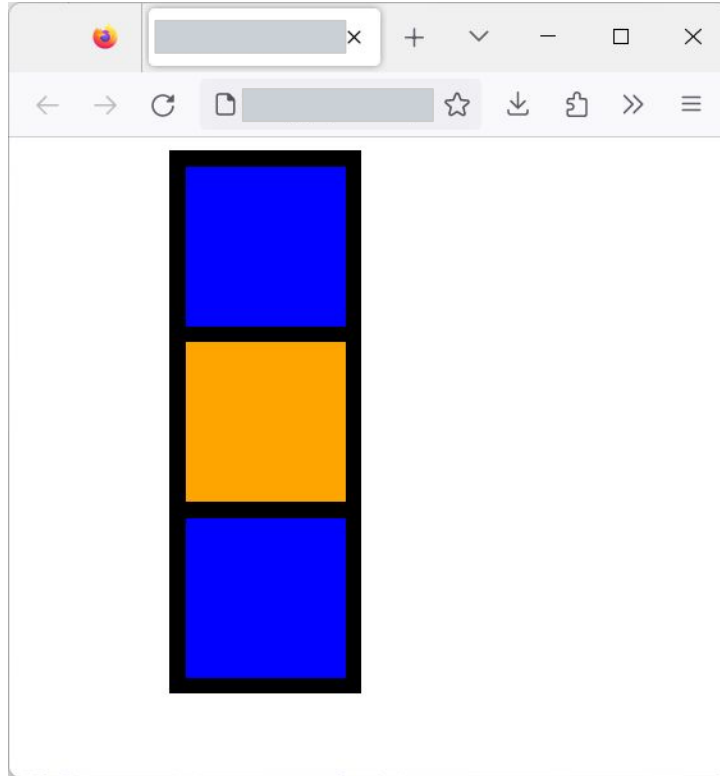
HTML utilizado para exemplo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="estilos/estilo.css" rel="stylesheet">
  </head>
  <body>
    <div id="container">
      <div class="caixa caixa-1"></div>
      <div class="caixa caixa-2"></div>
      <div class="caixa caixa-3"></div>
    </div>
  </body>
</html>
```

CSS Inicial

```
.caixa {  
  width: 100px;  
  height: 100px;  
  background-color: blue;  
  margin: 10px;  
}  
  
.caixa-2 {  
  background-color: orange;  
  position: static;  
}  
  
body div {  
  margin-left: 50px;  
  background-color: black;  
}  
  
#container {  
  position: fixed;  
  left: 50px;  
}
```

Resultado no browser




position

- ▷ A propriedade position define como um elemento pode ser posicionado (renderizado) em uma página HTML.
- ▷ Esta propriedade pode vir acompanhada de outras:
 - top
 - right
 - bottom
 - left

position

- ▷ É possível posicionar o elemento como:
 - relative
 - absolute
 - fixed
 - sticky
- ▷ Por padrão, todo elemento é posicionado como:
 - static

position

- ▷ É possível posicionar o elemento como:
 - relative
 - absolute
 - fixed
 - sticky
- ▷ Por padrão, todo elemento é posicionado como:
 -  ○ static (estático)

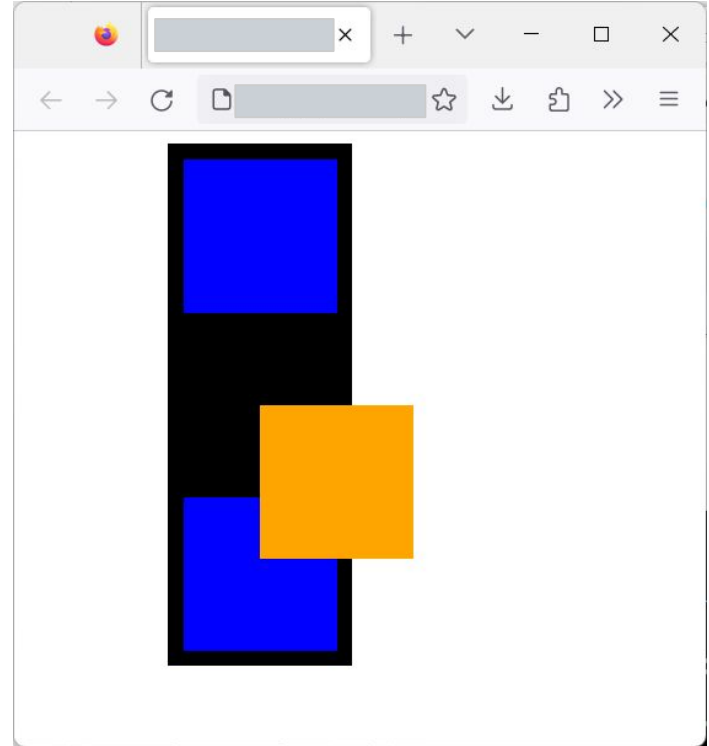
```
//Mantém o posicionamento padrão
.caixa-2 {
    background-color: orange;
    position: static;
}
```


position: relative

- ▷ Para um elemento posicionado como relative:
 - Seu posicionamento será calculado à partir do posicionamento padrão (static).
 - As propriedades top e bottom posicionam verticalmente o elemento.
 - As propriedades left e right posicionam horizontalmente o elemento.

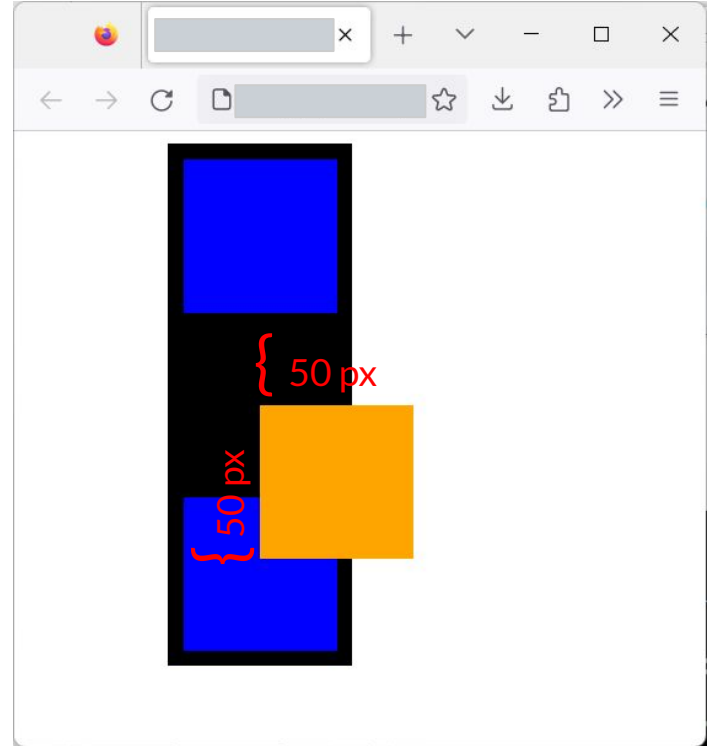
Exemplo → position: relative

```
.caixa-2 {  
  background-color: orange;  
  position: relative;  
  top: 50px;  
  left: 50px;  
}
```



Exemplo → position: relative

