Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS
Grid Layout

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

CSS Grid Layout Module

Sistema de layout **baseado em grade bidimensional** que muda completamente a maneira de **projetar interfaces de usuário**.

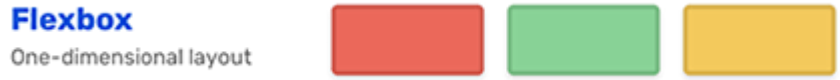
CSS sempre foi usado para fazer o layout de páginas web, mas nunca fez um bom trabalho.

- Primeiro, eram utilizadas tabelas;
- Depois `floats`;
- Enfim, posicionamento em bloco e em linha.

Todos esses métodos eram formas de ***hacks*** e deixavam de fora **muitas funcionalidades** importantes (centralização vertical, por exemplo).

O Flexbox também é uma ferramenta de layout muito boa, mas seu fluxo unidirecional pode ser aplicado em diferentes casos de uso.

E os dois módulos podem funcionar bem juntos!

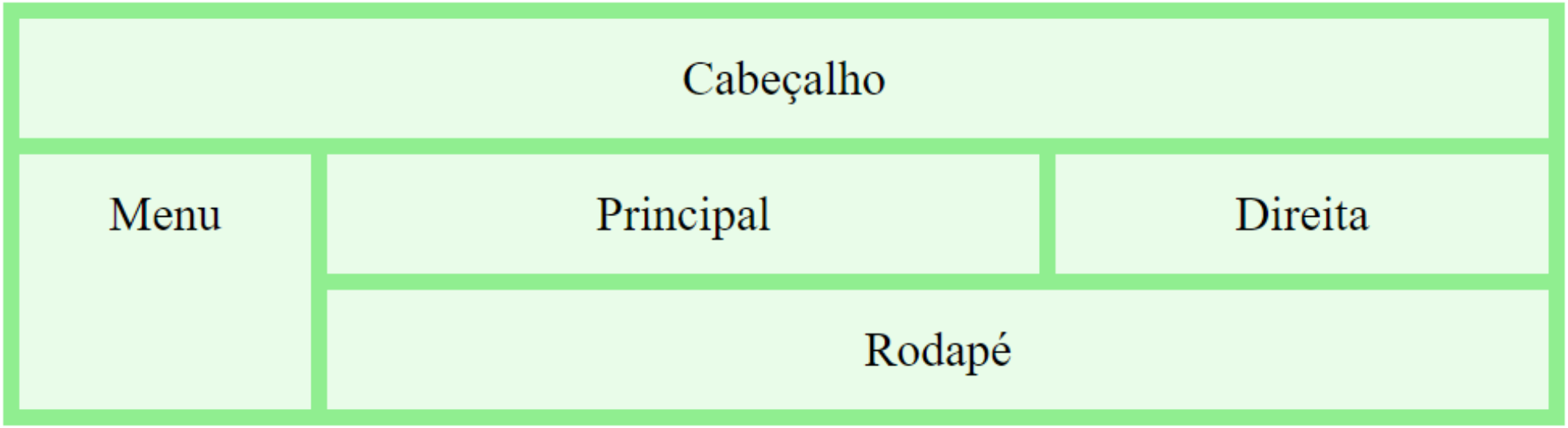


CSS Grid Layout Module

É um sistema de *layout* **baseado em grade**, com linhas e colunas.

Torna mais fácil projetar páginas web.

Dispensa a necessidade de utilizar flutuadores e posicionadores.



Exemplo_01.html

Conceitos básicos

grid layout

Elementos básicos do grid layout

É formado por um **elemento pai**, com **um ou mais filhos**.

Exemplo_02.html

1	2	3
4	5	6
7	8	9

Todos os filhos diretos do contêiner tornam-se **automaticamente grid items.**

Um elemento HTML se torna um **grid container** quando sua propriedade de **display** é definida como **grid** ou **inline-grid**.

grid items

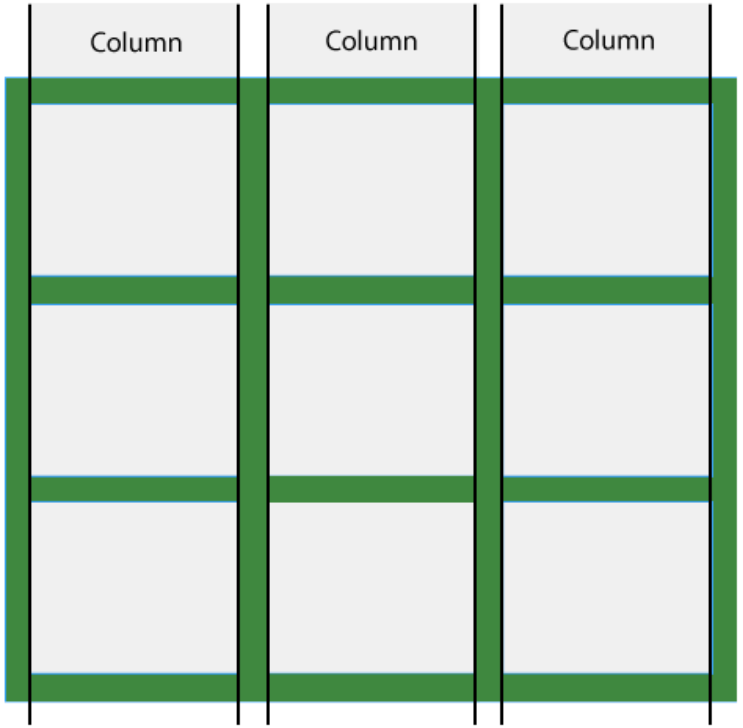
```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div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

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: lightgreen;
  padding: 10px;
}
```

