

## CSS Grid

É uma ferramenta de CSS que permite facilitar a criação de layouts, a diferença entre Flexbox e Grid é que o Flexbox trabalha de maneira unidimensional (linha), onde essa linha é quebrada de acordo com a largura de tela, já o grid é bidimensional, trabalhando com linhas e colunas, as duas ferramentas podem ser usadas juntamente em um mesmo bloco.

No grid dividimos o container em colunas onde cada item irá ocupar uma certa quantidade de colunas, ou linhas.



*fonte: [www.w3schools.com](http://www.w3schools.com)*

Assim como o Flexbox, utilizamos o grid utilizando o mesmo princípio de container e itens, onde os itens se encontram dentro do container e temos propriedades para específicas para os dois.

Para iniciar aplicamos alteramos a propriedade display no container onde podemos colocar o valor grid (trata os itens como bloco) ou o inline-grid (trata os elementos como elemento de linha).

Como o Grid conta com um grande número de propriedades, o foco da apostila serão as principais propriedades para o desenvolvimento de um layout.

```
<div class="container">
  <p>texto1</p>
  <p>texto2</p>
  <p>texto3</p>
  <p>texto4</p>
  <p>texto5</p>
```

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

## Propriedades para o Contêiner

### grid-template-columns

É a propriedade que usamos para definir quantas colunas queremos no layout, o valor que colocamos nessa propriedade é a largura que cada coluna terá.

Exemplo1:

Criamos 3 colunas onde a primeira e última tem 200px e a do meio 300px.

```
grid-template-columns: 200px 300px 200px;
```

Resultado:

texto1	texto2	texto3
texto4	texto5	

Como resultado, dos 5 itens, 3 ficaram na primeira linha e o restante foi para a segunda linha, neste caso o layout conta com 3 colunas e 2 linhas.

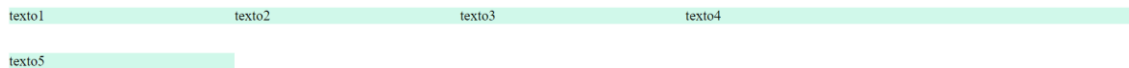
Exemplo2:

No próximo exemplo vamos usar uma unidade de medida criada especificamente para criar as colunas da grid que é o fr (fração) onde o valor dado é a quantidade que determinado item vai ocupar da fração da largura do container.

No exemplo criamos 4 colunas onde a última terá o dobro da largura:

```
grid-template-columns: 1fr 1fr 1fr 2fr;
```

Resultado:

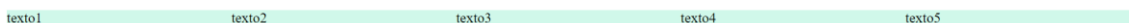


caso o objetivo é criar várias colunas com a mesma largura, podemos usar o comando repeat para tornar o código mais prática:

no exemplo vamos criar 5 colunas com o valor 1fr:

```
grid-template-columns: repeat(5,1fr);
```

Resultado:



## grid-template-rows

Essa propriedade é parecida com a grid-template-columns, a diferença é que irá determinar a quantidade de linhas e a altura de cada uma delas, não é uma tão essencial quanto a anterior, pois é possível deixar em aberto a quantidade de linhas e criá-las conforme a necessidade.

## row-gap e column-gap

Por padrão as colunas e linhas usam todo espaço disponível dentro do container, mas as vezes queremos um espaçamento entre as linhas e colunas da “tabela” criada pela grid, para isso usamos as propriedades.

As duas propriedades servem para criar esse espaçamento sem precisar usar a propriedade margin, row-gap para linha e column-gap para coluna.