```
.caixa-2 {  
  background-color: orange;  
  position: relative;  
  top: 50px;  
  left: 50px;  
}
```

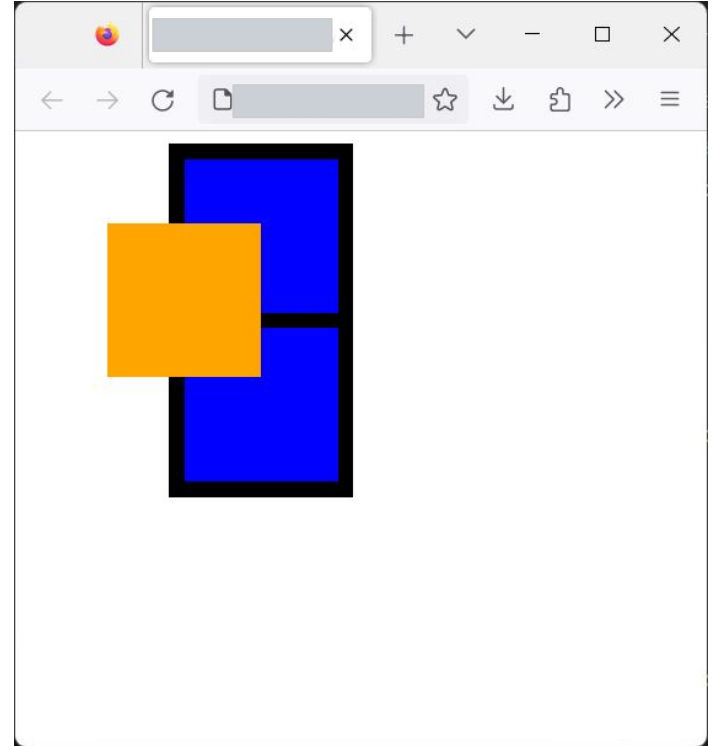


position: fixed

- ▷ Para um elemento posicionado como fixed:
 - Seu posicionamento será calculado em relação à viewport.
 - As propriedades top e bottom posicionam verticalmente o elemento.
 - As propriedades left e right posicionam horizontalmente o elemento.

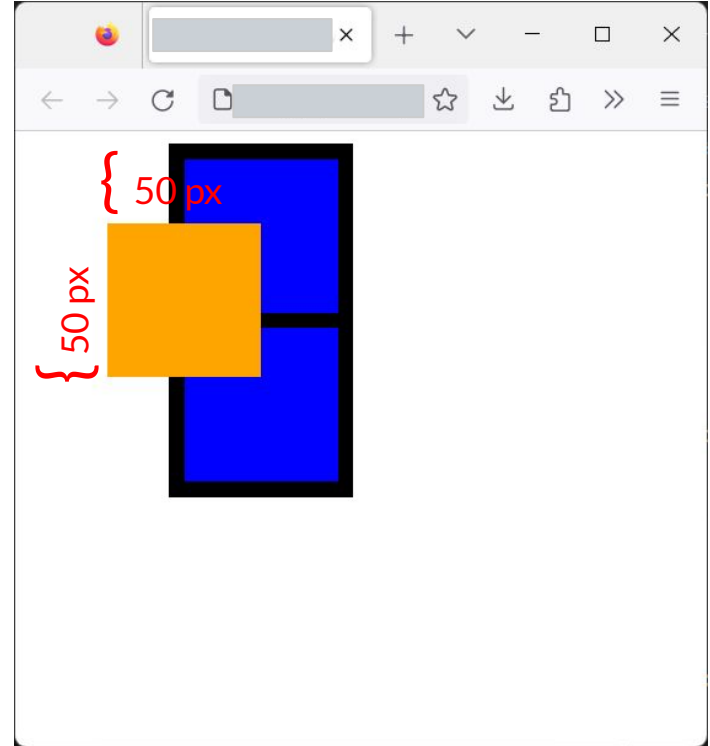
Exemplo → position: fixed

```
.caixa-2 {  
  background-color: orange;  
  position: fixed;  
  top: 50px;  
  left: 50px;  
}
```



Exemplo → position: fixed

```
.caixa-2 {  
  background-color: orange;  
  position: fixed;  
  top: 50px;  
  left: 50px;  
}
```

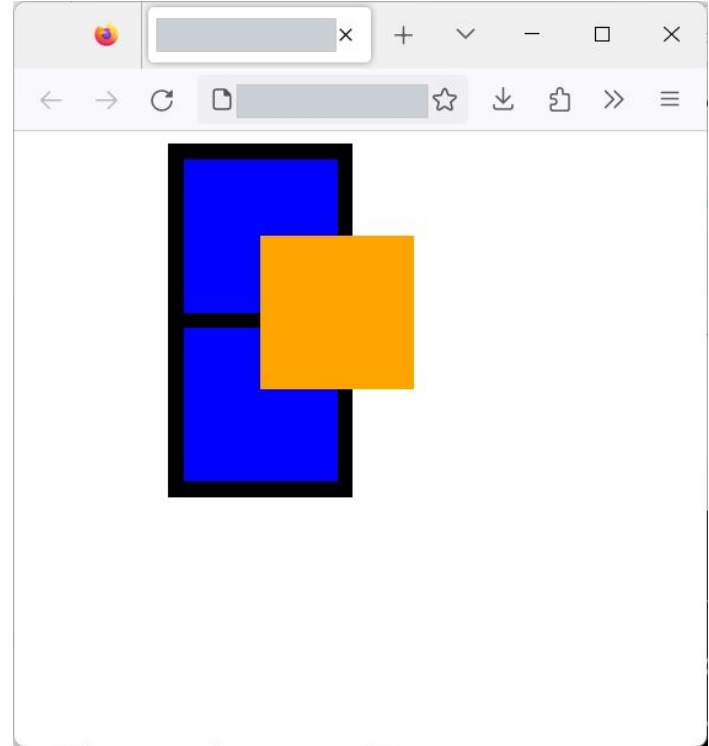


position: absolute

- ▷ Para um elemento posicionado como absolute:
 - Seu posicionamento será calculado em relação ao ancestral posicionado mais próximo.
 - As propriedades top e bottom posicionam verticalmente o elemento.
 - As propriedades left e right posicionam horizontalmente o elemento.

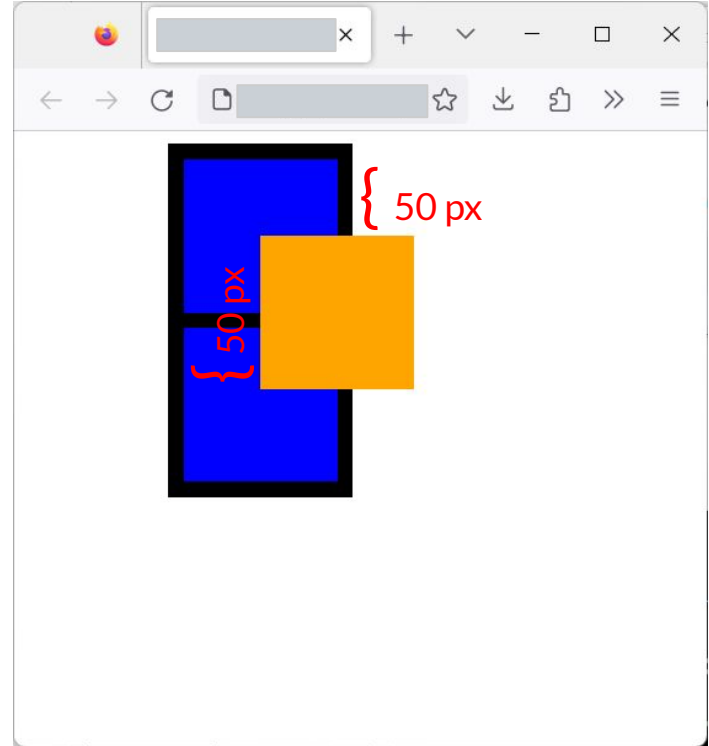
Exemplo → position: absolute

```
.caixa-2 {  
  background-color: orange;  
  position: absolute;  
  top: 50px;  
  left: 50px;  
}
```



Exemplo → position: absolute

```
.caixa-2 {  
  background-color: orange;  
  position: absolute;  
  top: 50px;  
  left: 50px;  
}
```



position: sticky

- ▷ Para um elemento posicionado como sticky (adesivo):
 - Seu posicionamento será relativo até que ultrapasse os limites do bloco no qual está contido. Após isso o posicionamento será fixo.
 - Os limites são dados pelas propriedades:
 - top e bottom
 - left e right

Exemplo → position: sticky

- ▷ Para testarmos este posicionamento, será necessário apagar a regra **#container** do CSS e adicionar a regra abaixo:

