Linguagens de Marcação e Scripts

—

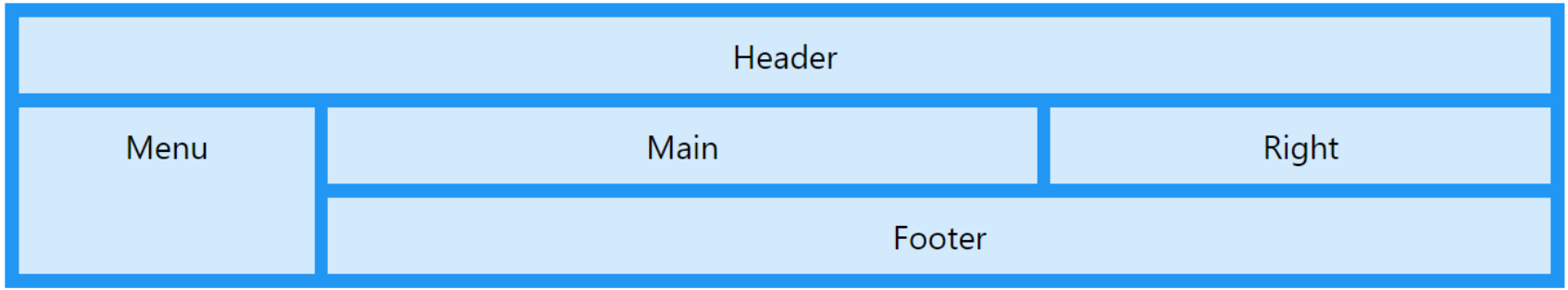
Prof. Aníbal Cavalcante de Oliveira
UFC - QXD0164 - 2019.2

Agenda - Aula 14

- CSS Grid Layout

CSS Grid Layout

CSS Grid layout é uma Especificação do W3C projetada para proporcionar um método bidimensional para criação de layout CSS que oferece a possibilidade de "layoutar" itens da página com uso de linhas e colunas.



O CSS Grid Layout oferece um sistema de layout baseado **em grade**, com **linhas e colunas**, facilitando o design de páginas da web sem a necessidade de usar elementos que flutuam, ou posicionamento absoluto e relativo.

CSS Grid Layout (exemploGridCss01)

CSS

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

HTML

```
<!DOCTYPE html>
<html>
<head>
    <link href="style.css">
</head>
<body>
    <div class="grid-container">
      <div class="item1">Header</div>
      <div class="item2">Menu</div>
      <div class="item3">Main</div>
      <div class="item4">Right</div>
      <div class="item5">Footer</div>
   </div>
</body>
</html>
```

Elemento Pai ou Contêiner (Grid)

Um layout de uma grid consiste de um elemento pai, com um ou mais elementos filhos.

O elemento pai deve possuir a propriedade de exibição (**display**) definida como "**grid**" ou "**inline-grid**" (aceita elementos na mesma linha).

Apenas os elementos filhos diretos do contêiner de grade se tornam automaticamente itens de grade.

