Trabalhando GRID CSS

Exemplo #1

HTML base do exemplo:

```
<!-- Exemplo de uso de grid -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="grid1.css">
    <title>Document</title>
</head>

<body>
    <div class="container-grid">
        <div class="div1">div1</div>
        <div>div2</div>
        <div class="div3">div3</div>
        <div>div4</div>
        <div>div5</div>
        <div>div6</div>
        <div>div7</div>
        <div>div8</div>
        <div>div9</div>
        <div>div10</div>
        <div>div11</div>

    </div>

</body>

</html>
```

CSS associado ao HTML base do exemplo:

```
.container-grid{
   display: grid;
   grid-template-columns: auto auto auto;
   background-color: orange;
   padding: 10px;
   gap:5px
}
.container-grid > div{
   background-color: aqua;
   border: 1px solid blue;
   font-size: 30px;
   text-align: center;
   padding:    10px;
}
```

Explicação do CSS

1. **`.container-grid{` :** Define uma regra de estilo para elementos com a classe `.container-grid`.
2. **`display: grid;` :** Define o tipo de **layout como grid** para o elemento com a classe `.container-grid`.
3. **`grid-template-columns:** auto auto auto;` : Define o número e a largura das colunas na grade. Neste caso, estamos usando a palavra-chave `auto` para que as colunas sejam dimensionadas automaticamente para se ajustarem ao conteúdo dentro delas. **Isso cria três colunas de largura igual**.
4. **`background-color: orange;` :** Define a cor de fundo do elemento com a classe `.container-grid` como laranja.
5. **`padding: 10px;` :** Adiciona um preenchimento interno de 10 pixels em todas as direções para o elemento com a classe `.container-grid`. Isso cria um espaço entre as bordas do container e seu conteúdo.
6. **`gap: 5px;` :** Define o espaçamento entre as células do grid, tanto na direção das linhas quanto na direção das colunas. Aqui, estamos definindo um espaçamento de 5 pixels entre as células.
7. **`.container-grid > div{` :** Define uma regra de estilo para os elementos `div` que são filhos diretos do elemento com a classe `.container-grid`.
8. **`background-color: aqua;` :** Define a cor de fundo dos elementos `div` como aqua.
9. **`border: 1px solid blue;` :** Adiciona uma borda de 1 pixel sólida azul ao redor dos elementos `div`.
10. **`font-size: 30px;` :** Define o tamanho da fonte dos elementos `div` como 30 pixels.
11. **`text-align: center;` :** Centraliza o texto dentro dos elementos `div`.
12. **`padding: 10px;` :** Adiciona um preenchimento interno de 10 pixels em todas as direções para os elementos `div`. Isso cria um espaço entre o conteúdo dos elementos e suas bordas.

Com base nessas regras de estilo CSS, o resultado produzido será:

- um container com layout de grid, composto por três colunas de largura igual.
- O fundo do container será laranja e haverá um preenchimento interno de 10 pixels em todas as direções.
- haverá um espaçamento de 5 pixels entre as células do grid.
- Dentro desse container, haverá vários elementos ***div*** com
  - uma cor de fundo *aqua*,
  - uma borda sólida azul de 1 pixel,
  - um tamanho de fonte de 30 pixels e
  - o texto centralizado.

- Cada elemento ***div*** terá um preenchimento interno de 10 pixels em todas as direções, criando um espaço entre seu conteúdo e sua borda.

Resultado renderizado:

Exemplo #2

HTML base do exemplo:

```
<!-- Exemplo de uso de grid -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="grid2.css">
    <title>Document</title>
</head>

<body>
    <div class="container">
        <div id="box-1" class="box">x1</div>
        <div id="box-2" class="box">x2</div>
        <div id="box-3" class="box">x3</div>
        <div id="box-4" class="box">x4</div>
        <div id="box-5" class="box">x5</div>
        <div id="box-6" class="box">x6</div>
    </div>

</body>

</html>
```

CSS associado ao HTML base do exemplo:

```
.container {
   border: 1px solid black;
   width: 100%;
   /*largura*/
   display: grid;
   grid-template-columns: 20% 60% 20%;
   grid-template-rows: 200px 200px;
   gap: 10px;
}

.box {
   border: 1px solid blue;
   background-color: tomato;

}
#box-1{
   grid-column: 1/4   ;
}
#box-2{
   grid-column: 1/3   ;
}
```

Explicação do CSS

Para `.container`:

1. `` `border: 1px solid black;` ``: Adiciona uma borda sólida preta de 1 pixel ao redor do elemento com a classe `.container`.
2. `` `width: 100%;` ``: Define a largura do elemento com a classe `.container` como 100% da largura do seu elemento pai.
3. `` `display: grid;` ``: Define o tipo de layout como grid para o elemento com a classe `.container`.
4. `` `grid-template-columns: 20% 60% 20%;` ``: Define o número e a largura das colunas na grade. Aqui, estamos definindo três colunas, com larguras de 20%, 60% e 20% da largura total do container, respectivamente.
5. `` `grid-template-rows: 200px 200px;` ``: Define o número e a altura das linhas na grade. Aqui, estamos definindo duas linhas, ambas com uma altura fixa de 200 pixels.
6. `` `gap: 10px;` ``: Define o espaçamento entre as células do grid, tanto na direção das linhas quanto na direção das colunas. Aqui, estamos definindo um espaçamento de 10 pixels.

Para **`.box` e `#box-1` e `#box-2`:**

1. `` `.box` ``: Define uma regra de estilo para elementos com a classe `.box`.
2. `` `border: 1px solid blue;` ``: Adiciona uma borda sólida azul de 1 pixel ao redor dos elementos com a classe `.box`.
3. `` `background-color: tomato;` ``: Define a cor de fundo dos elementos com a classe `.box` como vermelho-tomate.
4. `` `#box-1` e `#box-2` ``: Define uma regra de estilo para elementos com os IDs `box-1` e `box-2`, respectivamente.
5. `` `grid-column: 1/4;` `` e `` `grid-column: 1/3;` ``:(*) Define a localização das células dos elementos `#box-1` e `#box-2` no grid, respectivamente. Neste caso, `grid-column: 1/4;` indica que o elemento ocupará três colunas (da primeira à terceira), enquanto `grid-column: 1/3;` indica que o elemento ocupará duas colunas (da primeira à segunda).

(*)

A propriedade `grid-column` no formato `start / end` especifica as linhas de início e término de um elemento em um grid. No caso de `grid-column: 1/4`, o elemento começa na linha de início 1 e termina na linha de término 4.

Por que ocupa somete 3, e não 4? Veja a reposta:

- A linha de início (start) é a primeira coluna especificada, ou seja, a coluna número 1.

- A coluna 4 é a última coluna antes da coluna de término. Isso significa que o elemento ocupa três colunas no total. A coluna de término não é contada como parte do intervalo ocupado pelo elemento.

Com base nessas regras de estilo CSS, o resultado produzido será:

- um container com layout de grid, composto por três colunas de larguras relativas (20%, 60% e 20%) e duas linhas de altura fixa (200 pixels cada). Haverá um espaçamento de 10 pixels entre as células do grid.
- Dentro desse container, haverá vários elementos com a classe .box, cada um com uma borda sólida azul de 1 pixel e uma cor de fundo vermelho-tomate.
- O elemento com o ID box-1 ocupará três colunas, cobrindo toda a largura do container. Já o elemento com o ID box-2 ocupará duas colunas, cobrindo 80% da largura do container.

Resultado renderizado: