

Curso: Front-End do Bem



AULA 10

- O que é CSS Grid?
- Propriedades do CSS Grid
- Exemplos de CSS Grid
- Sugestões de Prática para essa aula
 - Links das 9 aulas anteriores
- O que vamos aprender na aula 11



O que é CSS Grid?

Assim como o CSS FlexBox visto na aula passada, o CSS Grid também é uma solução de layout que nos permite alinhar e distribuir itens dentro de um container. A principal diferença entre os 2 é que:

O CSS FlexBox é unidimensional (Atuando no Eixo X);

O CSS Grid é bidimensional (Atuando tanto no Eixo X quanto no Eixo Y)

E através dessa junção de linhas e colunas tanto do Eixo X quanto no Eixo Y vai formar literalmente uma grade (por isso chamamos de Grid).

Resumindo:

O CSS Grid cria containers e dentro desses containers são criados itens em forma de grade, o que nos permite:

- definir onde esse itens irão ficar;
- informar qual será o seu tamanho;
- e qual será o valor do espaçamento entre eles

Ahhhhh já ia me esquecendo: Assim como o FlexBox o CSS Grid também se ajusta em diferentes tamanho de tela (Lembram do Responsivo que eu mencionei na aula passada, com o intuito de plantar a sementinha em Vossas cabeças, olha ele aqui novamente).



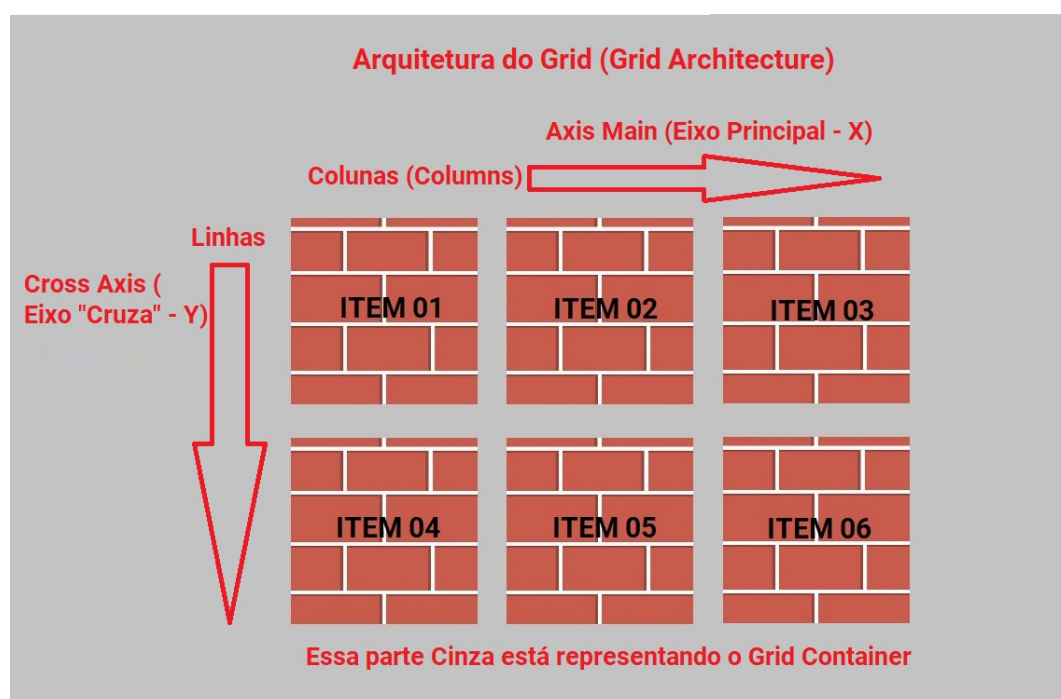
O que é CSS Grid?

A partir de agora vamos entender sobre as propriedades do CSS Grid:

- Aquelas que usamos nos containers;
- Aquelas que usamos dentro dos itens que estão dentro desses containers

Se fizermos aquela analogia marota para a nossa casa, a Arquitetura ficaria assim:

- A parede (parte cinza) seria o container;
- E os revestimentos (tijolinho) os itens dentro desse grid container(parede).



Vamos entender isso melhor com os conceitos dessas propriedades e exemplos já na próxima página dessa aula.

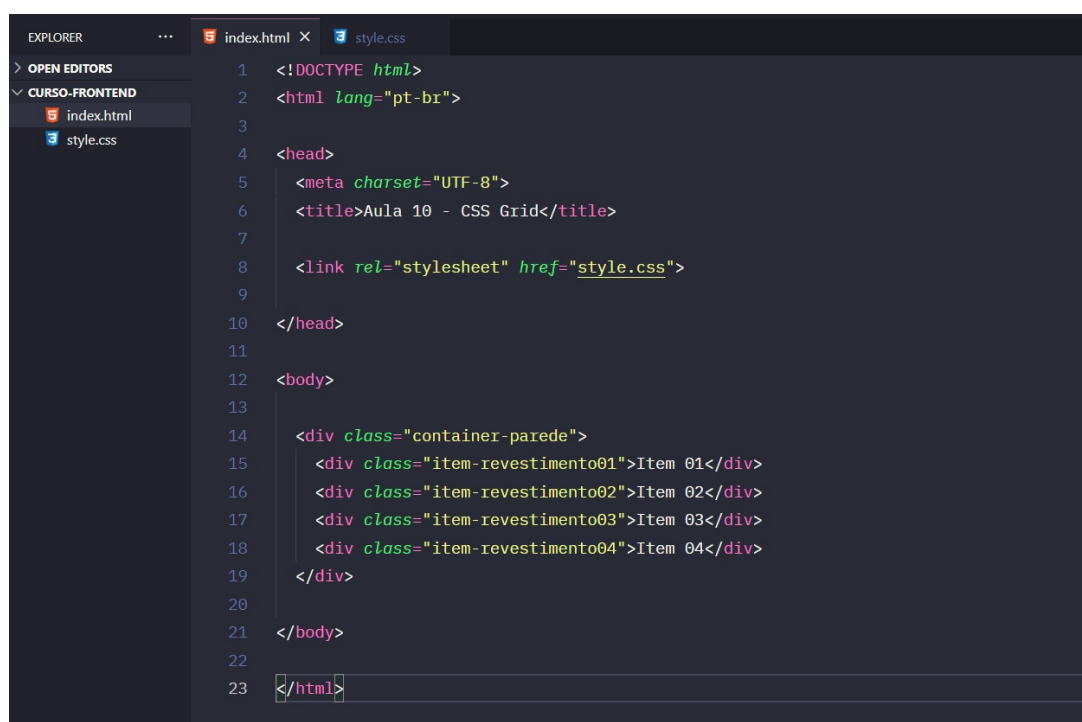
Partiu...