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

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

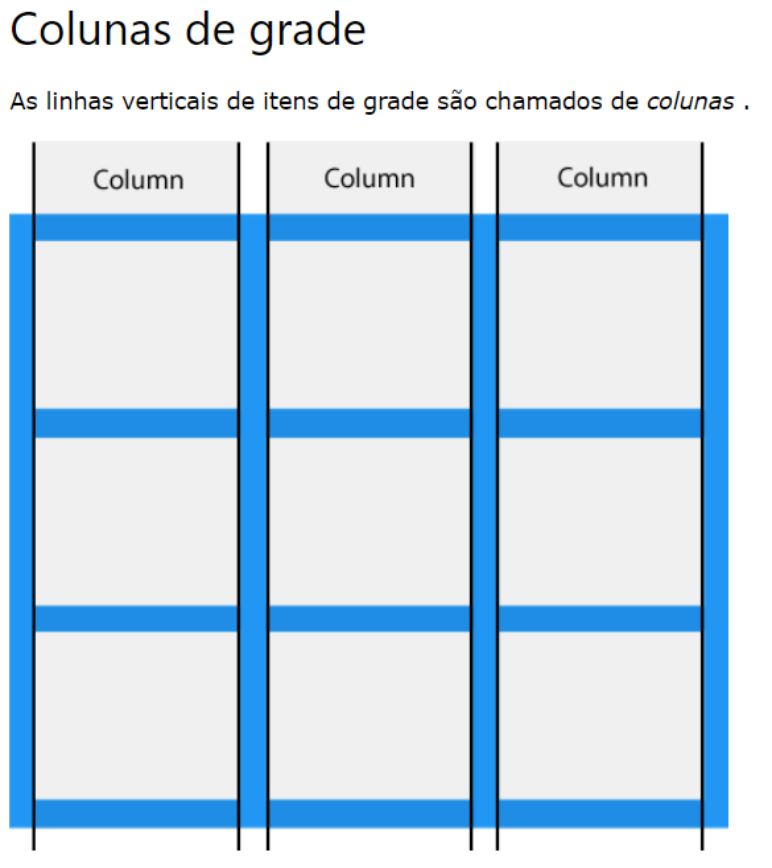
Elemento Pai ou Contêiner (Grid)

```
.grid-container {
  display: grid; /*inline-grid*/
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

1	2	3
4	5	6
7	8	9

Elemento Pai ou Contêiner (Grid) – Uma grid com colunas



Elemento Pai ou Contêiner (Grid) – Uma grid com colunas

A propriedade **grid-template-columns** define o número de colunas em seu layout de grade, e pode definir a largura de cada coluna.

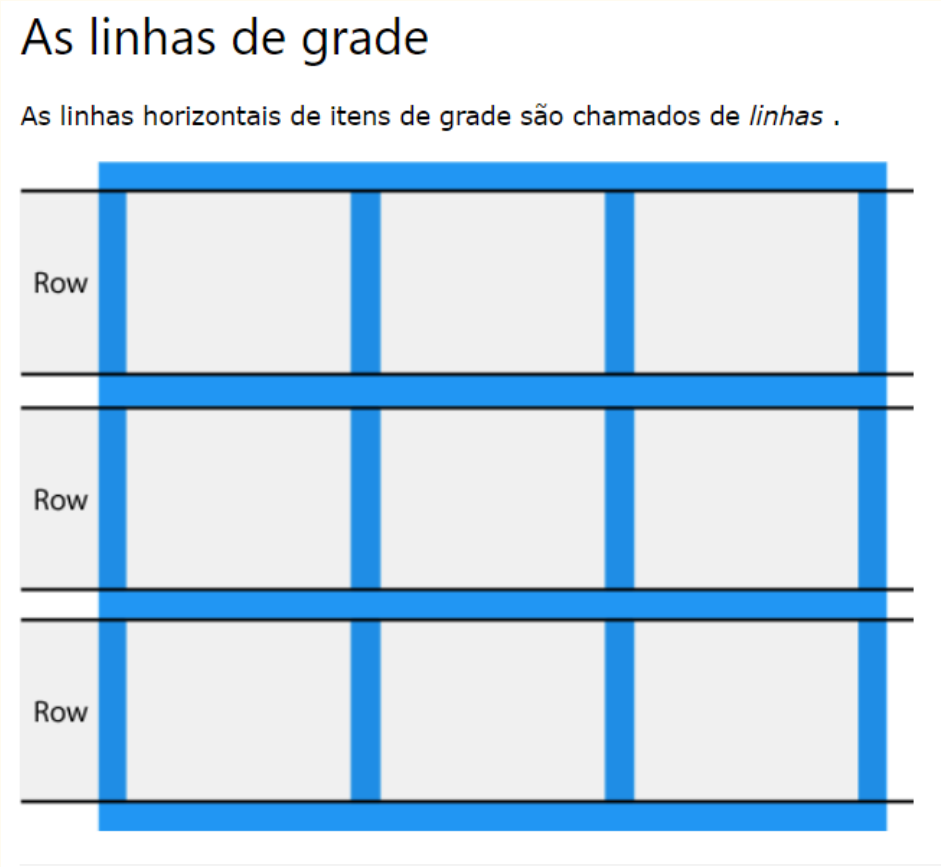
O valor é uma lista separada por espaços, em que cada valor define o comprimento da respectiva coluna.

Se você quiser que o seu layout de grade para conter 4 colunas, especifique a largura das 4 colunas, ou "**auto**" se todas as colunas devem ter a mesma largura.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
</div>
```

Elemento Pai ou Contêiner (Grid) – Uma grid de linhas



Elemento Pai ou Contêiner (Grid) – Uma grid com colunas

A propriedade **grid-template-rows** define a altura de cada linha.