```
.caixa-z {  
    background-color: silver;  
}
```

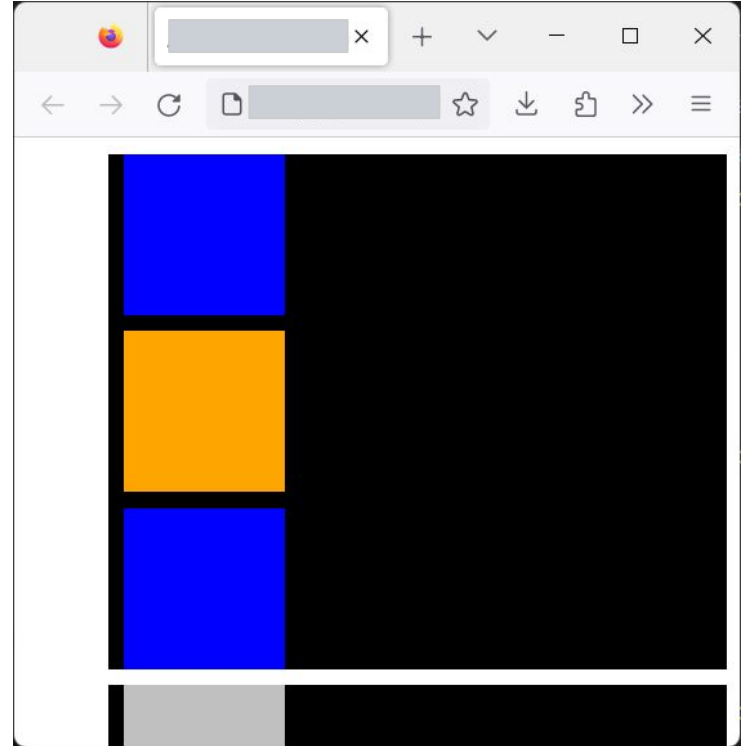
Exemplo → position: sticky

- ▷ Também será necessário alterar o HTML:

[illegible]

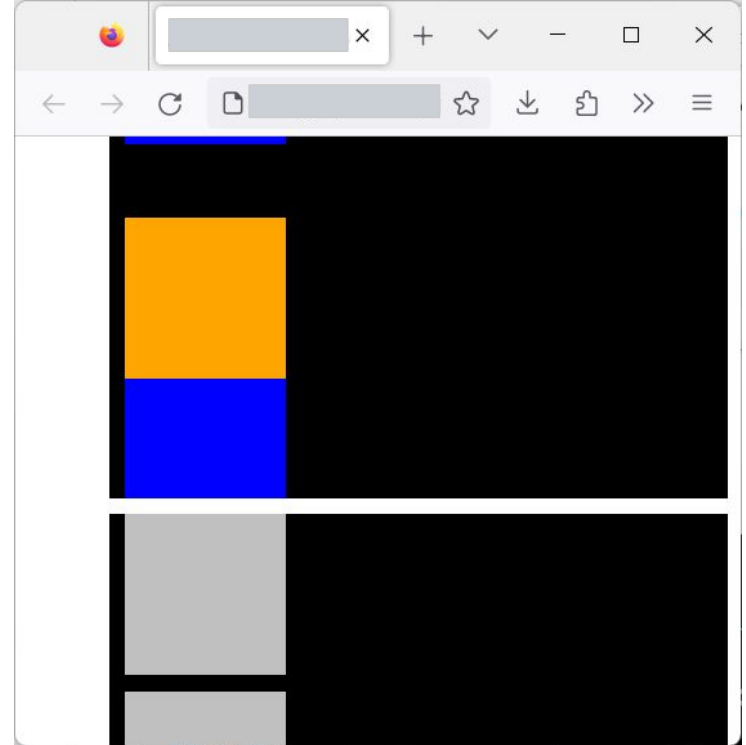
Exemplo → position: sticky

```
.caixa-2 {  
  background-color: orange;  
  position: sticky;  
  top: 50px;  
  left: 50px;  
}
```



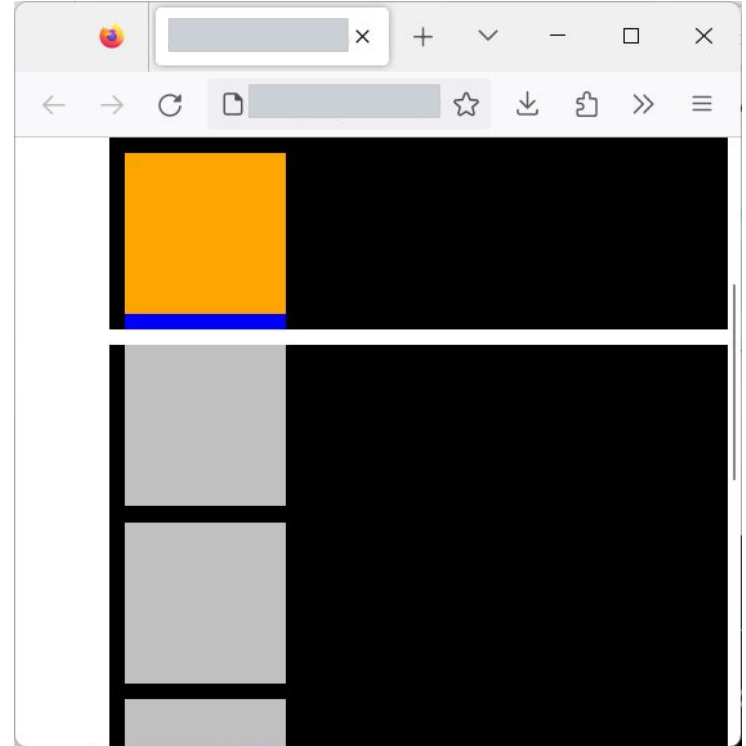
Exemplo → position: sticky (rolagem)

```
.caixa-2 {  
  background-color: orange;  
  position: sticky;  
  top: 50px;  
  left: 50px;  
}
```



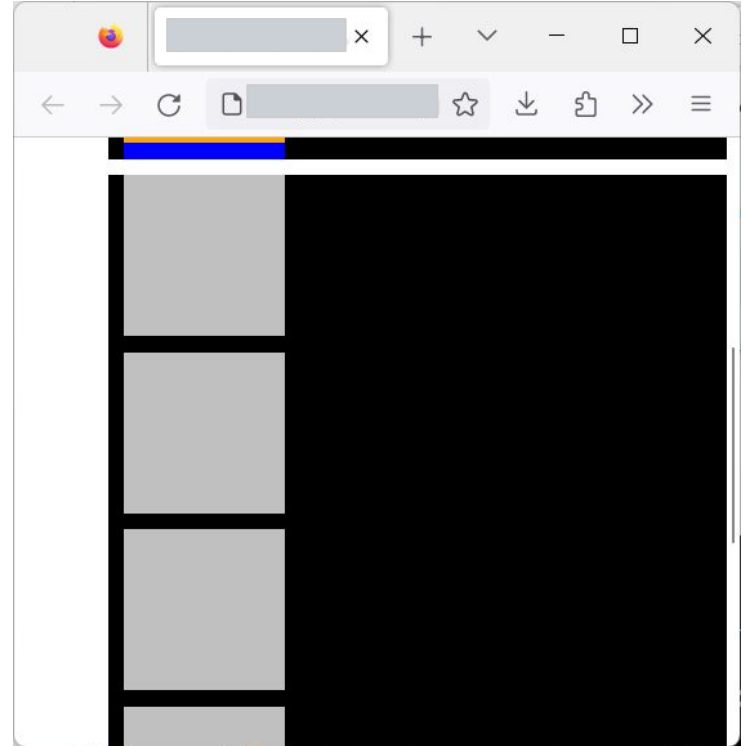
Exemplo → position: sticky (rolagem)

```
.caixa-2 {  
  background-color: orange;  
  position: sticky;  
  top: 50px;  
  left: 50px;  
}
```



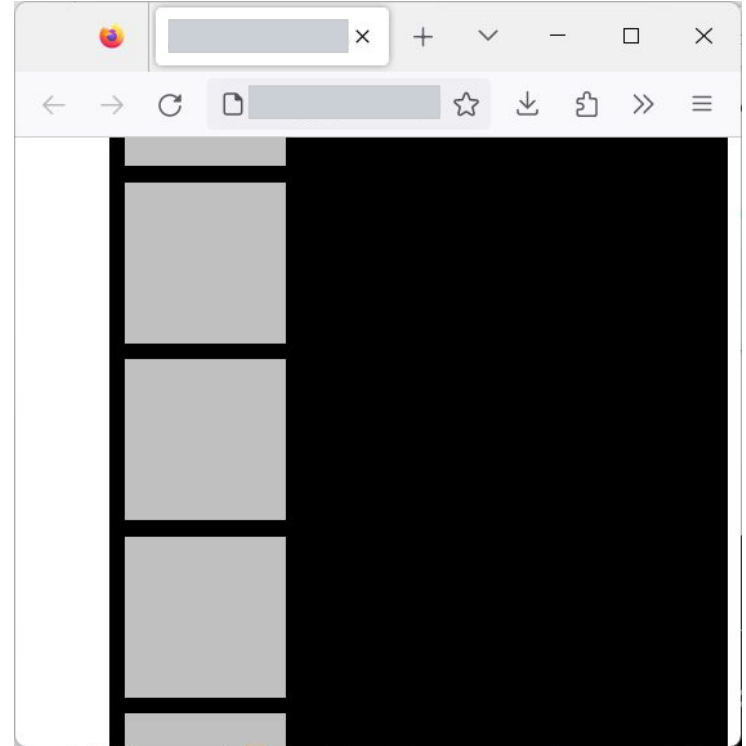
Exemplo → position: sticky (rolagem)