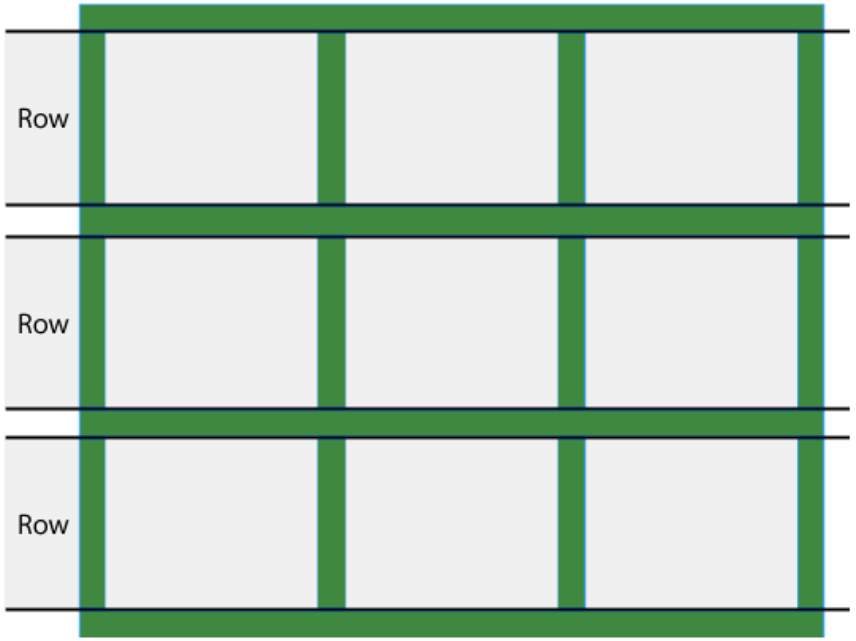
```
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid darkgreen;
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

grid columns e grid rows

Conceito similar ao que estamos acostumados com as planilhas.

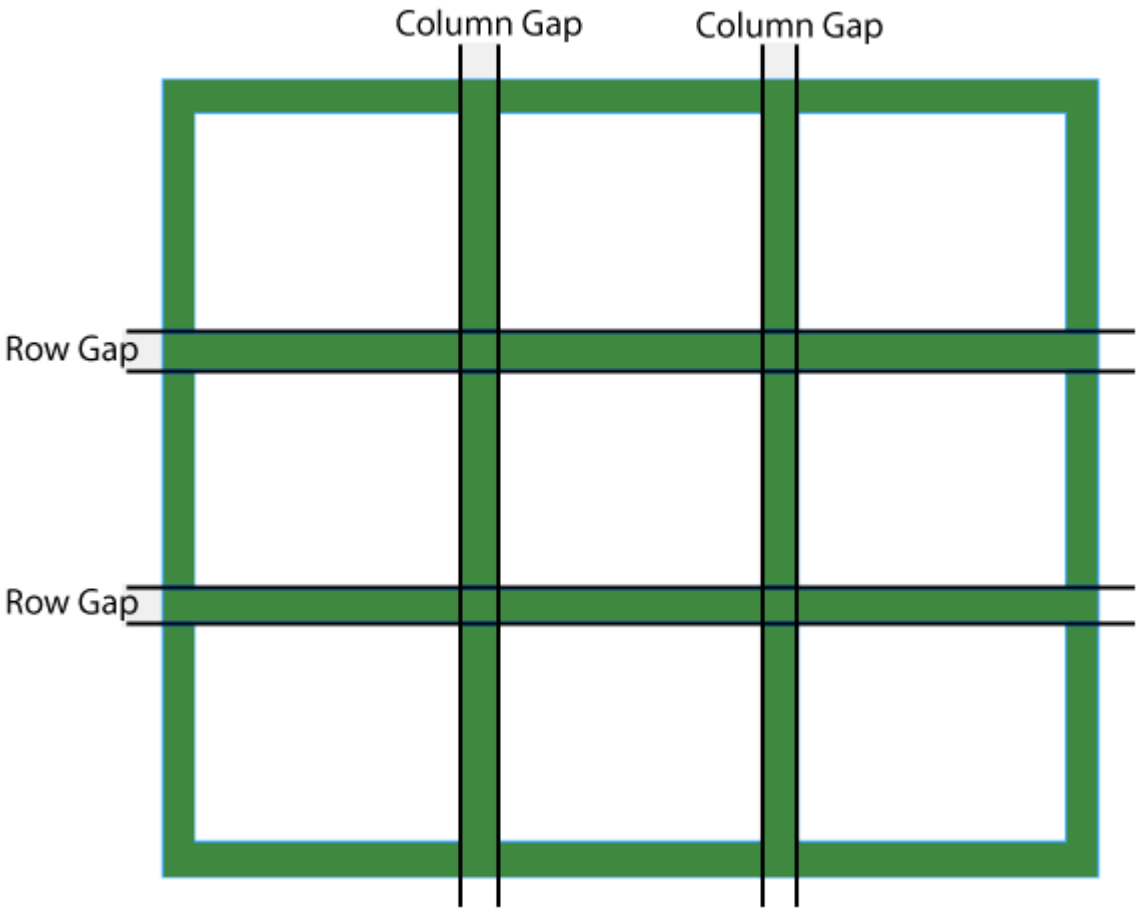


As linhas **verticais** dos `grid items` são chamadas de **colunas (`grid columns`)**.



As linhas **horizontais** dos `grid items` são chamadas de **fileiras (`grid rows`)**.

grid gaps -> lacunas



Os **espaços entre cada coluna/fileira** são chamados de **lacunas** (**`grid gaps`**).

É possível **ajustar o tamanho dos *gaps*** usando as propriedades:

`grid-column-gap`

`grid-row-gap`

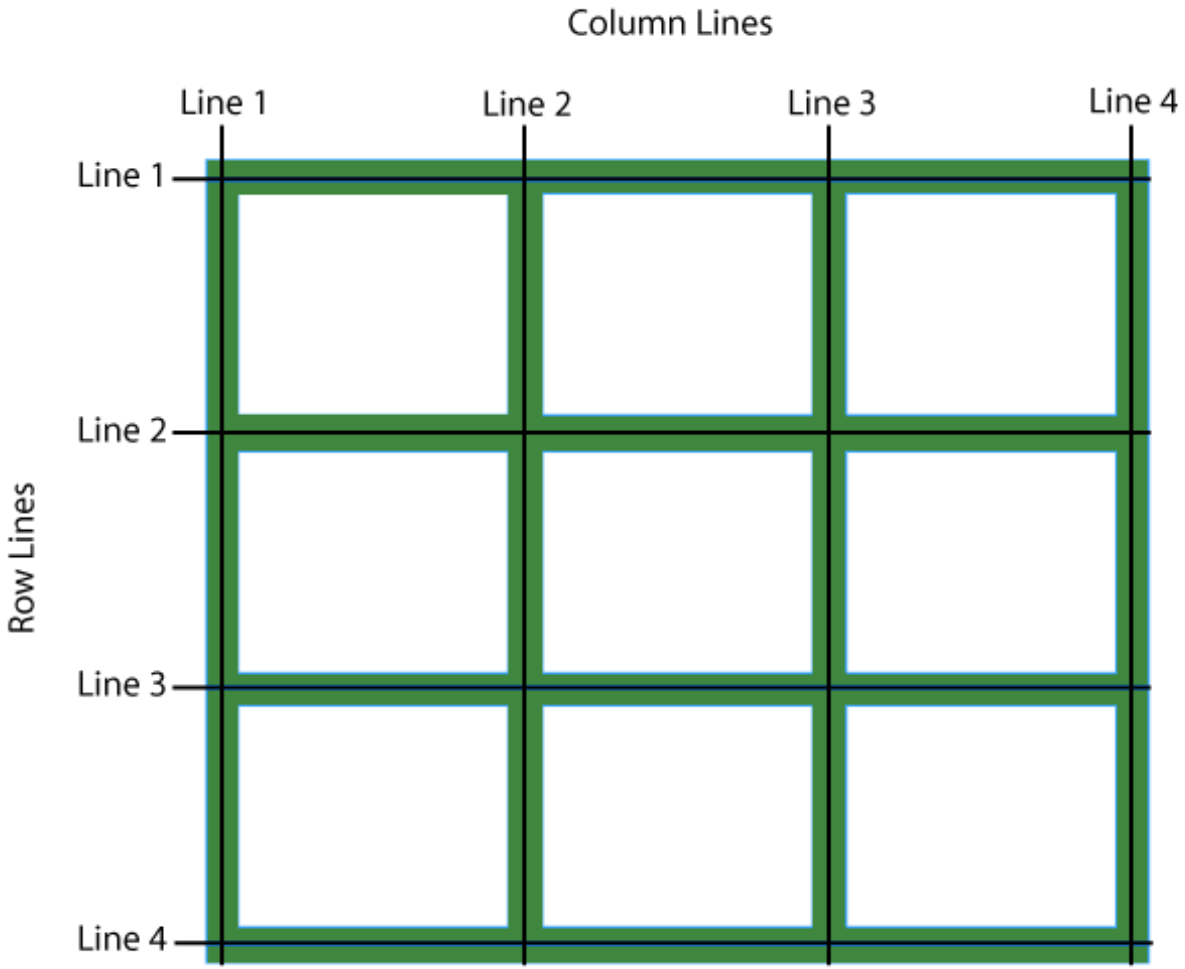
```
grid-gap: 50px 100px
grid-gap: 50px
```

Propriedade abreviada

Ao definir apenas um valor, este é aplicado em `rows` e `columns`.

grid lines

Linhas entre as colunas são chamadas de **column grid lines**.



Linhas entre as fileiras são chamadas de **row grid lines**.

Este exemplo apresenta um **grid** com três colunas e três linhas.

Isso acarreta em quatro **column lines** e quatro **row lines**.

Posicionamento - grid lines

As propriedades **grid-column-start** e **grid-column-end**, permitem definir o início e fim do `grid item` em termos de **linhas**.

columns

```
.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
```

Column line 1

Column line 3

1		2
3	4	5
6	7	8

Exemplo_03.html

rows