Propriedades do CSS Grid

Para começarmos a utilizar o CSS Grid vamos precisar de um container e o elemento que vamos usar será uma `<div>` `</div>` para representar esse container. E nele vamos utilizar a propriedade `display` como `grid`. Conforme estrutura abaixo:

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

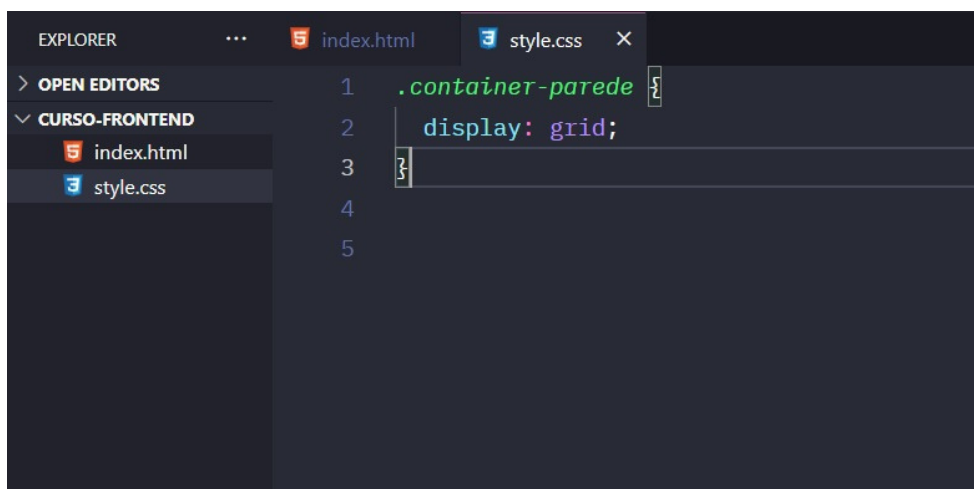
Vamos ver abaixo como ficou a estrutura do arquivo `index.html` a qual iremos usar para entender como o código é renderizado no navegador.

A screenshot of a code editor with a dark theme. The Explorer sidebar on the left shows a project named 'CURSO-FRONTEND' with files 'index.html' and 'style.css'. The main editor area displays the content of 'index.html' with line numbers from 1 to 23. The HTML code includes a DOCTYPE declaration, a lang attribute set to 'pt-br', a head section with meta charset, title 'Aula 10 - CSS Grid', and a link to 'style.css'. The body contains a 'container-parede' div which holds four 'item-revestimento' divs labeled 'Item 01' through 'Item 04'.

```
1 <!DOCTYPE html>  
2 <html lang="pt-br">  
3  
4 <head>  
5   <meta charset="UTF-8">  
6   <title>Aula 10 - CSS Grid</title>  
7  
8   <link rel="stylesheet" href="style.css">  
9  
10 </head>  
11  
12 <body>  
13  
14   <div class="container-parede">  
15     <div class="item-revestimento01">Item 01</div>  
16     <div class="item-revestimento02">Item 02</div>  
17     <div class="item-revestimento03">Item 03</div>  
18     <div class="item-revestimento04">Item 04</div>  
19   </div>  
20  
21 </body>  
22  
23 </html>
```

Estrutura definida no arquivo `index.html` vamos ver como informamos ao arquivo `style.css` que queremos utilizar o `display: grid`.