```
.caixa-2 {  
  background-color: orange;  
  position: sticky;  
  top: 50px;  
  left: 50px;  
}
```



Exemplo → position: sticky (rolagem)

```
.caixa-2 {  
  background-color: orange;  
  position: sticky;  
  top: 50px;  
  left: 50px;  
}
```



CSS layout

Flexbox e Grid Layout

Flexbox

- ▷ O flexbox disponibiliza ferramentas para criação de layouts complexos e flexíveis.
- ▷ Você pode utilizar o flexbox para organizar elementos em uma dimensão.

HTML utilizado para exemplo

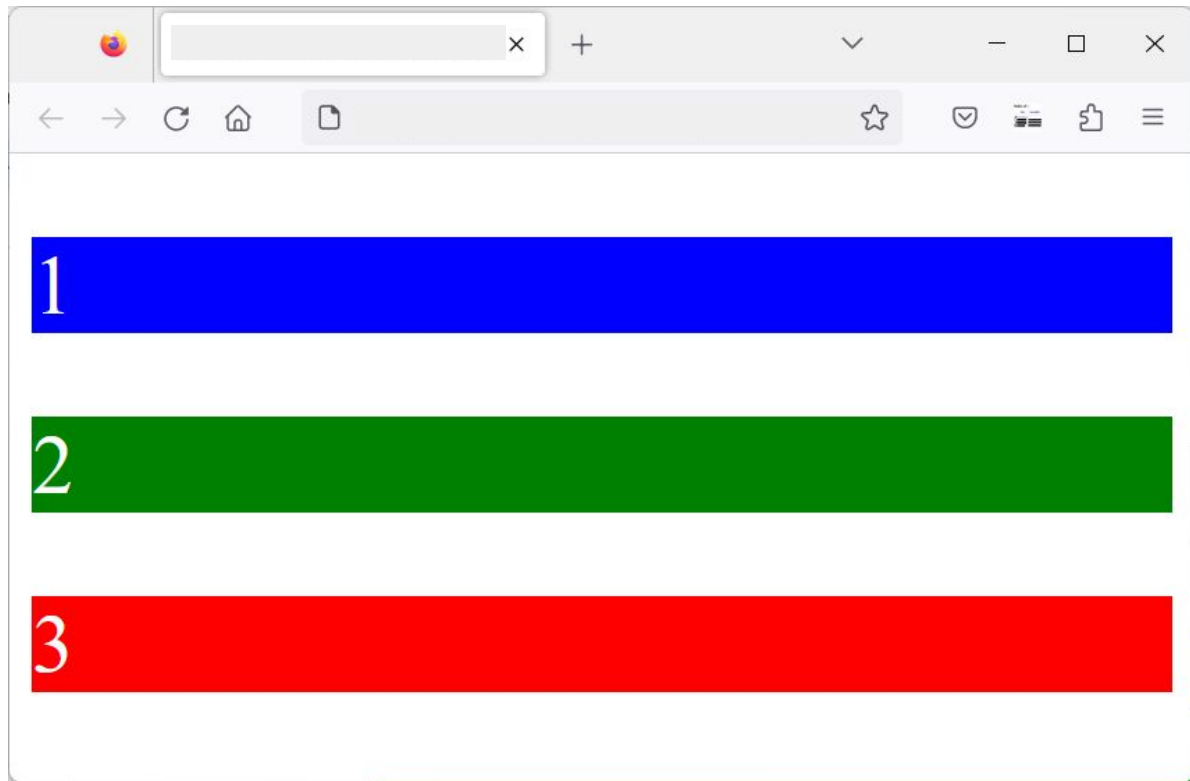
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="estilos/estilo.css" rel="stylesheet">
  </head>
  <body>
    <div id="container">
      <div class="caixa caixa-1"><p>1</p></div>
      <div class="caixa caixa-2"><p>2</p></div>
      <div class="caixa caixa-3"><p>3</p></div>
    </div>
  </body>
</html>
```

CSS Inicial

```
.caixa {  
    margin: 5px;  
}  
  
.caixa p {  
    color: white;  
    font-size: 3em;  
}
```

```
.caixa-1 {  
    background-color: blue;  
}  
  
.caixa-2 {  
    background-color: green;  
}  
  
.caixa-3 {  
    background-color: red;  
}
```

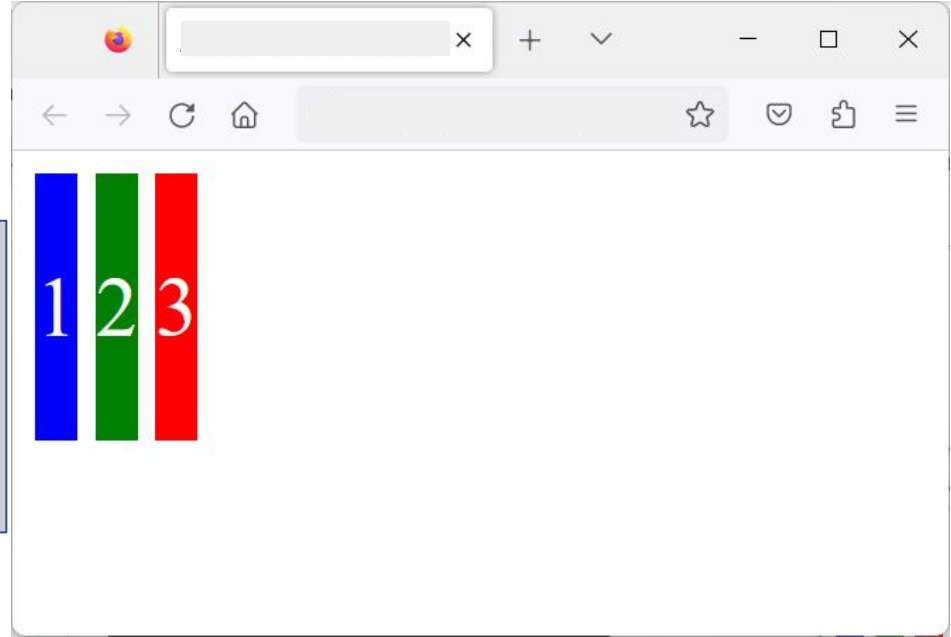
Resultado antes do flexbox



display:flex

```
#container {  
  display: flex;  
}
```

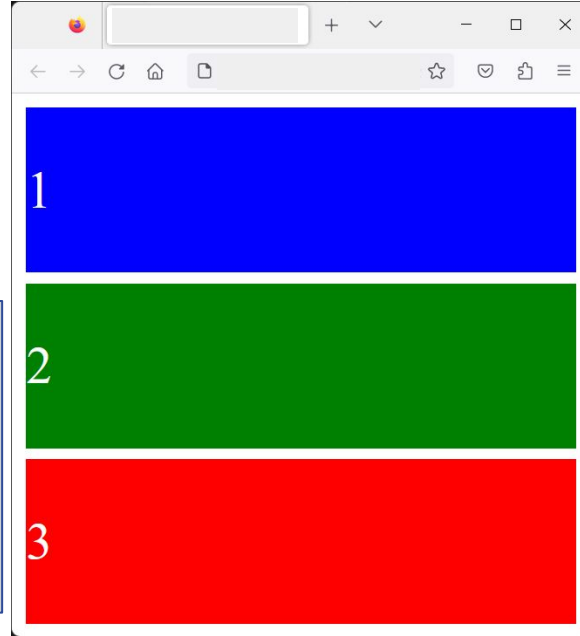
Ao definir a propriedade display com o valor flex, todos os filhos do elemento container se tornam itens do flexbox.



flex-direction

```
#container {  
  display: flex;  
  flex-direction: column;  
}
```

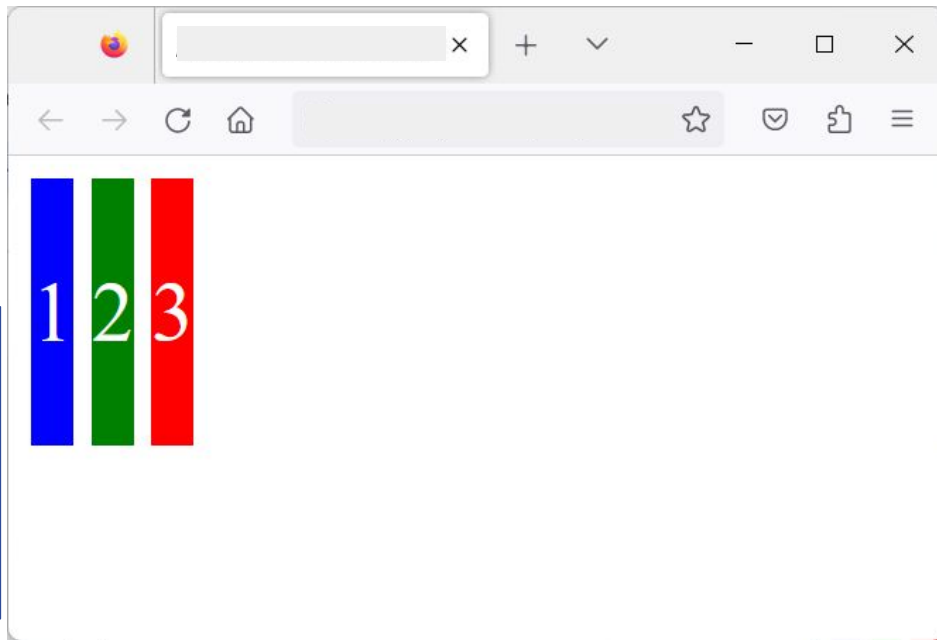
É possível mudar a disposição dos itens para coluna.



flex-direction

```
#container {  
  display: flex;  
  flex-direction: row;  
}
```

A disposição padrão é em linha.



flex:wrap

```
#container {  
  display: flex;  
  flex-wrap: wrap;  
}
```

Permite a quebra de linha flexível.