```
.item1 {
  grid-row-start: 1;
  grid-row-end: 3;
}
```

Row Line 1

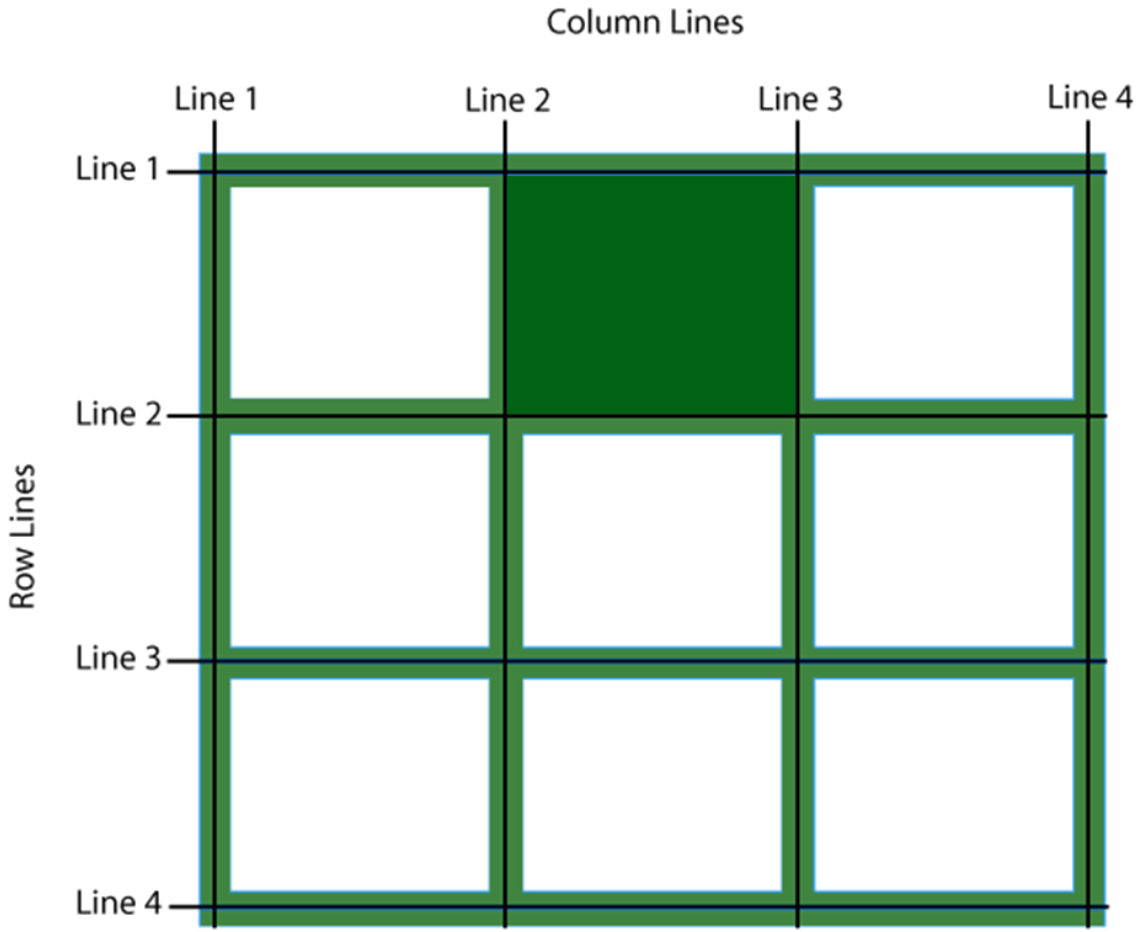
Row Line 3

1	2	3
	4	5
6	7	8

Exemplo_04.html

Da mesma forma, **grid-row-start** e **grid-row-end** possibilitam definir o início e fim do grid item em termos de **colunas**.

grid cells -> células

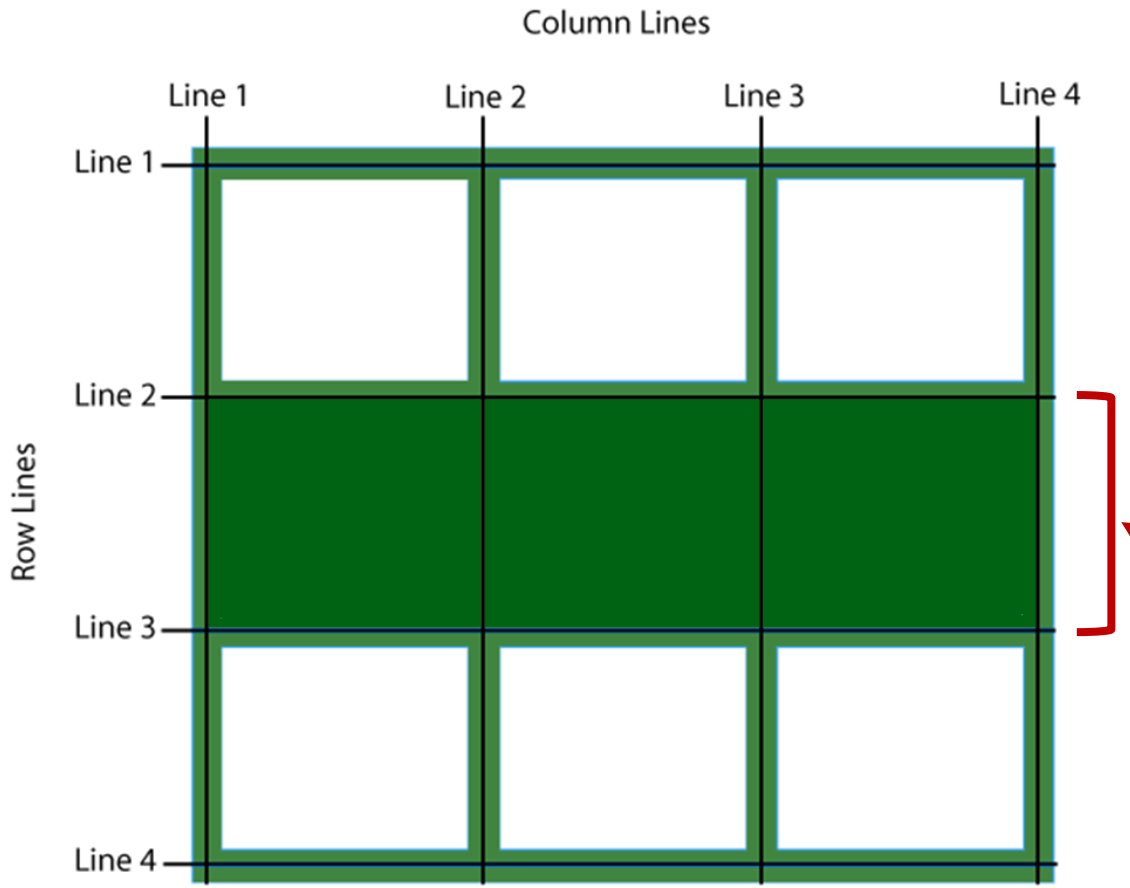


É uma "unidade" da grade (célula).

O espaço delimitado por duas `row grid lines` adjacentes e duas `column grid lines` adjacentes.

No exemplo, a **`grid cell`** destacada está entre as **`row lines`** 1 e 2 e as **`column lines`** 2 e 3.

grid track -> trilhas

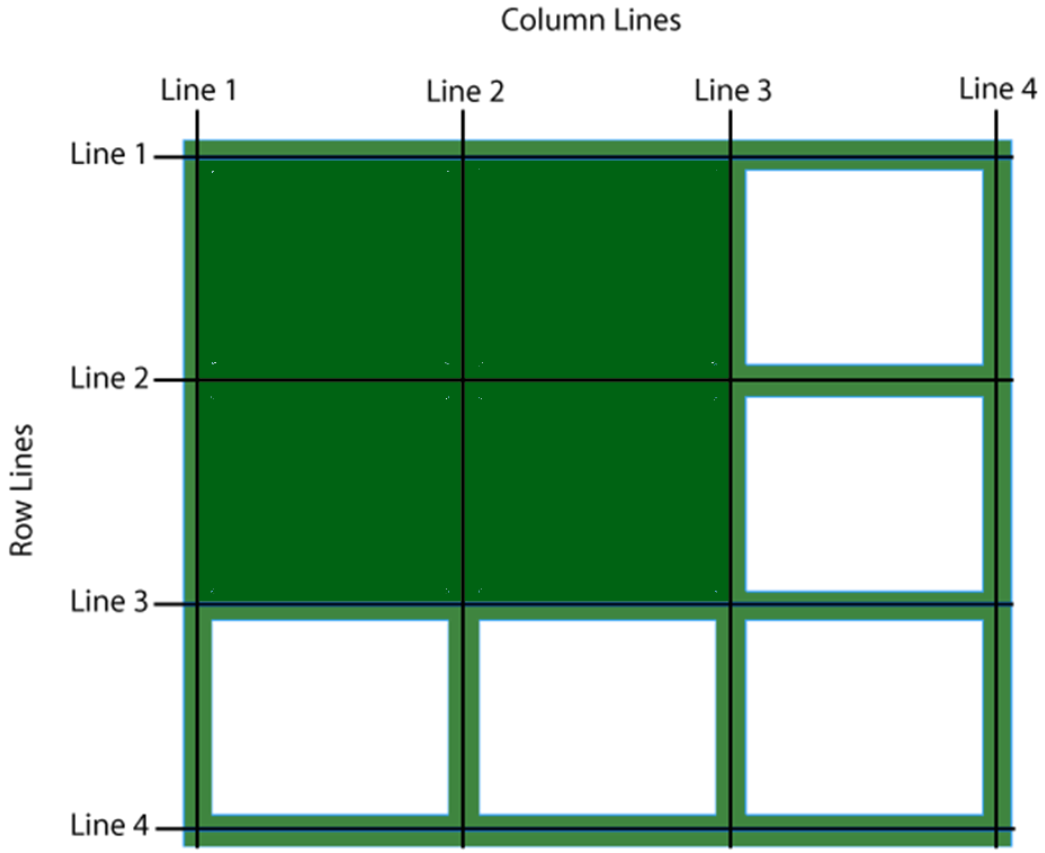


O espaço entre duas `grid lines` adjacentes.

As trilhas podem ser percebidas como as colunas ou fileiras do grid.

No exemplo, a `grid track` (trilha) este entre a segunda e a terceira `row lines`.

grid area



O espaço total delimitado por quatro `grid lines`.

Uma `grid area` pode ser composta por qualquer número de células.

No exemplo, a área é delimitada pelas:

- **`grid row lines` 1 e 3; e**
- **`grid column lines` 1 e 3.**

CSS

grid container

Propriedade display

Para fazer um elemento HTML se comportar como um **grid container**, é preciso definir a propriedade **display** como **grid** ou **inline-grid**.

Configura o container em nível de bloco

Configura o container *inline*

Exemplo.html

1	2	3
4	5	6
7	8	9

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

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

grid ou inline-grid.

1	2	3
4	5	6
7	8	9

`display:grid` e `display:inline-grid`

A principal diferença entre `display:grid` e `display:inline-grid` no CSS está no **comportamento do contêiner em relação ao fluxo de layout da página**.

`display:grid`

>> O elemento é transformado em um *grid container* e se comporta como um **elemento de bloco**. Ou seja, ele ocupa toda a largura disponível do contêiner pai, iniciando uma nova linha no fluxo do documento.

>> Usado quando desejamos que o *grid container* se comporte como um **elemento de bloco, ocupando a largura total disponível e empurrando os outros elementos para baixo.**

1	2	3
4	5	6
7	8	9

`display:inline-grid`

>> O elemento é transformado em um *grid container*, mas se comporta como um **elemento inline**. Ou seja, ele ocupa apenas a largura necessária para seu conteúdo e permite que outros elementos *inline* fiquem ao seu lado.

>> Usado quando precisamos que o contêiner de grade se comporte como um **elemento *inline*, permitindo que outros elementos inline ou inline-block fiquem na mesma linha.**

1	2	3
4	5	6
7	8	9

Propriedade grid-template-columns

Define o **número de colunas do grid**, e **pode definir a largura de cada coluna**.

Usa-se uma **lista separada por espaços**, onde **cada valor define a largura da respectiva coluna**.

Exemplo: para um *grid layout* com 4 colunas, basta especificar as larguras individuais, ou "**auto**" caso todas tenham a mesma largura.

OBS: Se mais de quatro itens forem colocados em um *grid* de 4 colunas, automaticamente uma nova linha é adicionada para os itens extras.

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
}
```