O valor é uma lista separada por espaços, em que cada valor define a altura da respectiva fileira:

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-template-rows: 80px 200px;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

Elemento Pai ou Contêiner (Grid) – Uma grid com colunas

Existem ainda as propriedades:

1 - **justify-content**: usada para alinhar toda a grid no interior do container.

2 - **align-content**: usada para alinhar verticalmente toda a grid no interior do container.

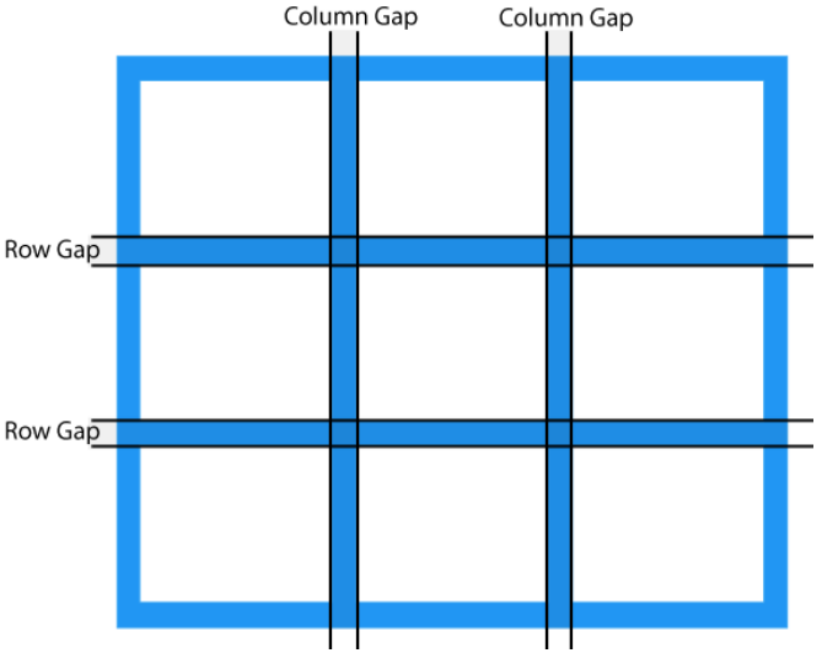
```
.grid-container {
  display: grid;
  justify-content: space-evenly;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```

```
.grid-container {
  display: grid;
  height: 400px;
  align-content: center;
  grid-template-columns: auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
```

* space-evenly | space-around | space-between | center | start | end

Elemento Pai ou Contêiner (Grid) – Espaços entre linhas e colunas

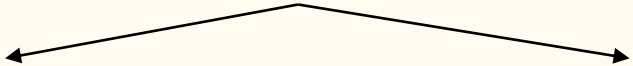
Os espaços entre cada coluna / linha são chamados de *lacunas* .



É possível ajustar o tamanho da folga, utilizando uma das seguintes propriedades:

`grid-column-gap`
`grid-row-gap`
`grid-gap`

Elemento Pai ou Contêiner (Grid) – Espaços entre linhas e colunas



grid-column-gap

```
.grid-container {
  display: grid;
  grid-column-gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