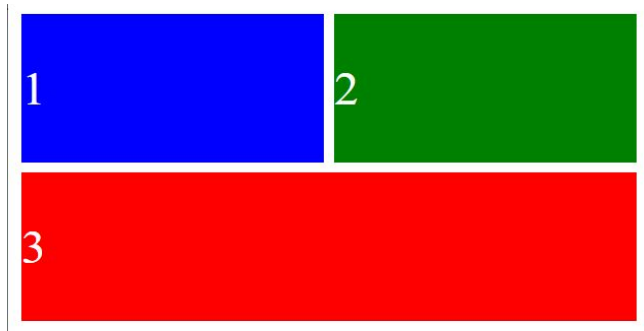
```
.caixa {  
  margin: 5px;  
  flex: 300px;  
}
```

Tamanho mínimo de cada item flexível.



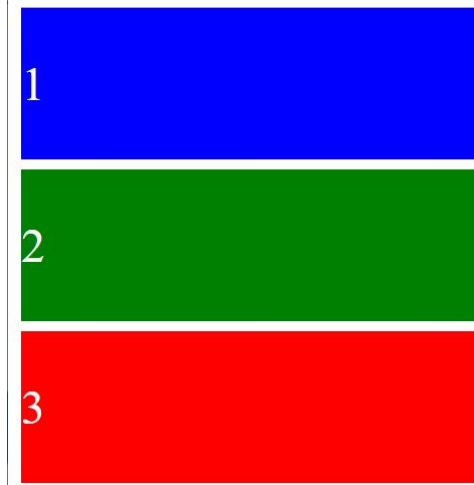
flex:wrap

```
#container {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.caixa {  
  margin: 5px;  
  flex: 300px;  
}
```



flex:wrap

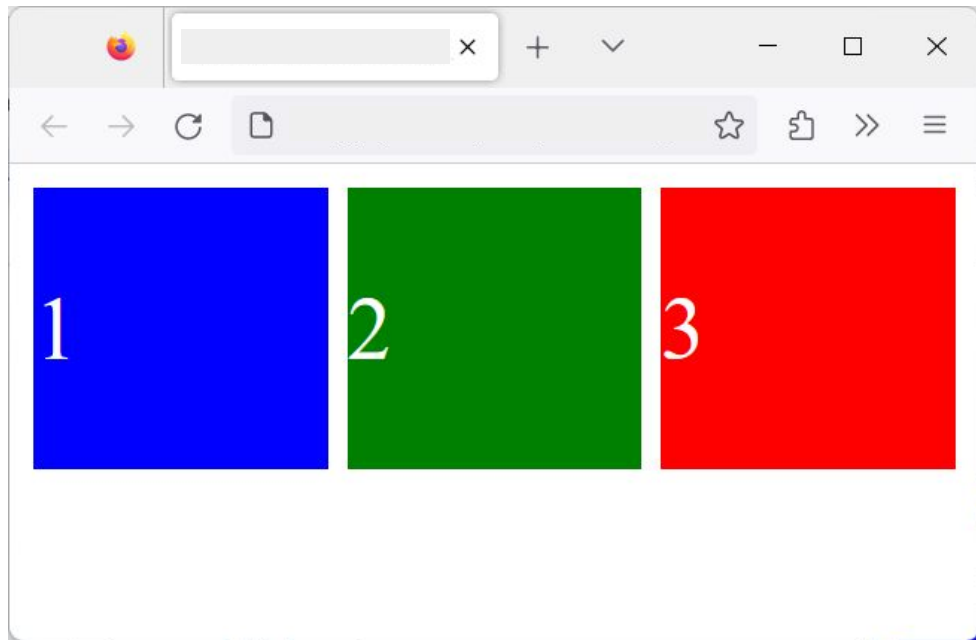
```
#container {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.caixa {  
  margin: 5px;  
  flex: 300px;  
}
```



flex:1

```
.caixa {  
  margin: 5px;  
  flex: 1;
```

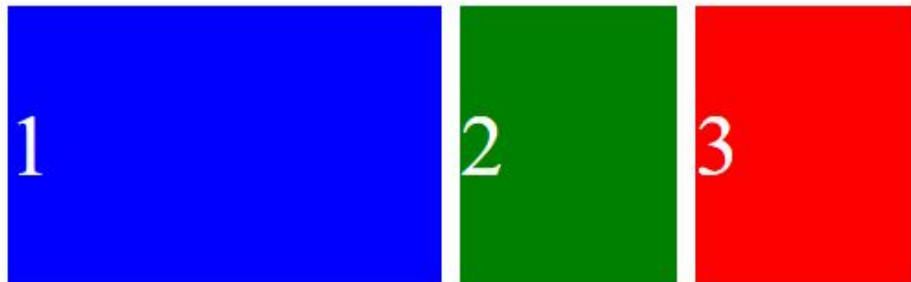
Todos os itens possuem a mesma proporção (1).



flex:?

```
.caixa {  
    margin: 5px;  
    flex: 1;  
}  
  
.caixa-1 {  
    flex: 2;  
}
```

O primeiro item possui a proporção 2.



alinhamento

```
.caixa {  
  margin: 5px;  
  flex: 1;  
  display: flex;  
  align-items: center;  
  justify-content: space-around;  
}
```



Grid Layout

- ▷ Um grid é um conjunto de linhas horizontais e verticais.
- ▷ Eles nos ajudam a criar layouts nos quais nossos elementos não saltam nem mudam de largura à medida que avançamos de uma página para outra, proporcionando maior consistência em nossos sites.

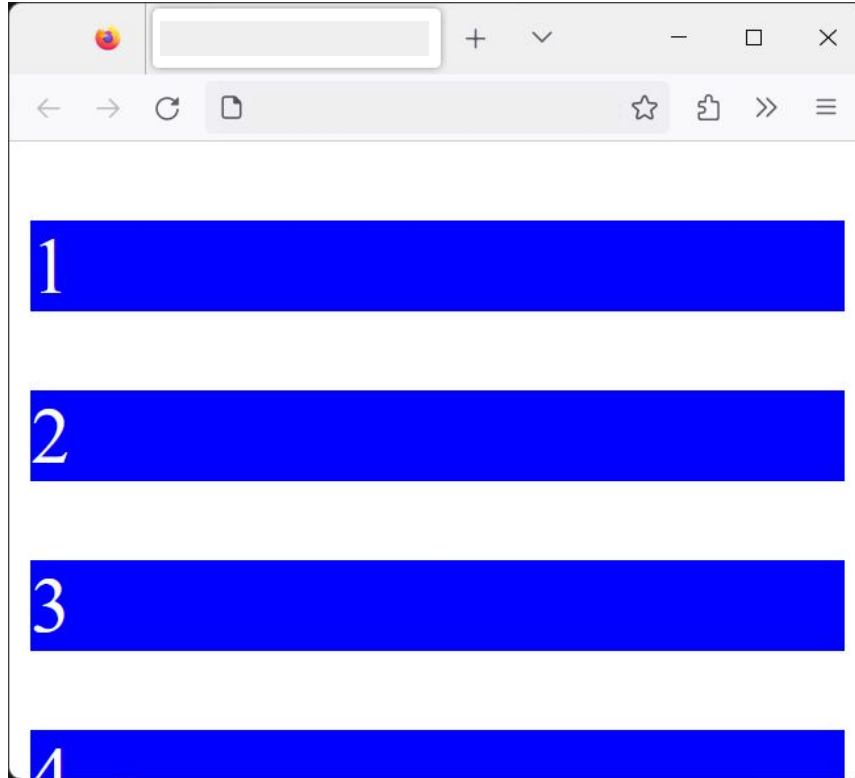
HTML utilizado para exemplo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="estilos/estilo.css" rel="stylesheet">
  </head>
  <body>
    <div id="container">
      <div class="cx"><p>1</p></div>
      <div class="cx"><p>2</p></div>
      <div class="cx"><p>3</p></div>
      <div class="cx"><p>4</p></div>
      <div class="cx"><p>5</p></div>
      <div class="cx"><p>6</p></div>
      <div class="cx"><p>7</p></div>
    </div>
  </body>
</html>
```

CSS Inicial