1 2 3 4

1	2	3	4
5	6	7	8

Exemplo01_grid-template-columns.html

```
.grid-container {
  display: grid;
  grid-template-columns: 80px 200px auto 40px;
}
```

1 2 3 4

1	2	3	4
5	6	7	8

Exemplo02_grid-template-columns.html

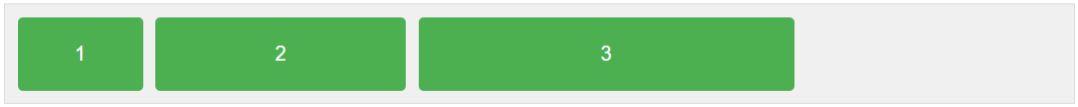
grid-template-columns

- Propriedade usada para **definir a largura das colunas** em um *grid container*.
  - Permite **especificar o número de colunas e suas larguras**.
  - É fundamental para criar **layouts de grade complexos e responsivos**.

Comprimentos fixos: como px, em, rem, etc. (ex.: 100px, 2em).

```
<div class="grid-container" style="grid-template-columns: 100px 200px 300px;">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
```

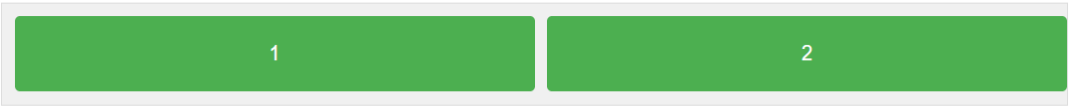
Exemplo 1: Larguras Fixas



Percentagens: relativas à largura do contêiner (ex.: 50%).

```
<div class="grid-container" style="grid-template-columns: 50% 50%;">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
</div>
```

Exemplo 2: Percentagens



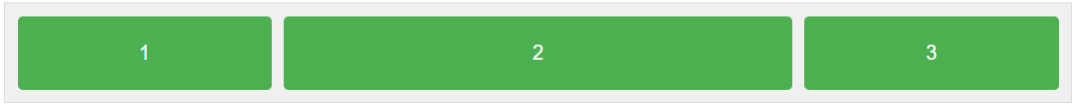
grid-template-columns

- Propriedade usada para **definir a largura das colunas** em um *grid container*.
  - Permite **especificar o número de colunas e suas larguras**.
  - É fundamental para criar **layouts de grade complexos e responsivos**.

Frações (fr): uma unidade flexível que distribui o espaço disponível (ex.: 1fr, 2fr).

```
<div class="grid-container" style="grid-template-columns: 1fr 2fr 1fr;">
          <div class="grid-item">1</div>
          <div class="grid-item">2</div>
          <div class="grid-item">3</div>
```

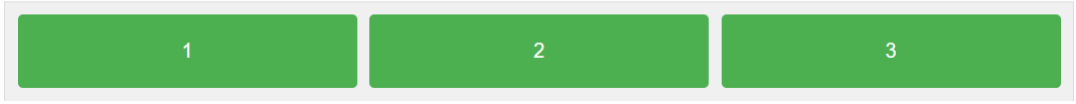
Exemplo 3: Frações (fr)



Funções como repeat(): para simplificar a criação de grades com padrões repetidos (ex.: repeat(3, 1fr)).

```
<div class="grid-container" style="grid-template-columns: repeat(3, 1fr);">
          <div class="grid-item">1</div>
          <div class="grid-item">2</div>
          <div class="grid-item">3</div>
```

Exemplo 4: Função repeat()

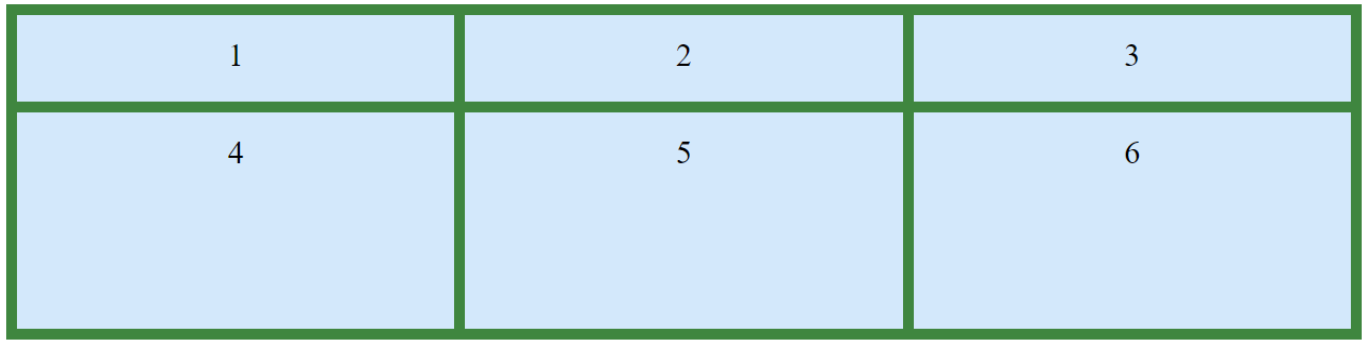


Propriedade grid-template-rows

Define a **altura** de cada fileira (linha).