grid-row-gap

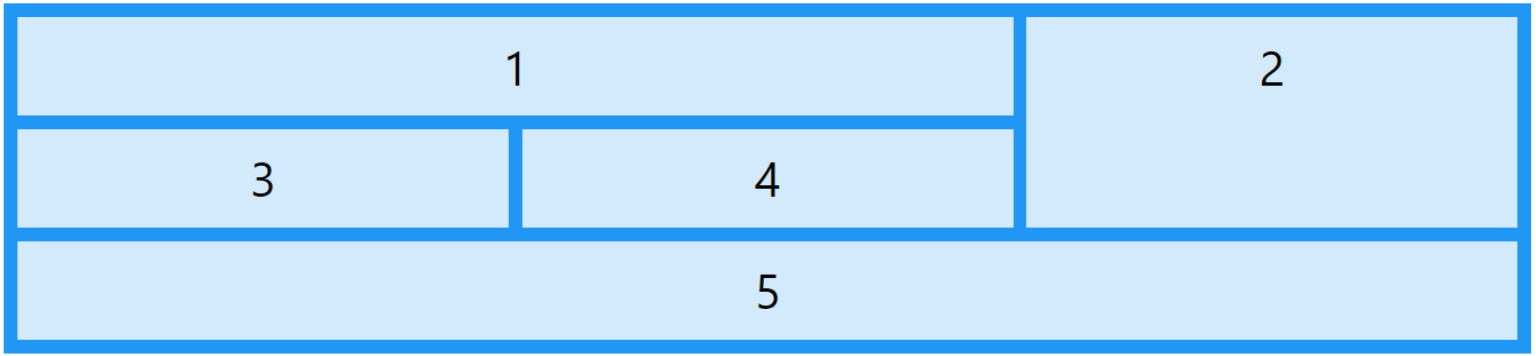
```
.grid-container {
  display: grid;
  grid-row-gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

* **grid-gap**: 50px 100px; /* a versão encurtada das duas propriedades*/

Elementos Filhos ou Itens

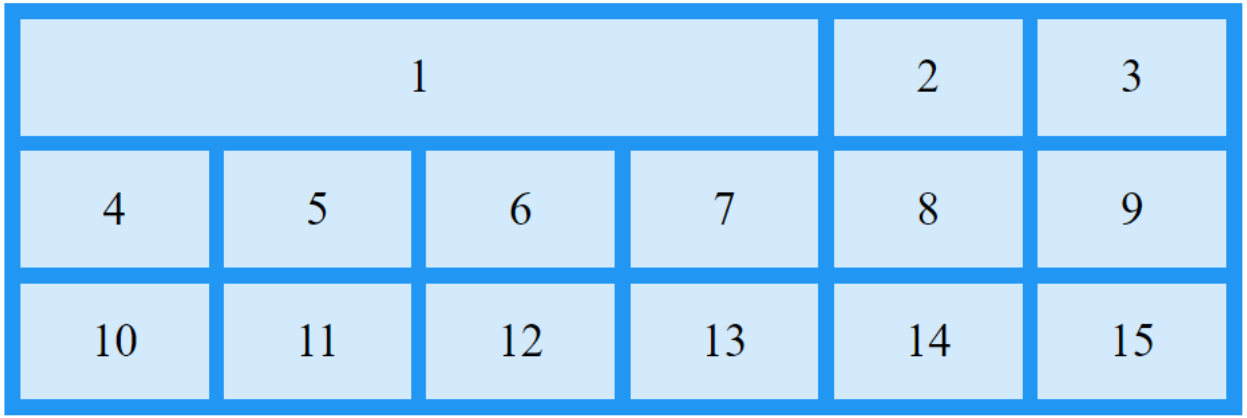
Uma grid pai é um contêiner para elemento filhos. Por padrão, um contêiner tem um item de grade para cada coluna, em cada linha, mas você pode estilizar os itens de grade para que possam abranger várias colunas e/ou linhas.

Por exemplo:



Elementos Filhos ou Itens – (Grid Column)

Faça "item1" começar na coluna 1 e termina antes da coluna 5:



Elementos Filhos ou Itens – (Grid Column)

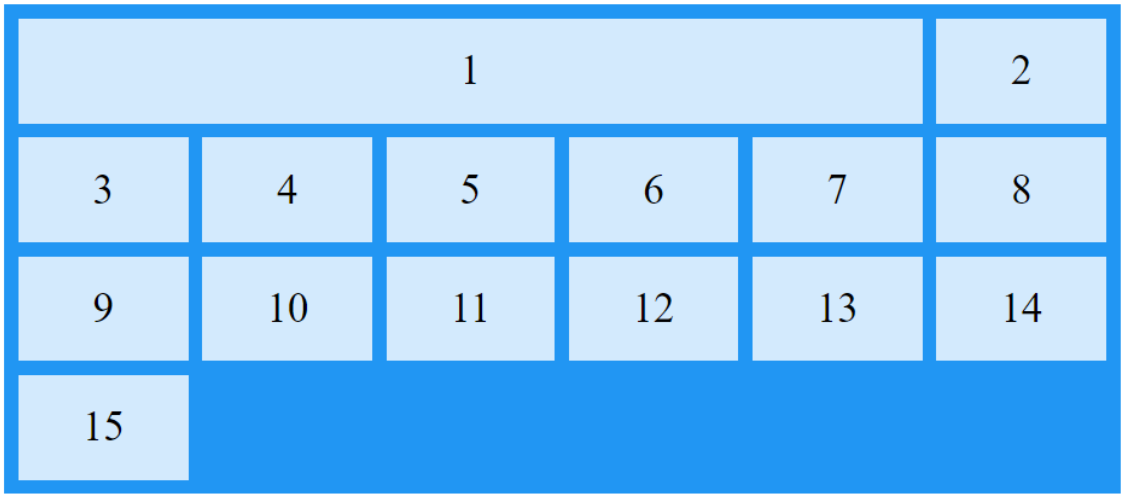
A propriedade **grid-column** define quantas colunas um item deve abranger.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item1 {
  grid-column: 1 / 5;
}
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
  <div class="item9">9</div>
  <div class="item10">10</div>
  <div class="item11">11</div>
  <div class="item12">12</div>
  <div class="item13">13</div>
  <div class="item14">14</div>
  <div class="item15">15</div>
</div>
```