```
.cx {  
  background-color: blue;  
  margin: 5px;  
}  
  
.cx p {  
  color: white;  
  font-size: 3em;  
}
```

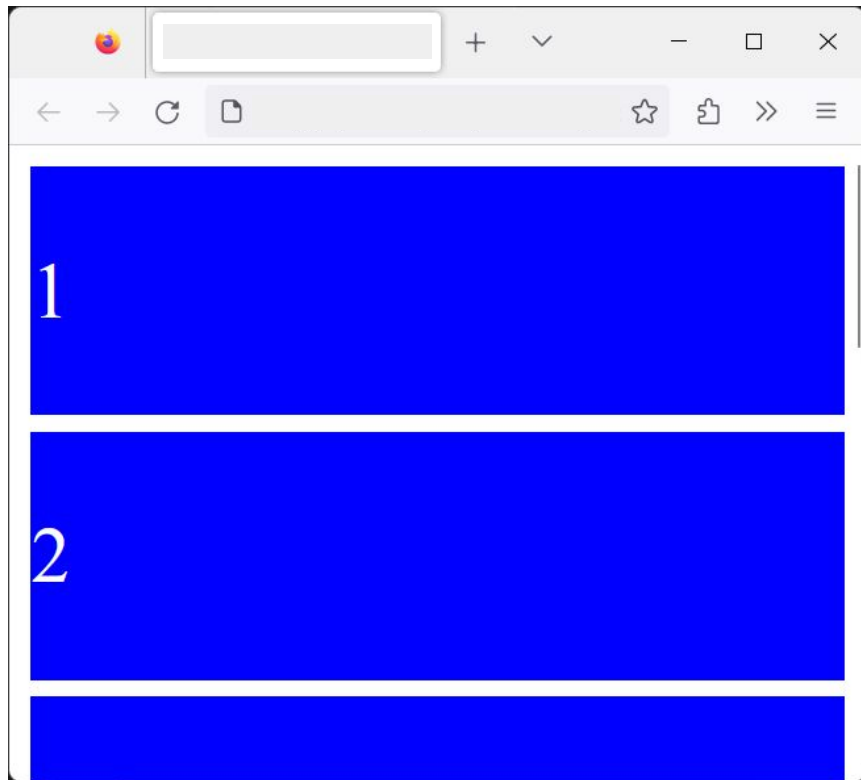
Resultado antes do grid



display:grid

```
#container {  
  display: grid;  
}
```

Ao definir a propriedade display com o valor grid, todas os filhos do elemento container se tornam itens do grid.

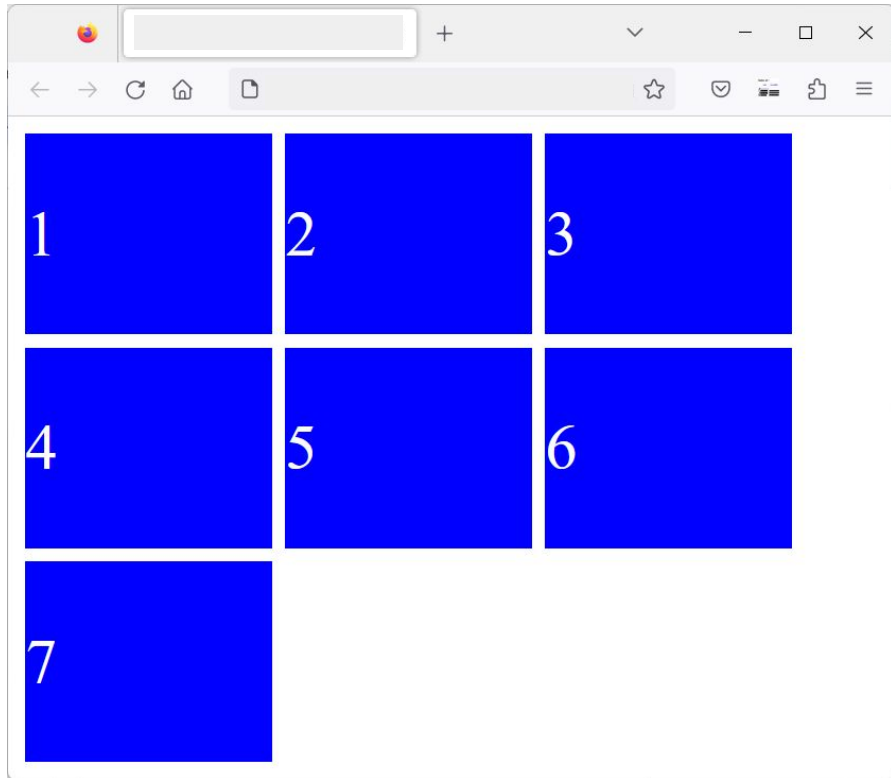


display:grid

```
#container {  
  display: grid;  
  grid-template-columns: 30% 30% 30%;  
}
```

É possível definir o número de colunas e o tamanho das mesmas. Experimente também:

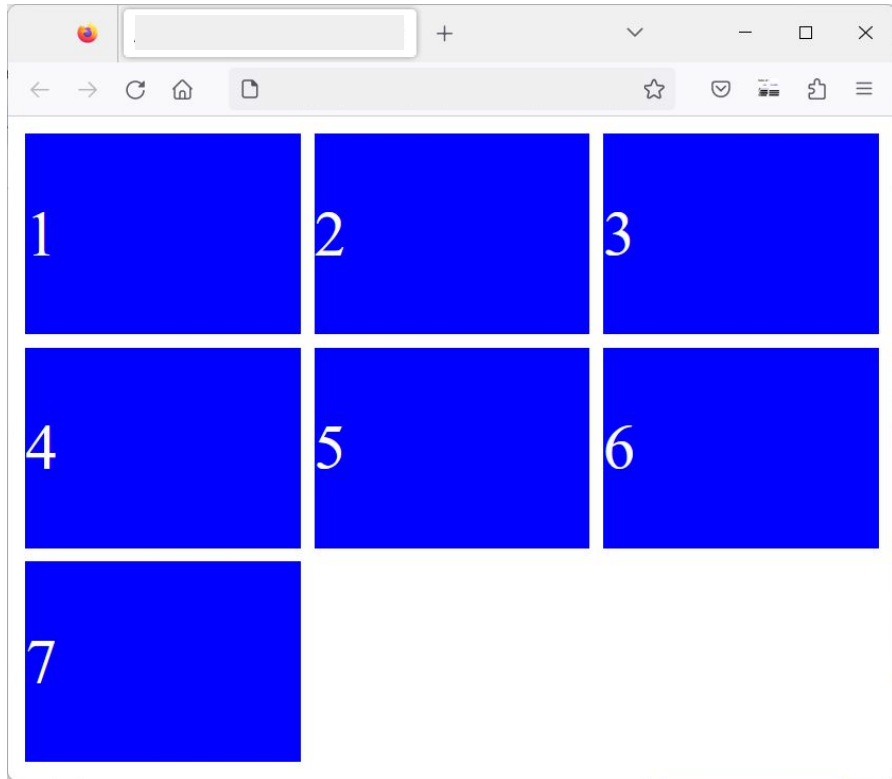
```
grid-template-columns: 100px 100px;
```



display:grid

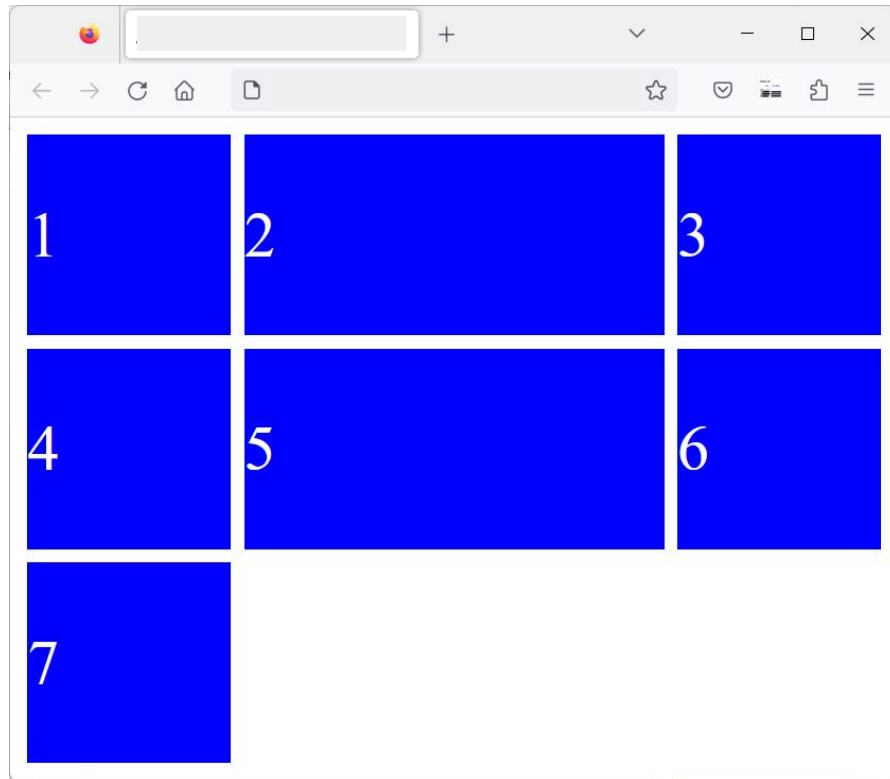
```
#container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
}
```

Você pode definir o tamanho de cada coluna como frações. O número que vem antes de fr indica a proporção de cada elemento.



display:grid