Propriedades do CSS Grid



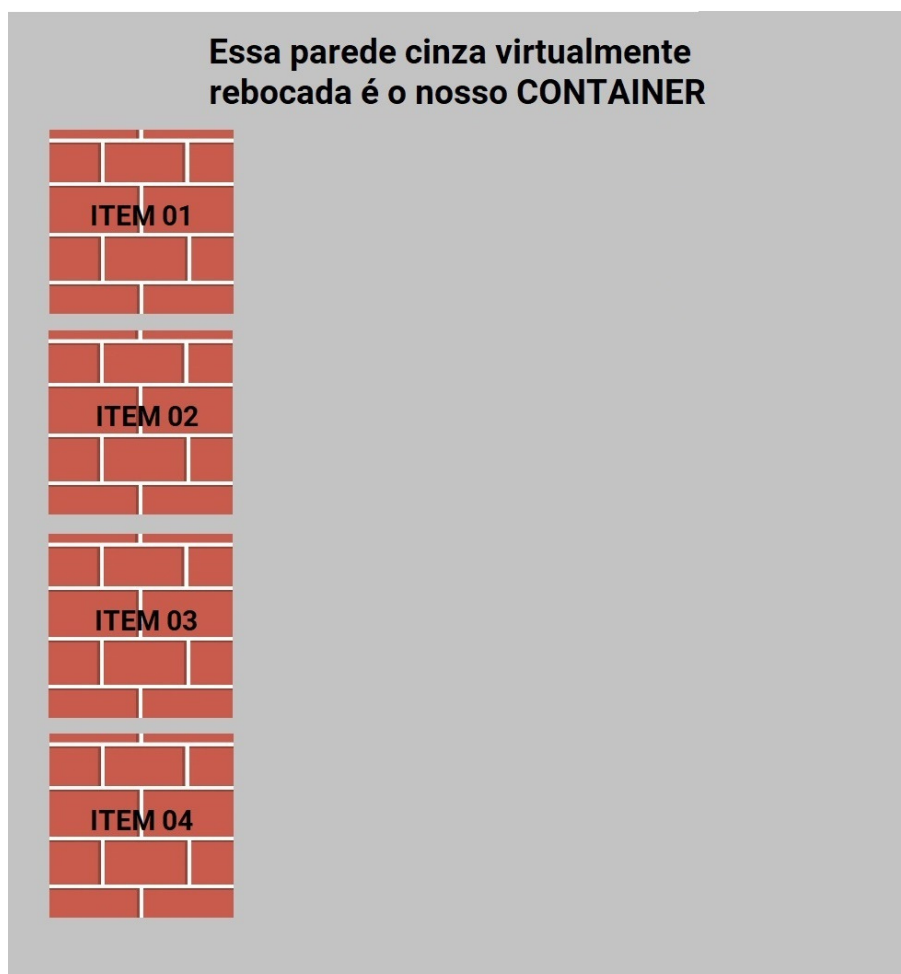
E bora ver o resultado disso renderizado no navegador:



Eita, Igor... esse tal de `display: grid` é um "brincalhão" não fez nada? Olhando "de primeira", realmente parece que ficou como se tivéssemos definido um `display` como `block`, já que os itens ficaram alinhados um embaixo do outro e sem nenhuma coluna definida.

Vamos ver o mesmo exemplo, só que agora fazendo analogia com a parede da nossa casa:

Propriedades do CSS Grid



Assim como no exemplo da página anterior os itens-revestimento (tijolinhos) ao inserirmos o display como grid (ficaram alinhados) um embaixo do outro como se nada tivesse acontecido. Mas nos bastidores ao informarmos para o nosso arquivo style que o display é grid... Sinalizamos mais ou menos assim: "style.css já foi definido o display como grid; agora descansa um pouquinho aí e aguarde o próximo comando para que as coisas comecem a fazer sentido e sejam mostradas na tela.

E a primeira propriedade que iremos ver é o grid-template-columns.

Propriedades do CSS Grid

Grid-Template-Columns:

Essa é a propriedade dentro do CSS Grid que vai definir o número de colunas as quais iremos trabalhar.

Para estruturarmos (montarmos) o nosso grid (grade), temos que definir antes qual será o formato que vamos utilizar.

Por exemplo: 4 columns X 2 rows (4 colunas x 2 linhas) – que é o que vamos utilizar nesse exemplo. Formato definido de 4 colunas e 2 linhas precisamos usar a propriedade `grid-template-columns` para que assim possamos definir a quantidade de colunas e a largura de cada uma delas.

Essa é a estrutura base para definirmos o `display` como `grid` e que vamos usar o `grid-template-columns`:

```
.container {  
  display: grid;  
  grid-template-columns: "aqui definimos o número de colunas"  
}
```

Agora que sabemos como funciona a estrutura base do código que representa o `grid-template-columns` (o qual digitaremos dentro do arquivo **style.css**)... bora ver como vai ficar a configuração com valores dentro dele.



Propriedades do CSS Grid

Grid-Template-Columns:

Para o nosso Grid Container (aqui representado por container-parede, vamos definir alguns valores para melhor entendimento do exemplo:

```
.container-parede {  
  max-width: 1200px;  
  (aqui definimos uma largura máxima de 1200px do nosso container)  
  display: grid;  
  (aqui habilitamos o uso do display grid)  
  color: white;  
  (configuramos a cor da fonte como branca)  
  gap: 10px;  
  (definimos um gap de 10px - lembra do cuidado com o "vão" entre o trem  
  e a plataforma, pois bem aqui estamos definindo esse vão o que será  
  melhor visualizado quando renderizarmos o código no navegador)  
  background-color: grey;  
  (Configuramos a cor do fundo da nossa parede como cinza, o que vai  
  facilitar o entendimento do gap de 10px configurado acima).  
  grid-template-columns: 150px 250px 350px 450px;  
  (Aqui definimos 4 colunas e cada coluna recebeu um valor manual)  
  Coluna 1: 150px  
  Coluna 2: 250px  
  Coluna 3: 350px;  
  Coluna 4: 450px  
}
```



/igor-rebolla

Propriedades do CSS Grid

Grid-Template-Columns:

Dentro do arquivo index.html (criamos 4 div's) com 4 classes diferentes, o que nos permite trabalhar cada uma com exclusividade:

```
.item-revestimento01  
.item-revestimento02  
.item-revestimento03  
.item-revestimento04
```

Só que nesse caso específico eu gostaria que mesmo essas div's tendo classes diferentes fossem definidos os mesmos valores para elas.