Elementos Filhos ou Itens – (Grid Column)

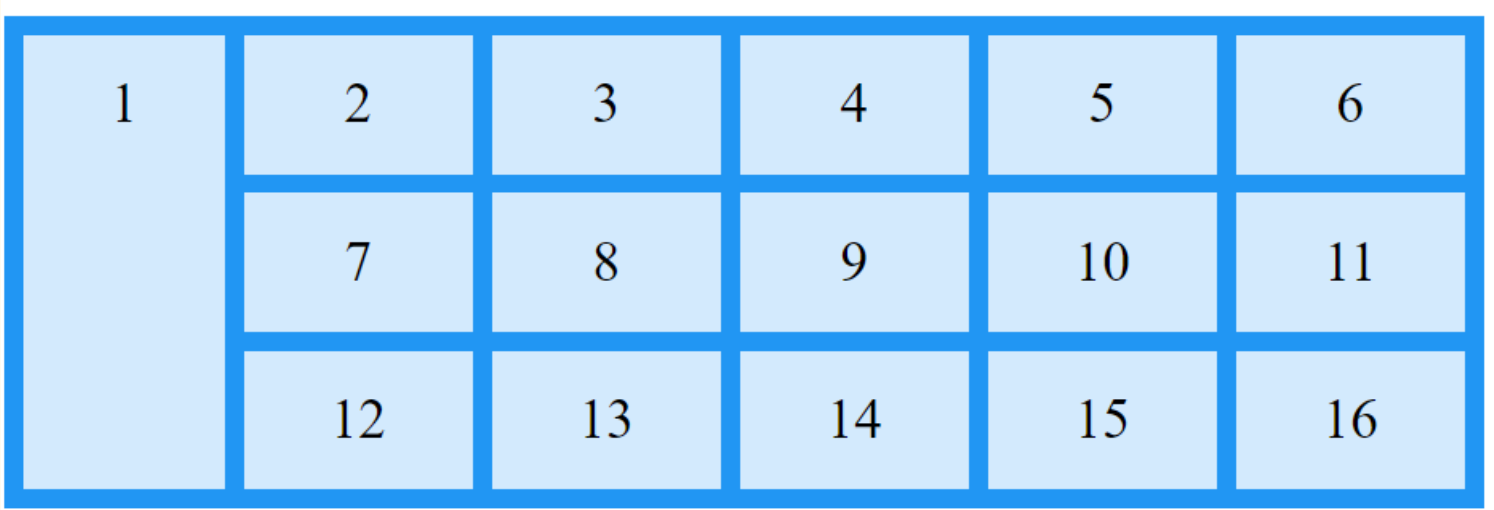
Faça "item1" começar na coluna 1 e **inclui a coluna** 5 (usando a palavra **span**):




```
.item1 { grid-column: 1 / span 5; }
```