## Gap

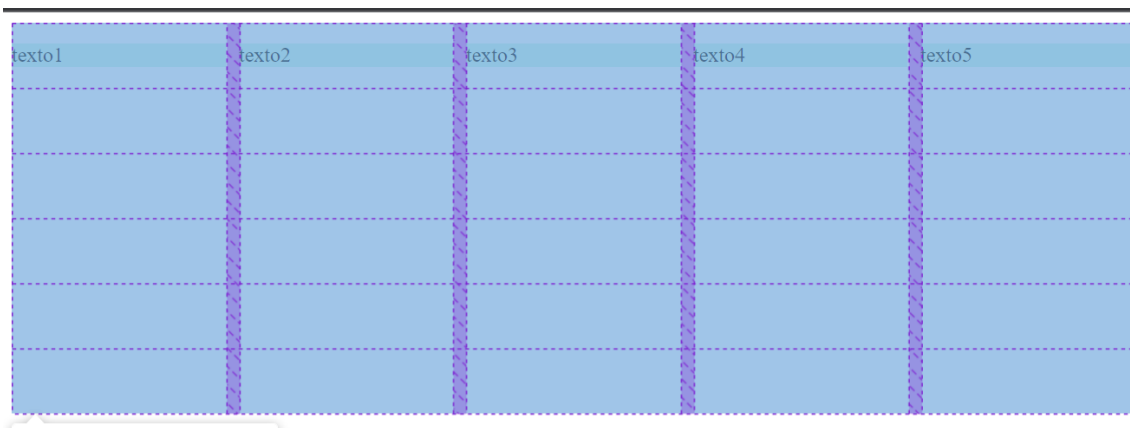
As duas propriedades podem ser resumidas em uma única propriedade gap onde primeiro colocamos o valor do row-gap e em seguida o valor do column-gap.

## Grid-template-areas

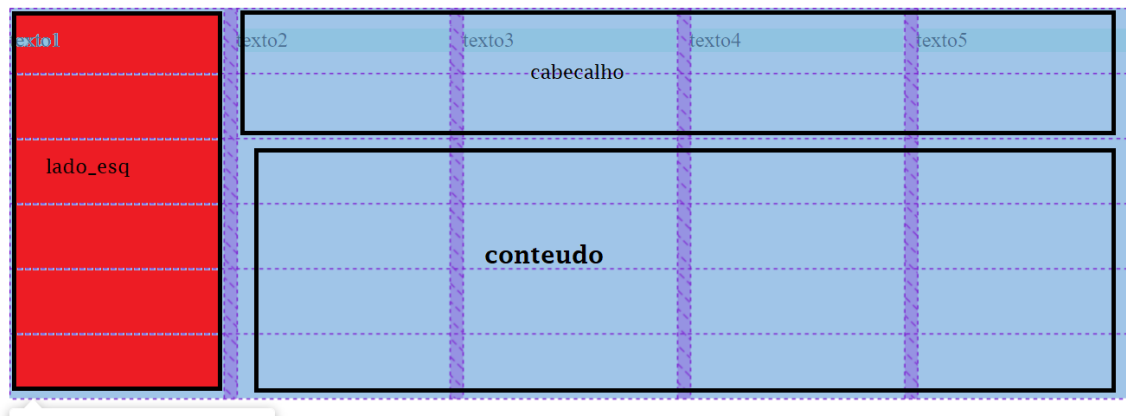
Com essa propriedade é possível nomear os blocos dentro da grid e posteriormente distribuí-lo entre o conteúdo, considere o seguinte código:

```
.container{  
  display:grid;  
  grid-template-columns: repeat(5,1fr);  
  grid-template-rows: repeat(6,1fr);  
  column-gap: 10px;  
}
```

Neste momento nossa grid está dividida em 5 colunas e 6 linhas com tamanhos igual, gerando o seguinte resultado:



Mas digamos que queremos dividir essa tabela em algumas partições:



Pela imagem teremos os seguintes blocos:

- “lado\_esq” que irá ocupar somente a coluna 1 e as linhas de 1 até 6.
- “cabecalho” que irá ocupar da coluna 2 até 5 e das linhas 1 e 2.
- “conteudo” que irá ocupar da coluna 2 até 5 e da linha 3 até 6.

Para criar essas divisões dentro da grid usamos a propriedade `grid-template-areas` distribuindo o nome que cada bloco irá ocupar da seguinte maneira.

```
grid-template-areas: "lado_esq cabecalho cabecalho cabecalho cabecalho"
                    "lado_esq cabecalho cabecalho cabecalho cabecalho"
                    "lado_esq conteudo conteudo conteudo conteudo"
                    "lado_esq conteudo conteudo conteudo conteudo"
                    "lado_esq conteudo conteudo conteudo conteudo"
                    "lado_esq conteudo conteudo conteudo conteudo";
```

Isso irá criar as divisões dentro do container onde posteriormente iremos distribuir o conteúdo com outra propriedade.

## Propriedades para os itens

### Grid área

É uma abreviação de quatro propriedades (`grid-row-start`, `grid-column-start`, `grid-row-end`, `grid-column-end`.) e serve para indicar qual bloco de grid o conteúdo irá ocupar.

Exemplo:

```
grid-area: lado_esq;
```

O item ocupará o espaço designado para o `lado_esq`.