Dentro do nosso arquivo style.css (Só colocar essas classes separadas por vírgulas). Bora ver como isso fica:

```
.item-revestimento01, .item-revestimento02, .item-revestimento03,  
.item-revestimento04 {  
  background-color: orange;  
  display: grid;  
  justify-items: center;  
  align-items: center;  
}
```

Resumindo: as 4 classes vão receber uma cor de fundo (laranja), habilitamos o display grid aqui também, e os itens (tijolinhos) serão tanto alinhados quanto justificados ao centro.



/igor-rebolla

Propriedades do CSS Grid

Grid-Template-Columns:

Vamos ver como ficou esse código dentro do nosso arquivo style.css

```
EXPLORER  ...  index.html  style.css  X
> OPEN EDITORS
CURSO-FRONTEND
  index.html
  style.css
1
2  .container-parede {
3    max-width: 1200px;
4    display: grid;
5    color: white;
6    gap: 10px;
7    background-color: grey;
8    grid-template-columns: 150px 250px 350px 450px;
9  }
10
11  .item-revestimento01, .item-revestimento02, .item-revestimento03, .item-revestimento04 {
12    background-color: orange;
13    display: grid;
14    justify-items: center;
15    align-items: center;
16  }
17
18
```

Vamos ver como esse código ficou renderizado no navegador:



- 1 - Representa nossa primeira coluna a qual foi definida com uma largura de 150px
- 2 - Representa nossa segunda coluna a qual foi definida com uma largura de 250px
- 3 - Representa nossa terceira coluna a qual foi definida com uma largura de 350px
- 4 - Representa nossa quarta coluna a qual foi definida com uma largura de 450px

A somatoria das 4 colunas nesse exemplo aqui... totalizam: 1200px

Entre as colunas notem um vão em cinza (esse é o gap de 10px que definimos no .container-parede)

Propriedades do CSS Grid

Grid-Template-Columns:

Tá Igor, e tem como esses valores das 4 colunas serem definidos de forma automática e se eu posso definir 3 valores manuais e um de forma automática?

Tem sim é e exatamente isso que veremos logo abaixo.

Vamos manter a estrutura do nosso arquivo index.html e focar aqui só no style.css.

Para definirmos as nossas colunas todas como automáticas ou seja com larguras iguais. Vamos definir o valor do grid-template-columns, como auto.

```
.container-parede {  
  max-width: 1200px;  
  display: grid;  
  color: white;  
  gap: 10px;  
  background-color: grey;  
  grid-template-columns: auto auto auto auto;  
}
```

E bora ver na imagem da próxima página como esse código ficou dentro do style.css:



Propriedades do CSS Grid

Grid-Template-Columns:

```
.container-parede {  
  max-width: 1200px;  
  display: grid;  
  color: white;  
  gap: 10px;  
  background-color: grey;  
  grid-template-columns: 200px 50px 300px auto;  
}
```

(Aqui definimos 4 colunas e cada coluna recebeu um valor)

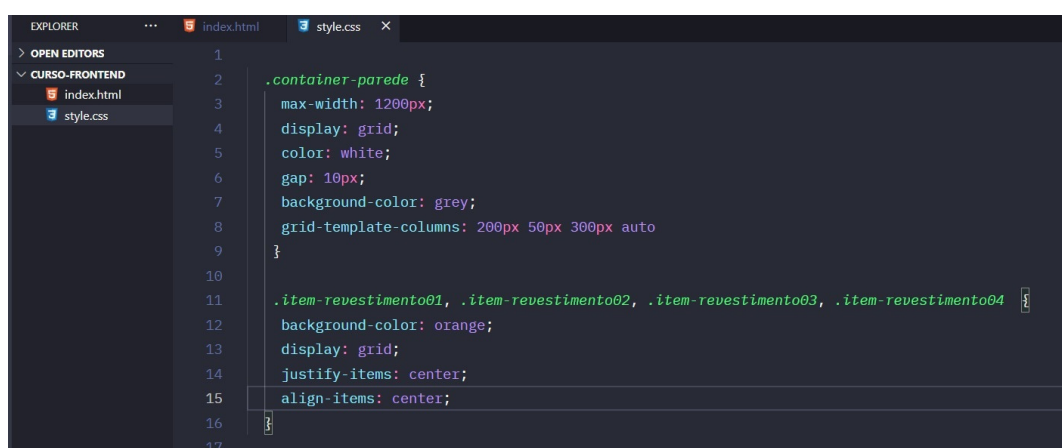
Coluna 1: 200px

Coluna 2: 50px

Coluna 3: 300px

Coluna 4: auto

E bora ver como esse código ficou no style.css:



Propriedades do CSS Grid

Grid-Template-Columns:

E bora ver como esse código será renderizado no navegador:



- 1 - A primeira coluna recebeu o valor de 200px
- 2 - A segunda coluna recebeu o valor de 50px
- 3 - A terceira colina recebeu o valor de 300px
- 4 - A quarta coluna recebeu o valor de auto

OBS.: Como sugestão eu peço para vocês diminuírem a tela do navegador a qual foi renderizada essa imagem da direita pra a esquerda e vejam que as colunas (1, 2 e 3) não sofrem alterações pois o valor foi definido manualmente já a coluna 4 que recebeu o valor auto vai diminuir conforme sua disposição no container.

Direção da setinha (esquerda para a direita) para ver o comportamento da quarta coluna.



Propriedades do CSS Grid

Grid-Template-Rows:

Essa é a propriedade dentro do CSS Grid que vai definir (o número de linhas e sua altura) no Grid as quais iremos trabalhar.