Elementos Filhos ou Itens – (Grid Row)

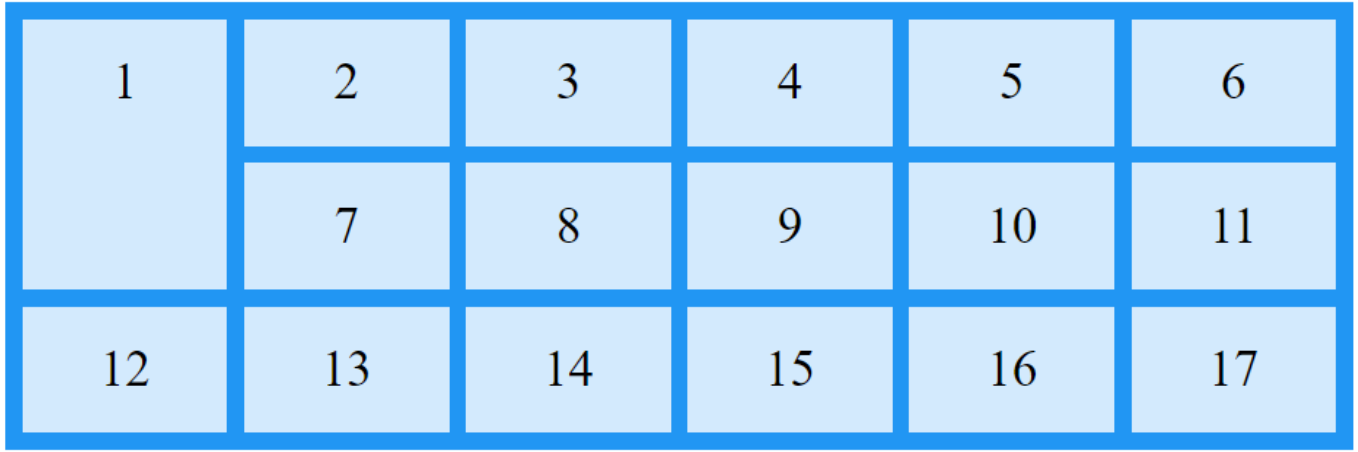
Faça "item1" começar na coluna da linha 1 e terminar na mesma coluna, só que antes da linha 4:




```
.item1 { grid-row: 1 / 4; }
```

Elementos Filhos ou Itens – (Grid Row)

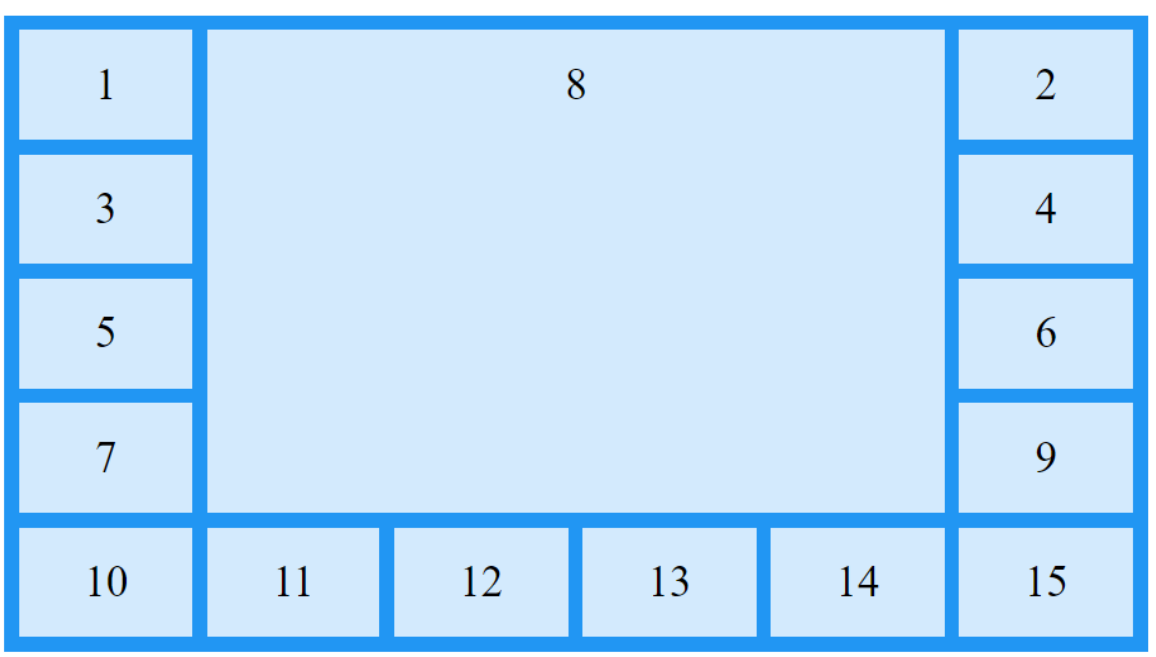
Faça "item1" começar na coluna da linha 1 e incluir na mesma coluna, a linha 2:




```
.item1 { grid-row: 1 / span 2; }
```