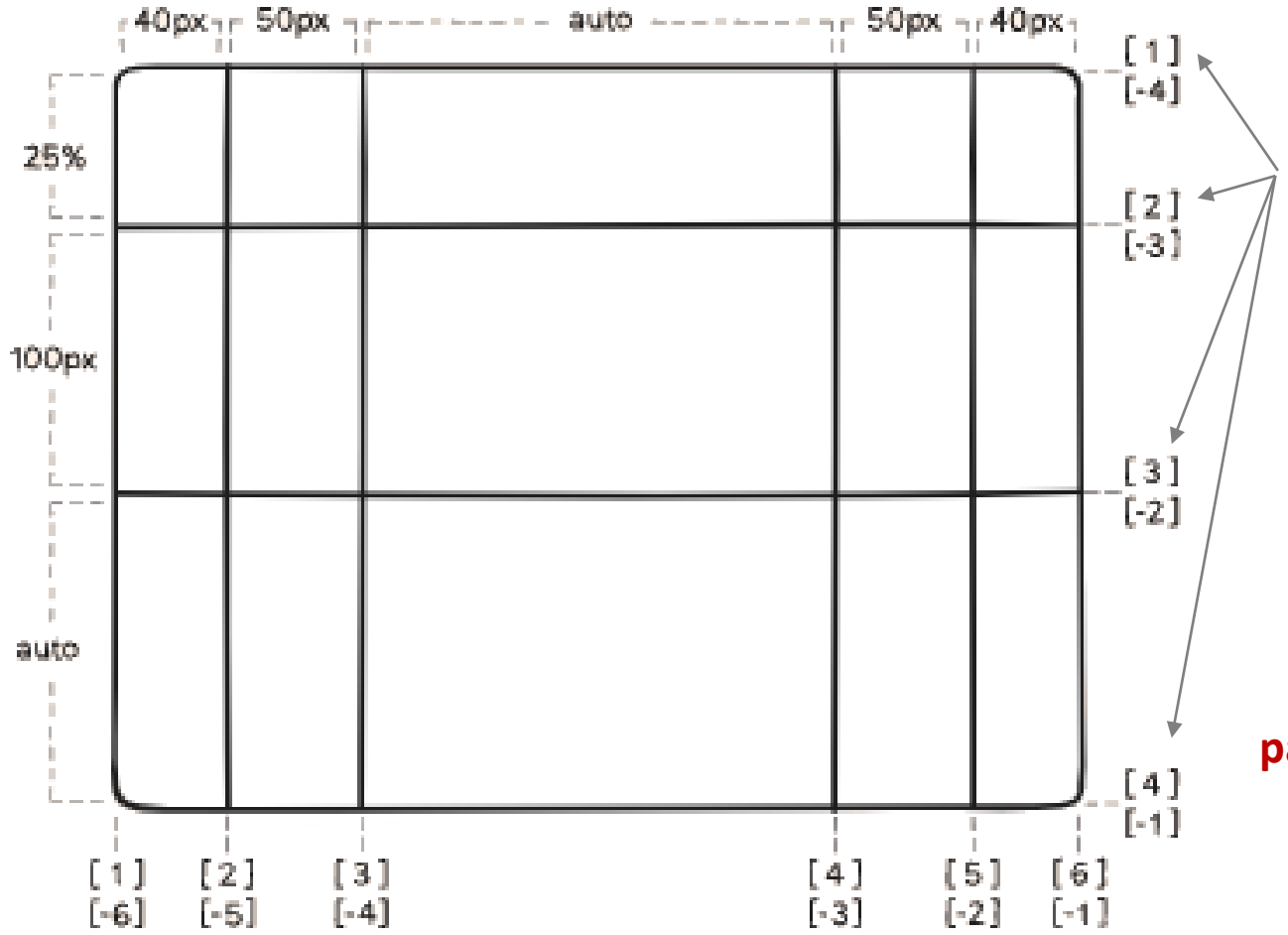
Uma lista separada por espaços, onde **cada valor define a altura da respectiva fileira (linha)**.

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



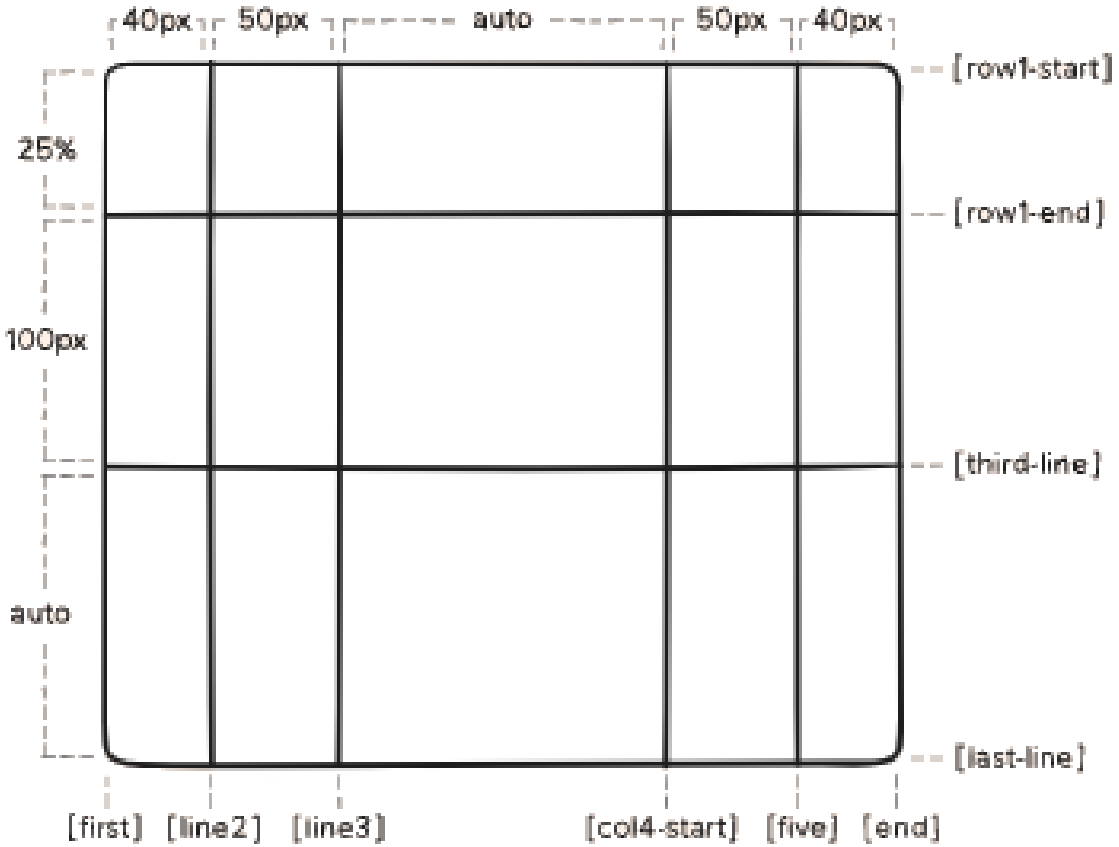
Exemplo_grid-template-rows.html

Identificando as grid lines



As grid lines recebem automaticamente números positivos dessas atribuições

Alternativamente, pode-se usar a ordem inversa, partindo do –1 para a última linha e decrementando a contagem, usando valores negativos.



É possível nomear as `grid lines` explicitamente.