```
#container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
}
```

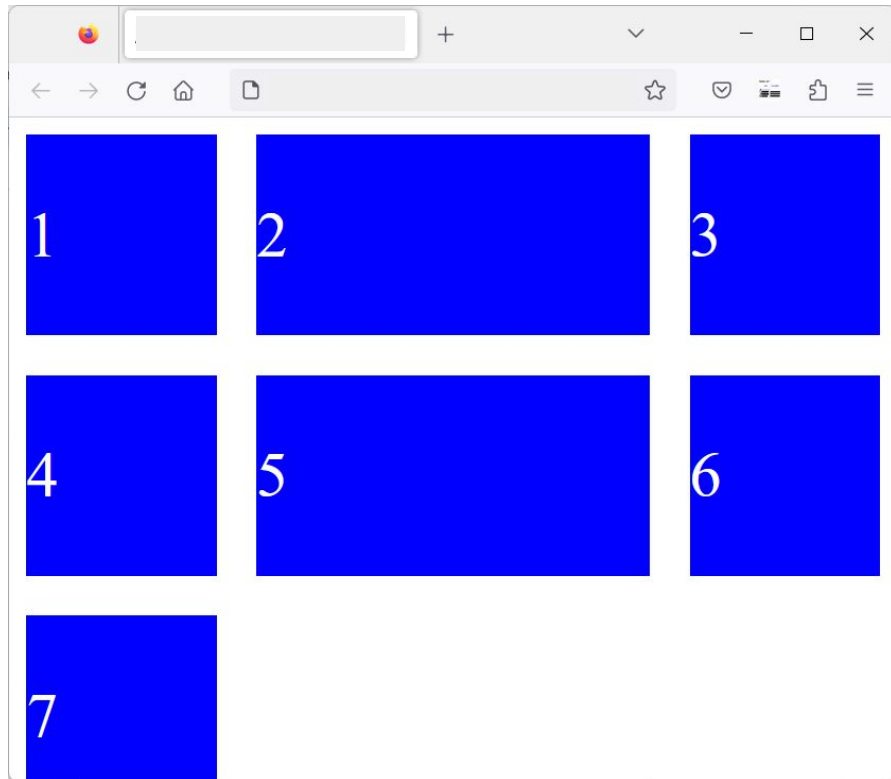


gap

```
#container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
  gap: 20px;  
}
```

É possível definir a distância entre as linhas e as colunas.

Experimente também as propriedades:
column-gap
row-gap

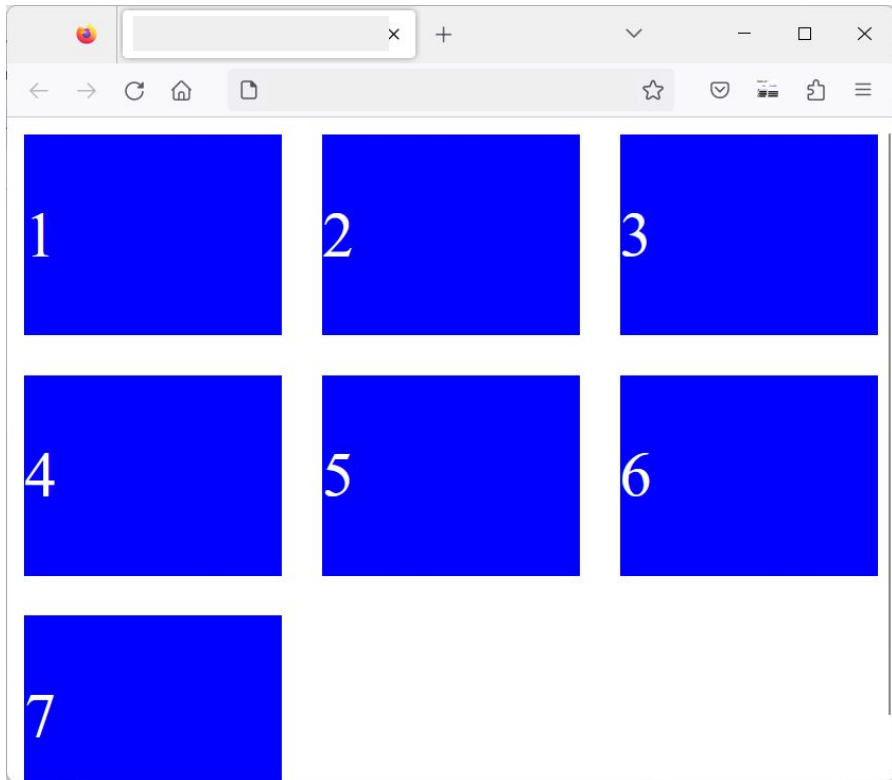


repeat

```
#container {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 20px;
```

Repete a informação à direita o número de vezes apresentado na esquerda. Experimente também:

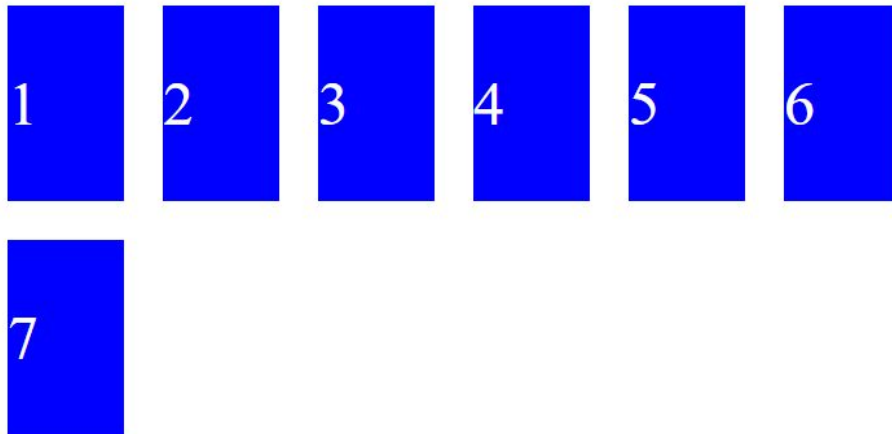
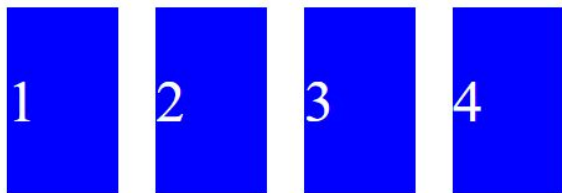
```
grid-template-columns: repeat(4, 1fr);
```



repeat