A propriedade Grid-Area

A propriedade grid-área é utilizada para definir o começo e o fim de uma linha e de uma coluna.



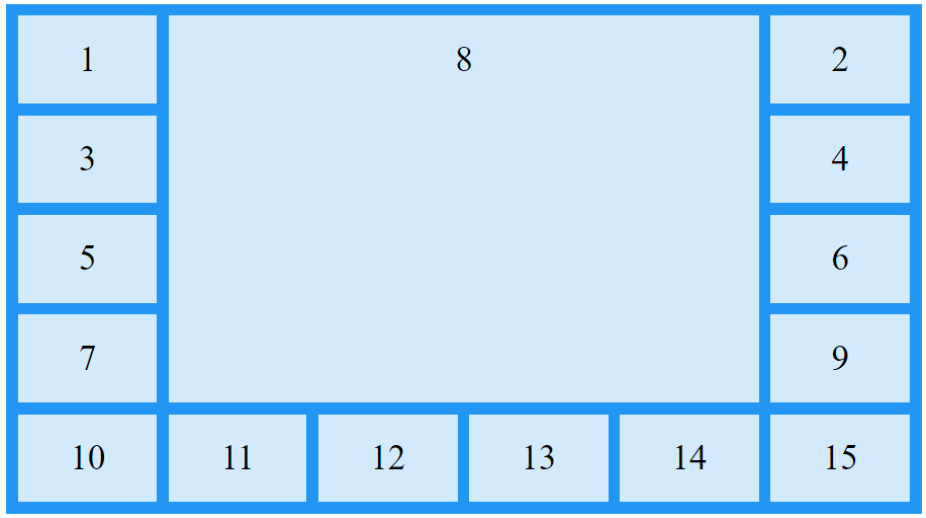

```
.item8 { grid-area: 1 / 2 / 5 / 6; }
```