Esses nomes e números poderão ser utilizados para se referir as `lines` que se deseja iniciar uma `grid area` ou uma `grid track`, por exemplo.

```
.container {
    grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
    grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
}
```

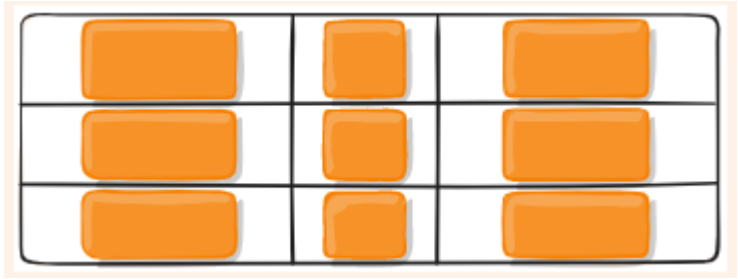
Algumas propriedades do contêiner

Propriedade justify-items

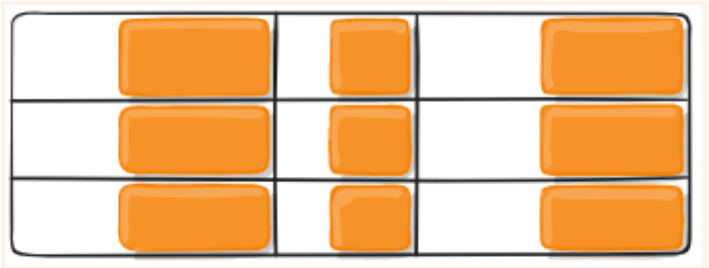
Alinha os **grid-items** ao longo do eixo da coluna (column).

```
.container {
  justify-items: start | end | center | stretch;
}
```

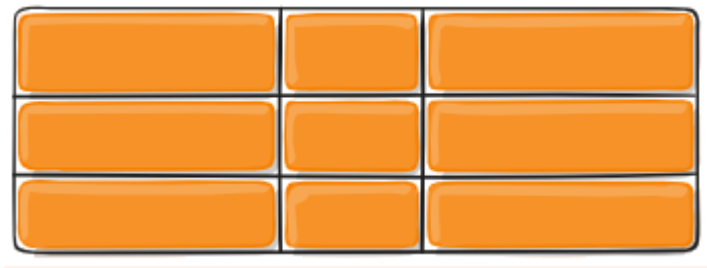
start



center



end



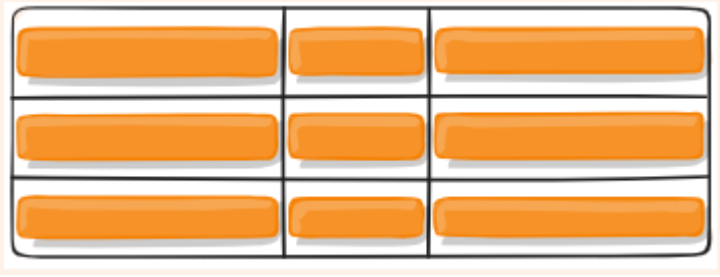
strech

Propriedade `align-items`

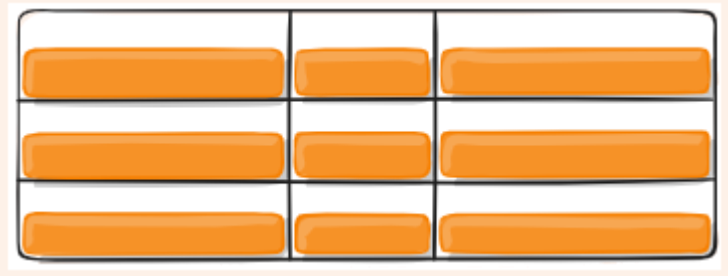
Alinha os **`grid-items`** ao longo do eixo da fileira (row).

```
.container {
  align-items: start | end | center | stretch;
}
```

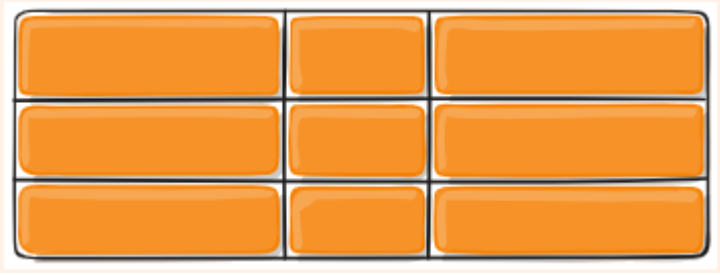
start



center



end



strech

Propriedade place-items

Configura ambas a propriedades **align-items** e **justify-items** na mesma declaração.

```
.container {
  place-items: <align-items> <justify-items>;
}
```

Útil para gerar uma centralização multidirecional rápida:

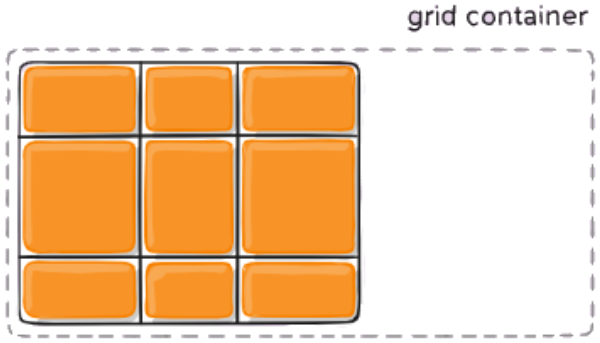
```
.center {
  display: grid;
  place-items: center;
}
```

Propriedade `justify-content`

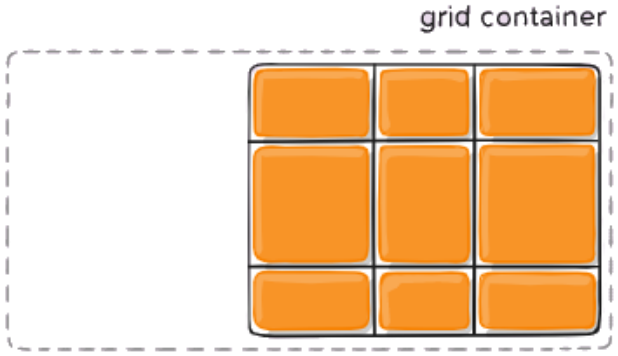
Às vezes, o tamanho total do **`grid`** pode ser menor que o tamanho do **`grid-container`**.

Com **`justify-content`**, é possível definir o alinhamento para a **`grid`** dentro **`grid-container`**.

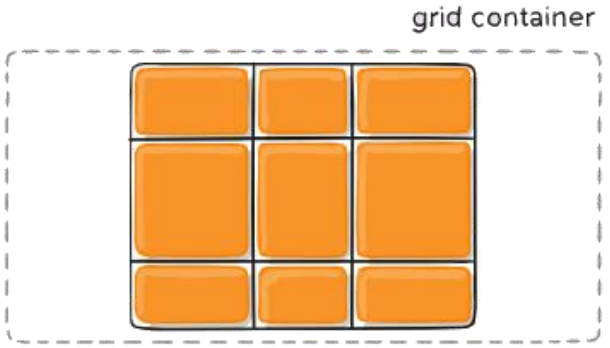
```
.container {
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```



`start`
Alinha o grid com a **borda inicial** do contêiner.



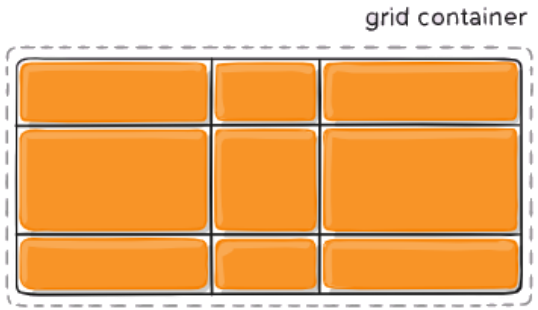
`end`
Alinha o grid com a **borda final** do contêiner



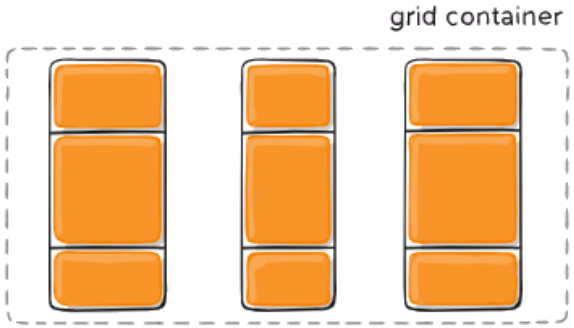
`center`
Alinha o grid no **centro** do contêiner

Propriedade `justify-content`

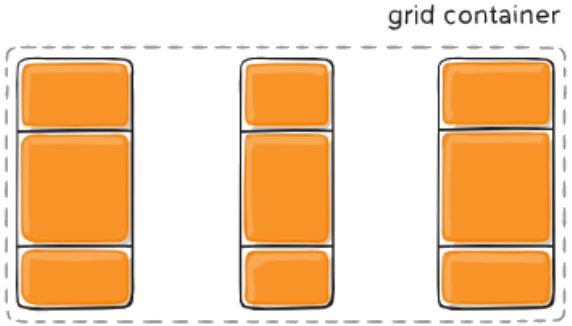
```
.container {
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```