Assim como no grid-template-columns, o grid-template-rows também tem sua configuração base

Essa é a estrutura base para definirmos o display como grid e que vamos usar o grid-template-rows:

```
.container {  
  display: grid;  
  grid-template-rows: "aqui definimos o número de linhas"  
}
```

r

Agora que sabemos como funciona a estrutura base do código que representa o grid-template-rows (o qual digitaremos dentro do arquivo style.css)... bora ver como ficará a configuração com 2 valores (um valor de 125px para a primeira linha(row) e outro de 275px para a segunda linha) a terceira e quarta linha(row) permanecem com seus valores padrões (nesse exemplo).

```
.container-parede {  
  max-width: 1200px;  
  display: grid;  
  color: white;  
  gap: 10px;  
  background-color: grey;  
  grid-template-rows: 125px 275px;  
}
```

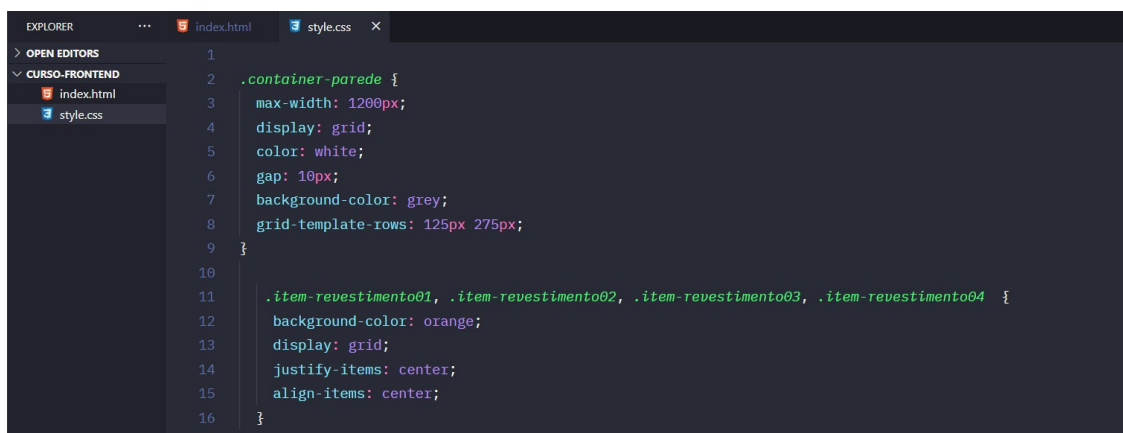


/igor-rebolla

Propriedades do CSS Grid

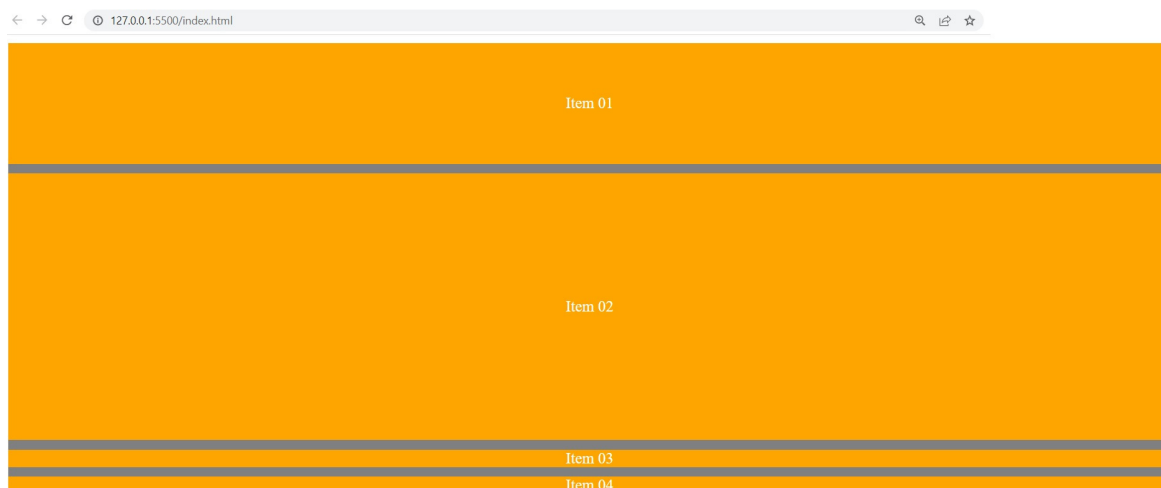
Grid-Template-Rows:

E bora ver como esse código ficou no style.css:



```
1
2 .container-parede {
3   max-width: 1200px;
4   display: grid;
5   color: white;
6   gap: 10px;
7   background-color: grey;
8   grid-template-rows: 125px 275px;
9 }
10
11 .item-revestimento01, .item-revestimento02, .item-revestimento03, .item-revestimento04 {
12   background-color: orange;
13   display: grid;
14   justify-items: center;
15   align-items: center;
16 }
```

E bora ver como esse código ficou renderizado no navegador:



- A linha correspondente ao Item 01 está com uma altura de 125px
- A linha correspondente ao Item 02 está com uma altura de 275px
- As linhas correspondentes aos itens 03 e 04 estão com sua altura padrão pois não definimos nenhum valor para elas.