A propriedade Grid-Area

A sintaxe é: **linha-começo, coluna-começo, linha-final, coluna-final**

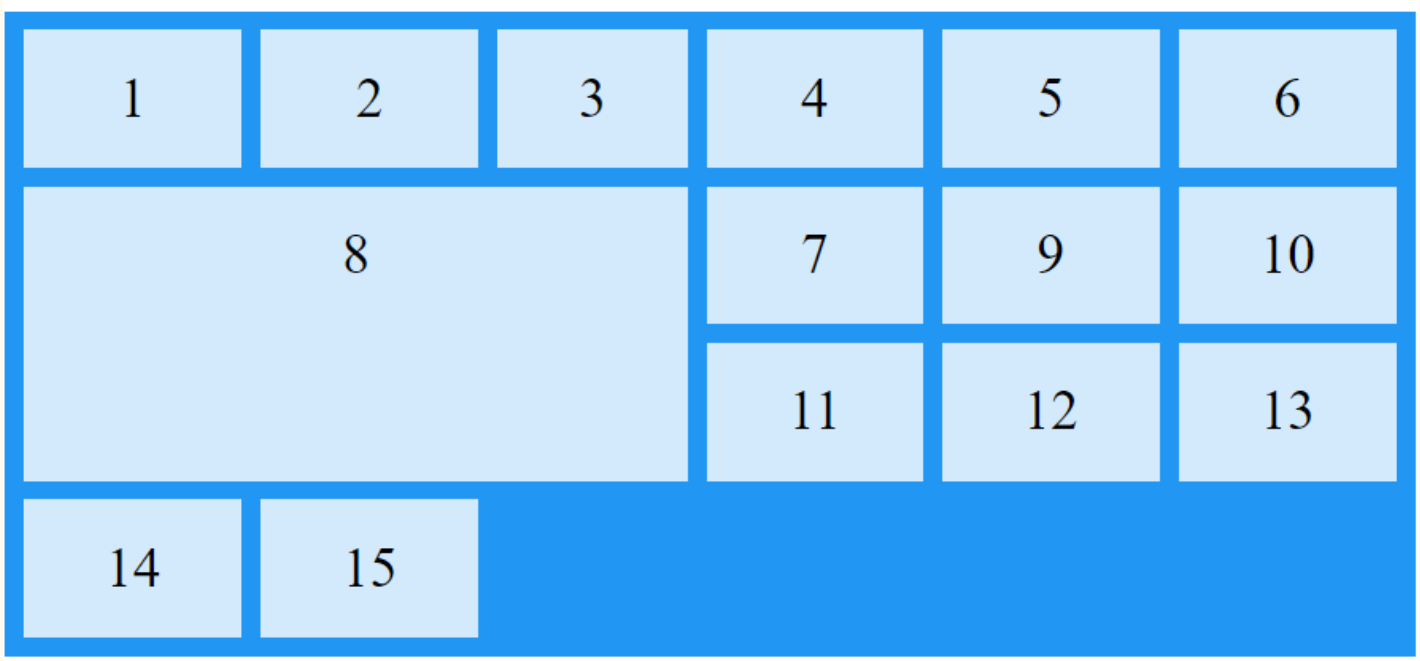
No exemplo abaixo: o **item8** começará na linha 1 e na coluna 2, e terminará na linha 5 e coluna 6:

```
.item8 { grid-area: 1 / 2 / 5 / 6; }
```



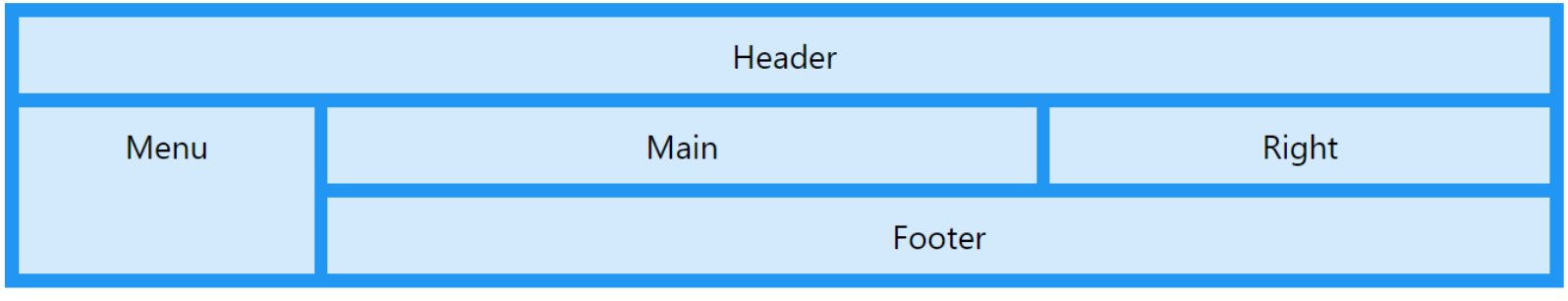
A propriedade Grid-Area

Resposta:



Nomeado os itens da Grid (grid-template-areas)

Podemos nomear os itens do nosso layout e referenciá-los no dentro do contêiner. Para isso utilizamos as propriedades **grid-area** e **grid-template-areas**:




```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
```

```
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```

Nomeado os itens da Grid (grid-template-areas)

Montamos a grade como os nomes dados.

```
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu   main   main   main   right  right'
    'menu   footer footer footer footer footer';
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```