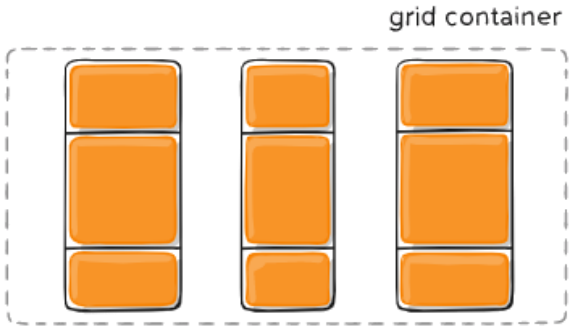
`stretch`: redimensiona os `grid-items` para **preencher toda a largura do contêiner**



`space-around`: coloca uma quantidade **uniforme de espaço** entre cada `grid-item`, com **espaços de metade deste tamanho nas extremidades**



`space-between`: coloca uma **quantidade uniforme de espaço** entre cada `grid-item` da grade, **sem espaço nas extremidades**.



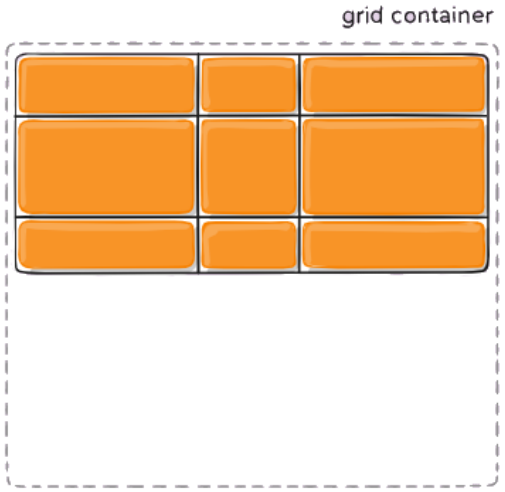
`space-evenly`: coloca uma **quantidade uniforme de espaço** entre cada `grid-item`, **incluindo as extremidades**

Propriedade `align-content`

Às vezes, o tamanho total do **`grid`** pode ser menor que o tamanho do **`grid-container`**.

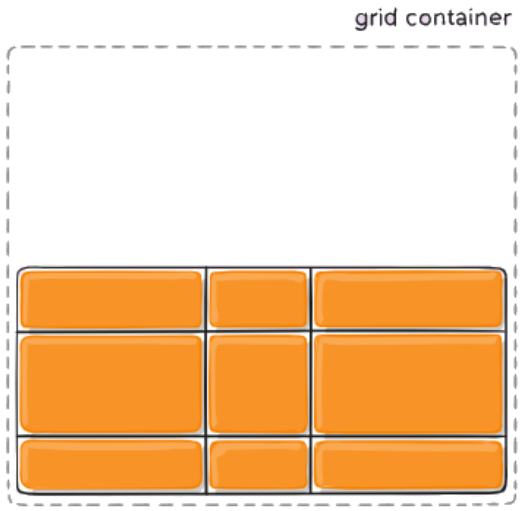
`align-content` é usada para alinhar **verticalmente** toda a grid dentro do contêiner.

```
.container {
        align-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```



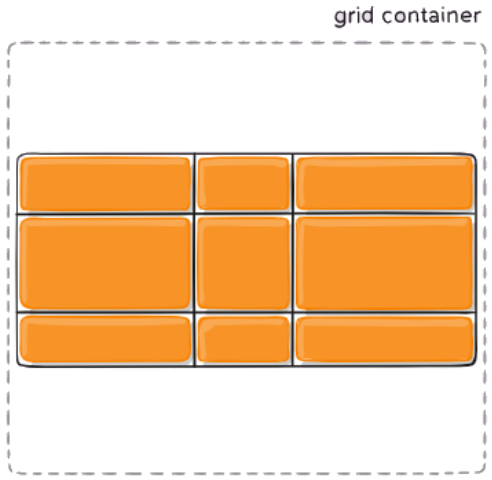
`start`

Alinha o grid com a **borda inicial** do contêiner.



`end`

Alinha o grid com a **borda final** do contêiner

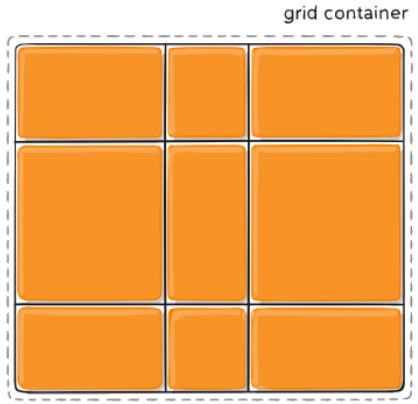


`center`

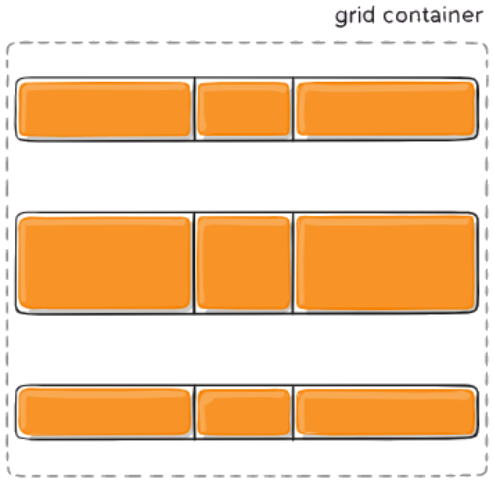
Alinha o grid no **centro** do contêiner

Propriedade `align-content`

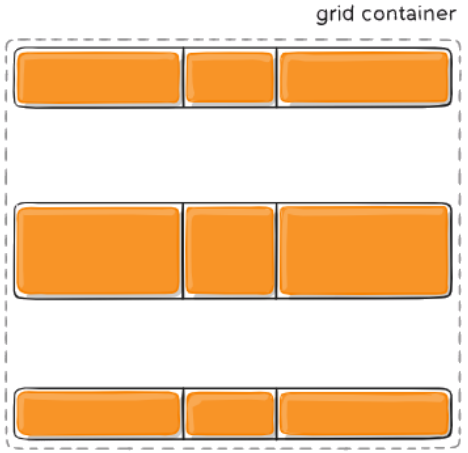
```
.container {
  align-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```



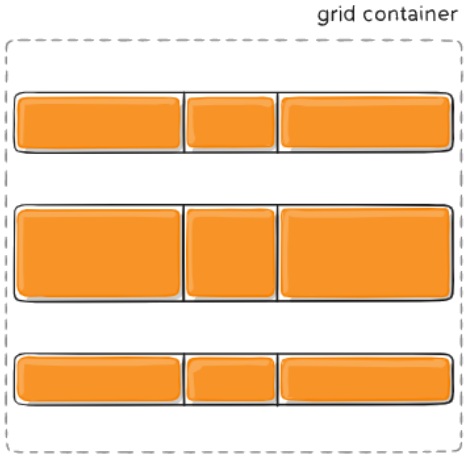
`stretch`: **redimensiona** os `grid-items` para **preencher toda a altura** do contêiner



`space-around`: coloca uma **quantidade uniforme de espaço** entre cada `grid-item`, com **espaços de metade deste tamanho nas extremidades**



`space-between`: coloca uma quantidade **uniforme de espaço** entre cada `grid-item`, **sem espaço nas extremidades**



`space-evenly`: coloca uma **quantidade uniforme de espaço** entre cada `grid-item`, **incluindo as extremidades**

Propriedade `place-content`

Configura ambas a propriedades **`align-content`** e **`justify-content`** na mesma declaração.

```
.container {
  place-content: <align-content> <justify-content>;
}
```