```
#container {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, 100px);  
  gap: 20px;  
}
```

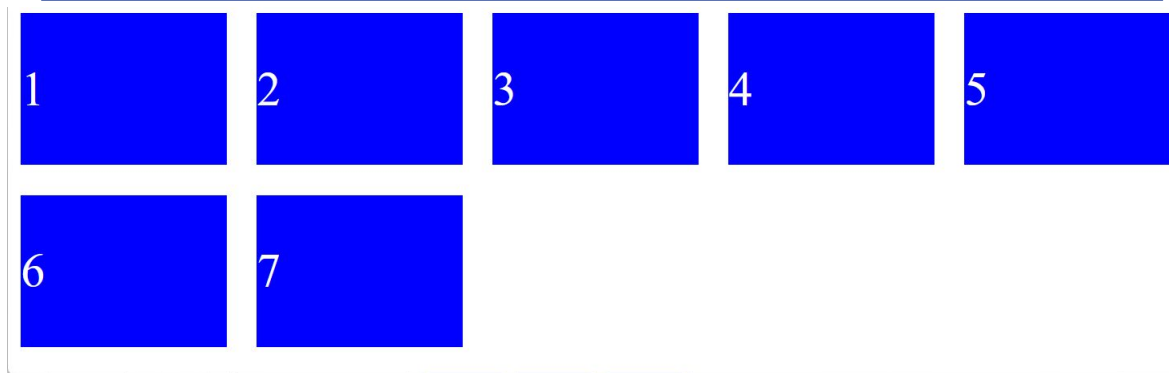
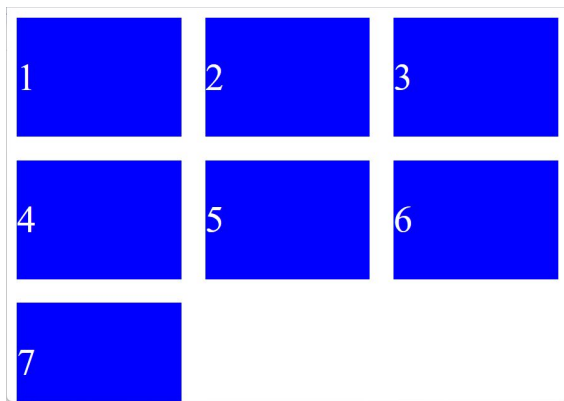
O número de elementos será calculado automaticamente em acordo com o tamanho à direita e a view port.



repeat

```
#container {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
  gap: 20px;  
}
```

O tamanho mínimo de cada coluna será de 200 pixels, porém este tamanho será proporcional a 1 fr sempre que puder ser maior que 200 pixels.

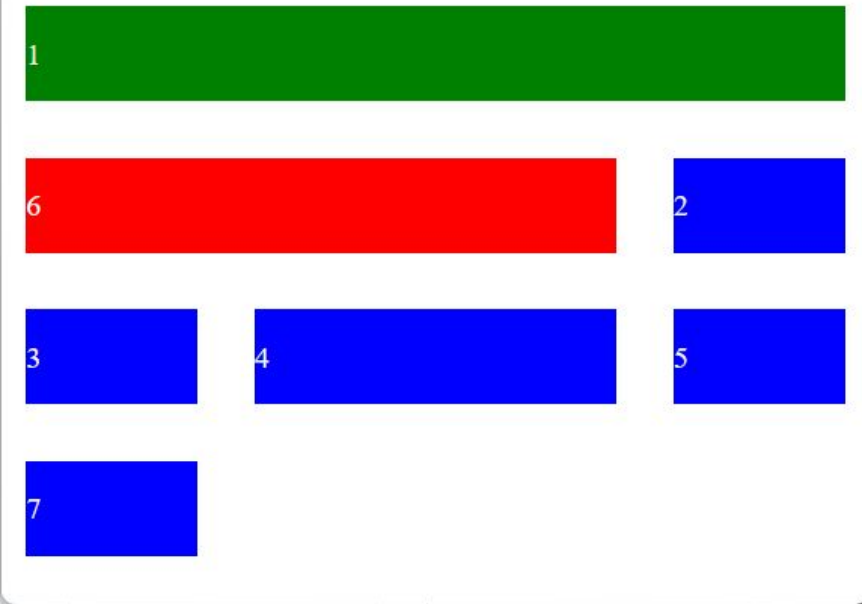


Alteração no HTML utilizado

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="estilos/estilo.css" rel="stylesheet">
  </head>
  <body>
    <div id="container">
      <div class="cx cx1"><p>1</p></div>
      <div class="cx cx2"><p>2</p></div>
      <div class="cx cx3"><p>3</p></div>
      <div class="cx cx4"><p>4</p></div>
      <div class="cx cx5"><p>5</p></div>
      <div class="cx cx6"><p>6</p></div>
      <div class="cx cx7"><p>7</p></div>
    </div>
  </body>
</html>
```

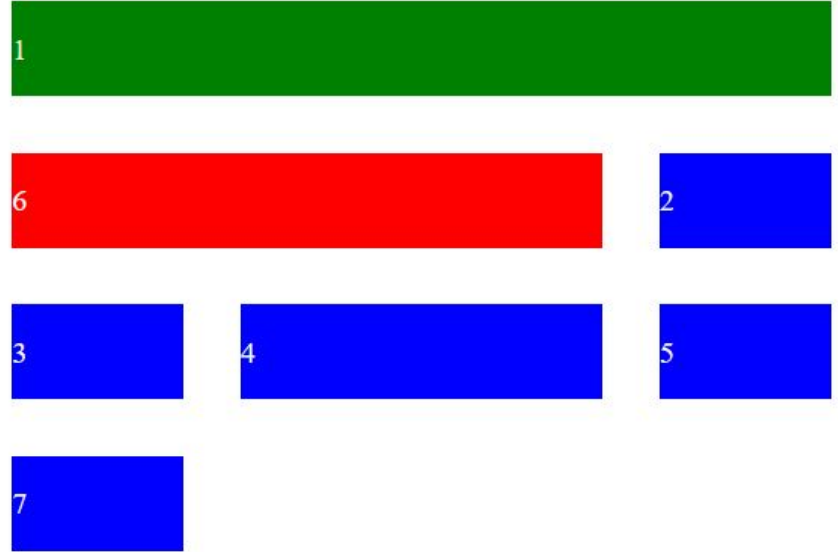
Alteração no CSS utilizado

```
#container {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  gap: 20px  
}  
  
.cx {  
  background-color: blue;  
}  
  
//Continua no próximo slide...
```



Alteração no CSS utilizado

```
.cx p {  
  color: white;  
  font-size: 1em;  
}  
  
.cx1 {  
  grid-column: 1 / 4;  
  grid-row: 1;  
  background-color: green;  
}  
  
.cx6 {  
  grid-column: 1 / 3;  
  grid-row: 2;  
  background-color: red;  
}
```



Alteração no CSS utilizado

```
.cx p {  
  color: white;  
  font-size: 1em;  
}
```

Coluna inicial: 1. Parar antes da
coluna: 4

```
.cx1 {  
  grid-column: 1 / 4;  
  grid-row: 1;  
  background-color: green;  
}
```

Coluna inicial: 1. Parar antes da
coluna: 3

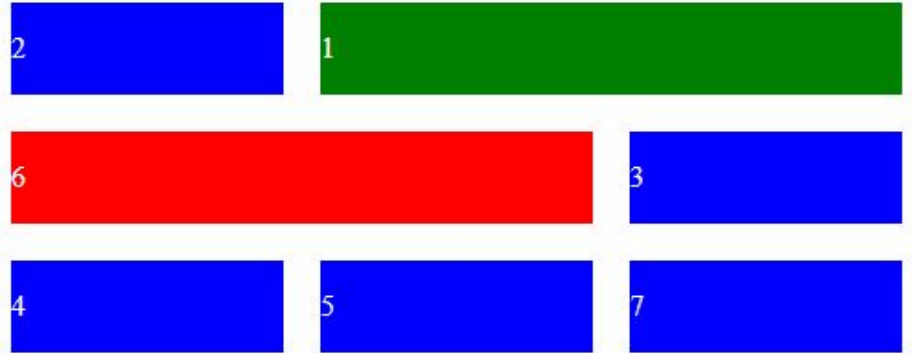
```
.cx6 {  
  grid-column: 1 / 3;  
  grid-row: 2;  
  background-color: red;  
}
```



Alteração no CSS utilizado

```
.cx p {  
  color: white;  
  font-size: 1em;  
}  
  
.cx1 {  
  grid-column: 2 / 4;  
  grid-row: 1;  
  background-color: green;  
}  
  
.cx6 {  
  grid-column: 1 / 3;  
  grid-row: 2;  
  background-color: red;  
}
```

Coluna inicial: 2. Parar antes da
coluna: 4



Fim de material

Dúvidas?