Propriedades do CSS Grid

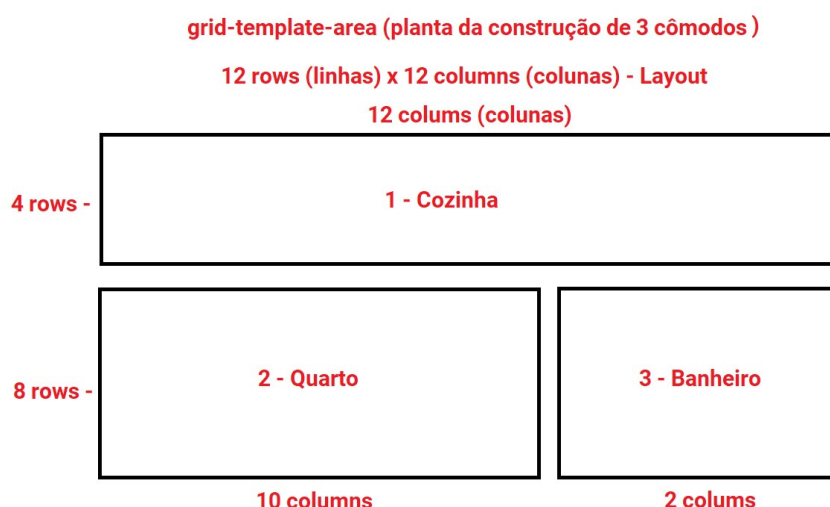
Grid-Template-Areas:

O `grid-template-areas` é usado para especificar a quantidade de espaço que uma célula do grid (grade) deve conter em termos de colunas e linhas no container pai.

Para melhor entendimento vamos criar o seguinte layout abaixo tomando como base o exemplo da construção de 3 cômodos em nosso terreno.

- Vamos definir o nosso terreno com uma largura máxima de 800px
- E os cômodos serão: (Cozinha, Quarto, Banheiro).

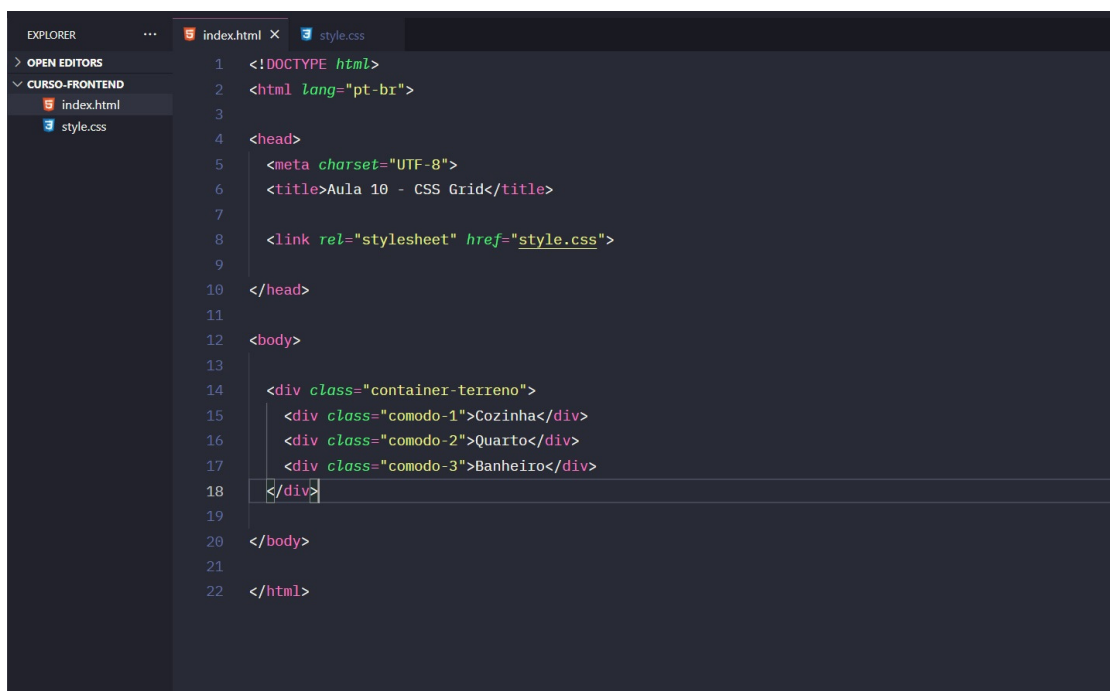
Com base nessas informações (nossa planta/layout) ficou assim:



Como temos 3 cômodos, vamos definir essa estrutura no nosso arquivo `index.html`.

