ntander Coders 2024.1 | Front-End | #1178

rid Layout

Isaias Soares
Id: 1178008

Antes de começarmos a falar sobre *Grid Layout*, se faz necessário entender o conceito de *grid* (grades).

Conforme encontramos na documentação do MDN, *grid* é definido por:

> *Grid* é uma malha formada pela interseção de um conjunto de linhas horizontais e um conjunto de linhas verticais – um dos conjuntos define colunas e outro linhas.

Levando isso em consideração, o *grid layout* nos permite dispor os elementos na página de maneira que seria mais complexo implementar utilizando outras técnicas.

Vamos ver isso na prática:

Para aplicarmos o *grid layout*, precisamos definir um "*grid container*". Uma elemento no qual estarão dispostos os elementos que queremos disponibilizar em forma de grade.

Criação do grid container

```
/* CSS */
.container {
```

```
  display: grid;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}

<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

Imagem 1 - `display: grid` *(Fonte da imagem: do autor)*

Até esse momento, não conseguimos vislumbrar a aplicação da *grid* na tela. Para começarmos a dispor os elementos em grade, se faz necessário adicionar *column tracks*, espaços entre as linhas em um *grid*.

Definindo um column track para o container:

```css
/* CSS */
.container {
  display: grid;
  grid-template-columns: 250px 250px 250px 250px;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}
```

```html
<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

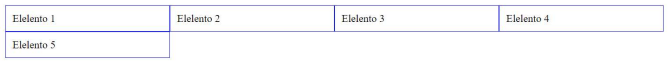

Imagem 2 - *`display: grid`, definindo largura fixa para as células (Fonte da imagem: do autor)*

Note que, ao definirmos o `grid-template-columns` com valor de `250px 250px 250px 250px`, são criadas quatro colunas fixas com largura de 250 pixels e os elementos criados dentro do container são dispostos, um em cada célula criada.

Até então, podemos definir várias colunas em nossa tela, mas todas elas possuem uma largura fixa. Pensando em construir células com layout flexível, podemos utilizar a unidade `fr`, que representa uma fração do espaço disponível no container.

Aplicando `fr` ao exemplo anterior, mantendo as células com tamanho igual entre si:

```
/* CSS */
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}

<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

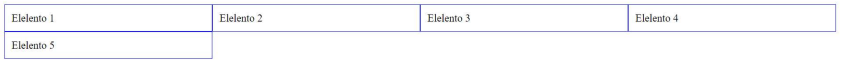

Imagem 3 - *`display: grid`, definindo largura dinâmica (Fonte da imagem: do autor)*

Aplicando `fr` ao exemplo anterior, mantendo as células com tamanhos diferentes entre si:

```
/* CSS */
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}

<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
```

```
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

Elelento 1	Elelento 2	Elelento 3
Elelento 4	Elelento 5	

Imagem 4 - *`display: grid`, definindo largura dinâmica e proporções (Fonte da imagem: do autor)*

Com a utilização da unidade `fr`, conseguimos definir que a primeira célula terá proporcionalmente o dobro das demais. Dividimos o nosso layout em quatro partes, onde duas partes serão para a primeira coluna e uma parte para cada uma das demais.

Podemos, também, mesclar larguras fixas e dinâmicas em nosso layout.

```
/* CSS */
.container {
  display: grid;
  grid-template-columns: 350px 1fr 2fr;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}
```

```html
<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

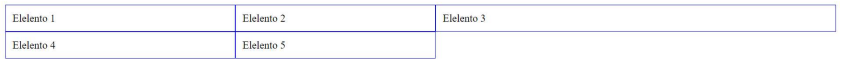

Imagem 5 - *`display: grid`, mesclando largura fixa e dinâmica (Fonte da imagem: do autor)*

Para o exemplo acima, definimos a largura fixa de 350px, dividindo o espaço restante em outras duas colunas de tamanho flexível.

Uma outra opção que podemos usar para definir o nosso layout é a utilização da notação `repeat()`. Usando um dos exemplos que vimos acima, onde definimos o `grid-template-columns` com valor de `1fr 1fr 1fr 1fr`, podemos obter o mesmo resultado aplicando ao `grid-template-columns` o valor de `repeat(4, 1fr)` e obtermos o mesmo resultado. Vejamos:

```css
/* CSS */
.container {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
```

```
}

.item {
  border: 1px solid blue;
  padding: 10px;
}

<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

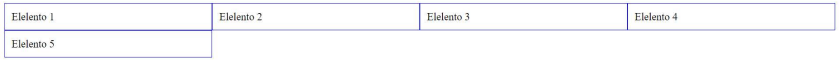

Imagem 6 - `display: grid`, *utilizando* `repeat()` *(Fonte da imagem: do autor)*

Até então, utlizamos o `grid-template-columns` para definir as colunas do nosso *grid*. As linhas são criadas implicitamente, de acordo com a quantidade de elemento a serem dispostos na tela.

Se quisermos definir um tamanho para as células criadas implicitamente, podemos utilizar as propriedades `grid-auto-rows` e `grid-auto-columns`, como veremos no exemplo abaixo:

```css
/* CSS */
.container {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-auto-rows: 100px;
}

.item {
  border: 1px solid blue;
  padding: 10px;
}
```

```html
<!-- HTML -->
<div class="container">
  <div class="item">Elelento 1</div>
  <div class="item">Elelento 2</div>
  <div class="item">Elelento 3</div>
  <div class="item">Elelento 4</div>
  <div class="item">Elelento 5</div>
</div>
```

Resultado:

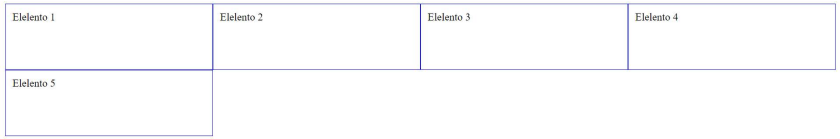

Imagem 7 - *`display: grid`, definindo altura para as células (Fonte da imagem: do autor)*

Há ainda muitas outras possibilidades a serem exploradas utilizando o *grid*. Para ver essas opções, recomendamos aprofundar na documentação.

Referências e Links úteis

- Conceitos básicos de Grid Layout. MDN Web Docs, 19/08/2021. Acesso em: 27/06/2022.
- Guia CSS Grid Layout. Origamid. Acesso em 27/06/2022.
- A Complete Guide to Grid. Chris House (CSS Tricks), 12/05/2021. Acesso em 27/06/2022
- Grid - Game.

Próximo Tópico