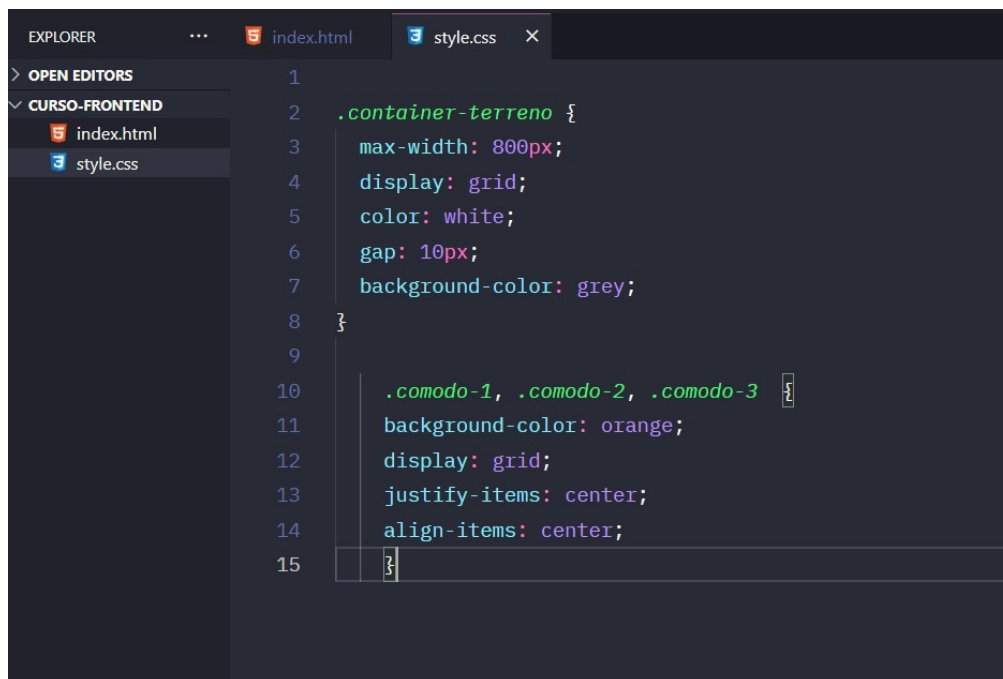
Propriedades do CSS Grid

Grid-Template-Areas:



```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>Aula 10 - CSS Grid</title>
7
8   <link rel="stylesheet" href="style.css">
9
10 </head>
11
12 <body>
13
14   <div class="container-terreno">
15     <div class="comodo-1">Cozinha</div>
16     <div class="comodo-2">Quarto</div>
17     <div class="comodo-3">Banheiro</div>
18   </div>
19
20 </body>
21
22 </html>
```

Com a estrutura definida no arquivo index.html, vamos definir agora as propriedades do nosso (layout/planta) dentro do arquivo style.css



```
1
2 .container-terreno {
3   max-width: 800px;
4   display: grid;
5   color: white;
6   gap: 10px;
7   background-color: grey;
8 }
9
10 .comodo-1, .comodo-2, .comodo-3 {
11   background-color: orange;
12   display: grid;
13   justify-items: center;
14   align-items: center;
15 }
```

Propriedades do CSS Grid

Grid-Template-Areas:

Vamos ver como o código da página anterior ficou renderizado no navegador:



Agora bora utilizar o grid-template-areas. O código dentro do style.css ficará assim:

```
EXPLORER
  OPEN EDITORS
    CURSOR-FRONTEND
      index.html
      style.css
  style.css
1
2 .container-terreno {
3   max-width: 800px;
4   height: 100vh;
5   display: grid;
6   color: white;
7   gap: 10px;
8   background-color: grey;
9   grid-template-areas:
10    "Cozinha Cozinha Cozinha Cozinha  Cozinha Cozinha Cozinha Cozinha  Cozinha Cozinha Cozinha Cozinha"
11    "Quarto  Quarto  Quarto  Quarto   Quarto  Quarto  Quarto  Quarto   Quarto  Quarto  Banheiro Banheiro"
12    "Quarto  Quarto  Quarto  Quarto   Quarto  Quarto  Quarto  Quarto   Quarto  Quarto  Banheiro Banheiro";
13 }
14
15 .comodo-1, .comodo-2, .comodo-3 {
16   background-color: orange;
17   display: grid;
18   justify-items: center;
19   align-items: center;
20   height: 100px;
21 }
22
23 .comodo-1 {
24   grid-area: Cozinha;
25 }
26
27 .comodo-2 {
28   grid-area: Quarto;
29 }
30
31 .comodo-3 {
32   grid-area: Banheiro;
```

Antes da visualização do código acima no navegador, vamos entender o que esse montão de Cozinha, Quarto e Banheiro estão fazendo ai.

Propriedades do CSS Grid

Grid-Template-Areas:

No nosso layout/planta da página 17 dessa aula (ficou definido) que nossa Cozinha seria composta por 12 columns x 4 rows. Cada Cozinha que digitamos dentro do grid-template-areas abaixo, representa 1 columns, por isso digitamos 12x a palavra Cozinha e como queremos que essa cozinha tenha 1 altura de 4 linhas (só precisamos digitar 1x essa linha para representar essas 4 linhas).

grid-template-areas:

```
"Cozinha Cozinha Cozinha Cozinha Cozinha Cozinha Cozinha Cozinha  
Cozinha Cozinha Cozinha Cozinha"
```

E para o Quarto e o Banheiro a mesma coisa. Definimos que o quarto seria composto por 10 colunas e 8 linhas e o Banheiro de 2 colunas e 8 linhas. É exatamente isso que o código abaixo está representando 10x a palavra Quarto escrita (o que representa as 10 colunas) e 2x a palavra Banheiro escrita (o que representa as 2 colunas). Notem que aqui a linha foi repetida 2x ou seja (Digitamos 20x a palavra Quarto e 4x a palavra Banheiro), isso significa que teremos 8 linhas.

```
"Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto  
Quarto Banheiro Banheiro"
```

```
"Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto Quarto  
Quarto Banheiro Banheiro";
```



/igor-rebolla

Propriedades do CSS Grid

Grid-Template-Areas:

```
.comodo-1{  
  grid-area: Cozinha;  
}
```

```
.comodo-2{  
  grid-area: Quarto;  
}
```

```
.comodo-3{  
  grid-area: Banheiro;  
}
```

E aqui para cada cômodo especificado, temos que usar o grid-area com o nome do respectivo cômodo para fazer uma espécie de assinatura/verificação no grid-template-areas.

Se um quarto cômodo (por exemplo: lavanderia) fosse adicionado na nossa planta/layout, bastaria especificar esse cômodo dentro do grid-template-areas e assim feito criar um novo grid-area: lavanderia e fazer sua assinatura conforme exemplo abaixo:

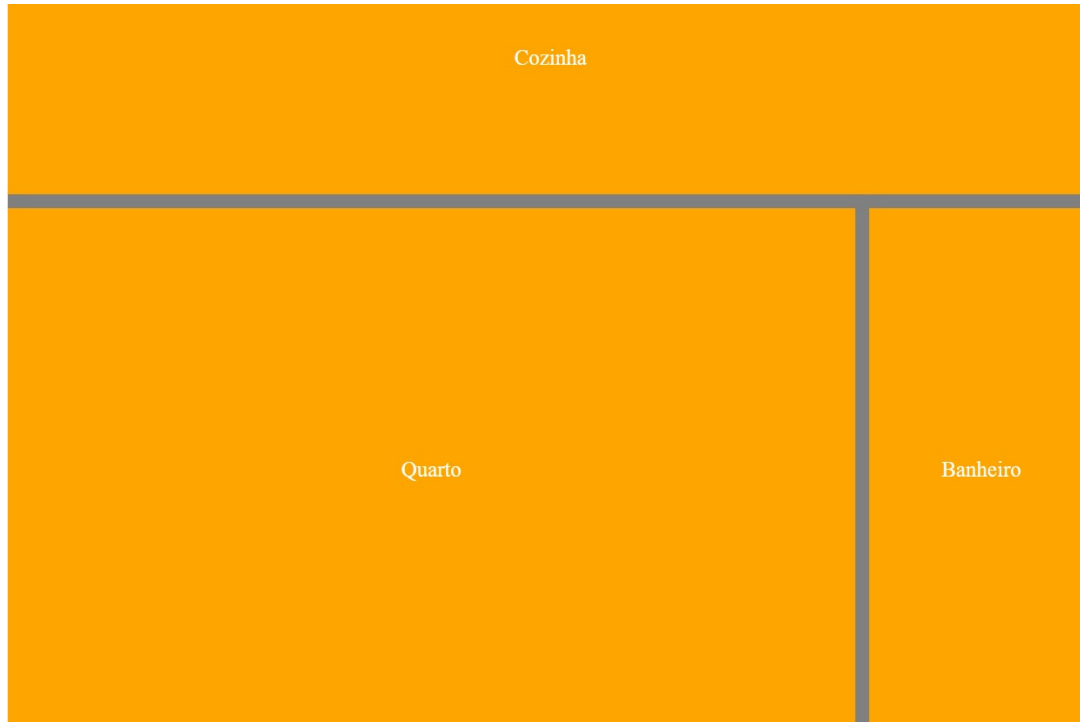
```
.comodo-4{  
  grid-area: Quintal;  
}
```

Agora sim podemos ver o código renderizado no navegador:

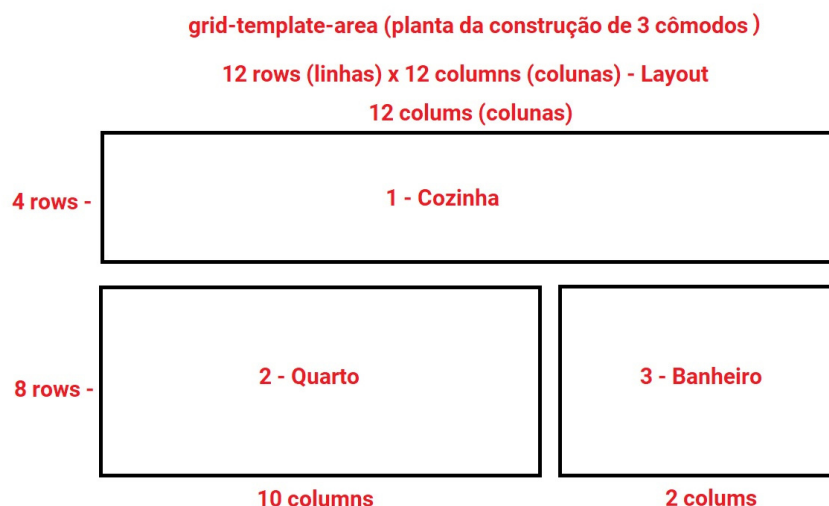


Propriedades do CSS Grid

Grid-Template-Areas:



Acima podemos ver o resultado final do nosso layout/planta (3 Cômodos). E abaixo o layout/planta que foi definido na página 17 dessa aula.



Links das 9 aulas anteriores

Para acessar cada aula, basta clicar na aula de interesse abaixo que será direcionado diretamente para o post com a respectiva aula no LinkedIn:

[Aula 01](#)

[Aula 02](#)

[Aula 03](#)

[Aula 04](#)

[Aula 05](#)

[Aula 06](#)

[Aula 07](#)

[Aula 08](#)

[Aula 09](#)



Sugestões de prática para essa aula:

1. Abrir o Microsoft Visual Studio Code, depois a pasta Curso-FrontEnd e os arquivos: index.html e style.css;
2. Refaça os exemplos dessa aula;
3. Exemplos feitos (Conforme item 2) bora criar o seu próprio layout/planta - primeiro com 3 cômodos e depois com 4 cômodos.
4. Dentro da imagem da página 19 dessa aula (tem uma propriedade height com um valor definido de 100vh. Pesquisar o que esse VH faz no CSS.

Programação é prática, curiosidade e repetição!!!!



O que vamos aprender na aula 11:

Na décima primeira e próxima aula, continuaremos nossos estudos sobre o Segundo Pilar do Curso - O CSS 3

Feedbacks dessa décima aula são bem-vindos.

Nos vemos na décima primeira aula que será disponibilizada no dia 10/02/2022 às 8h30 (Horário de Brasília) - Dentro do meu perfil no LinkedIn.