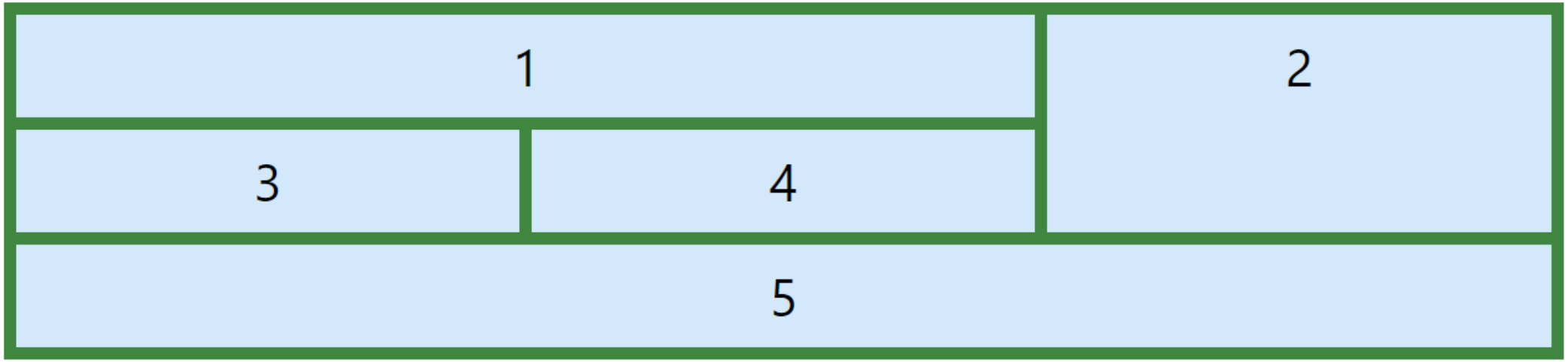
CSS

grid items

Elementos filhos - `grid items`

Um *grid container* contém **`grid items`**.

Por padrão, um container tem um `grid item` para cada coluna, em cada linha.



Mas é possível estilizar o **`grid items`** para que eles ocupem várias colunas e/ou linhas (fileiras).

Propriedade `grid-column`

Define em qual(is) **coluna(s)** colocar um **`grid item`**.

É possível definir onde o `grid item` começa e onde ele termina.

1				2	3
4	5	6	7	8	9
10	11	12	13	14	15

Para posicionar um item pode-se referir **aos números de linha**.

Também é possível usar a palavra-chave "**span**" para definir quantas colunas o item irá abranger.

`grid-column` propriedade abreviada para as propriedades:

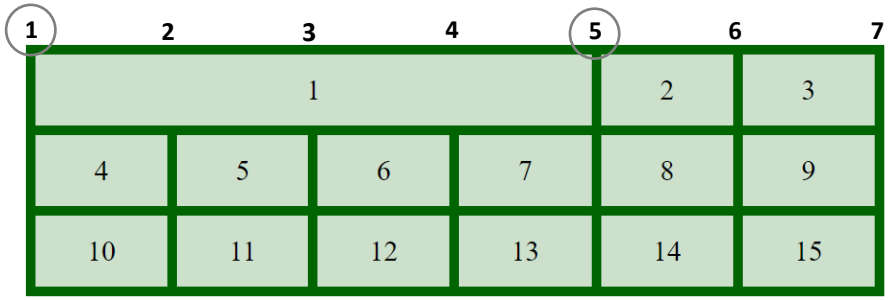
`grid-column-start;` e **`grid-column-end`**.

Propriedade grid-column

Para posicionar um item se referindo aos números de linha.

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

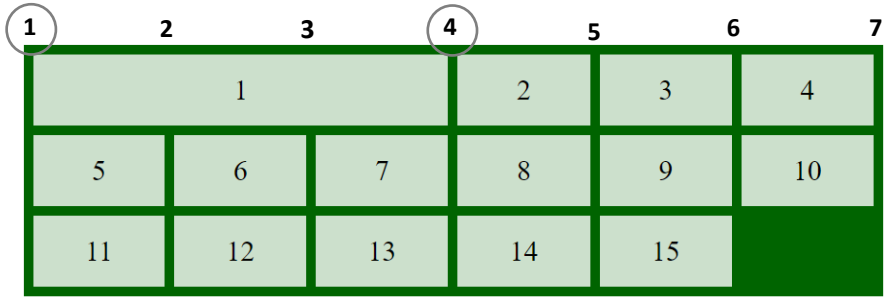
item1 começa na coluna 1 e termina antes da coluna 5



Ou é possível usar a palavra-chave "span" para definir quantas colunas o item abrangerá.

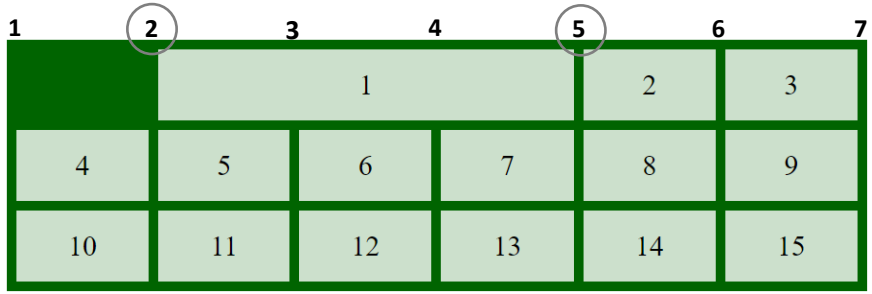
```
.item1 {
  grid-column: 1 / span 3;
}
```

item1 começa na coluna 1 e se espalha por 3 colunas




```
.item1 {
  grid-column: 2 / span 3;
}
```

item1 começa na coluna 2 e se espalha por 3 colunas



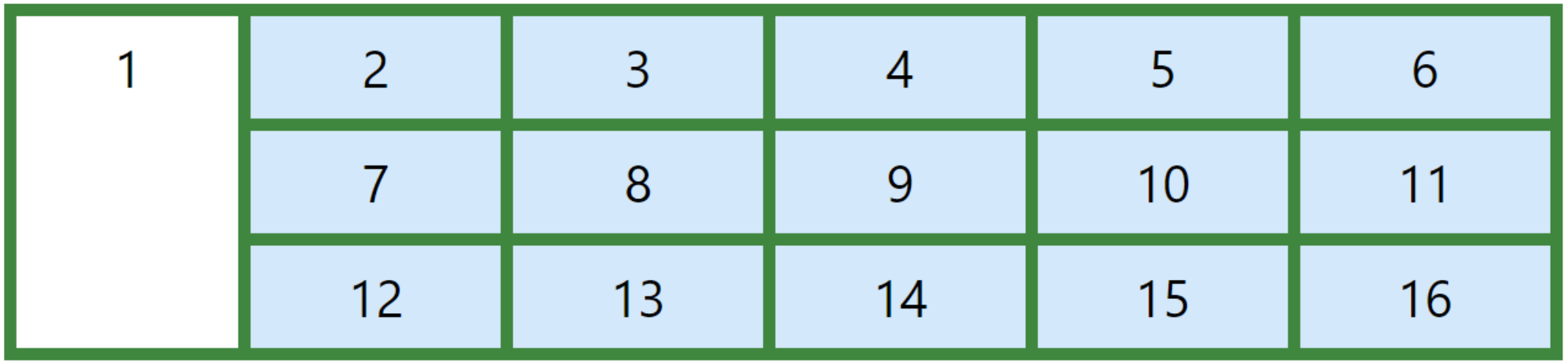

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="">2</div>
  <div class="">3</div>
  <div class="">4</div>
  <div class="">5</div>
  <div class="">6</div>
  <div class="">7</div>
  <div class="">8</div>
  <div class="">9</div>
  <div class="">10</div>
  <div class="">11</div>
  <div class="">12</div>
  <div class="">13</div>
  <div class="">14</div>
  <div class="">15</div>
</div>
```

Exemplo_05.html

Propriedade `grid-row`

Define em qual(is) **fileira(s)** colocar um **`grid item`**.

É possível definir onde o **`grid item` começa e onde ele termina**.



Para colocar um item pode-se referir aos **números de linha da fileira.**

Ou usar a palavra-chave "**span**" para definir quantas fileiras o item irá abranger.

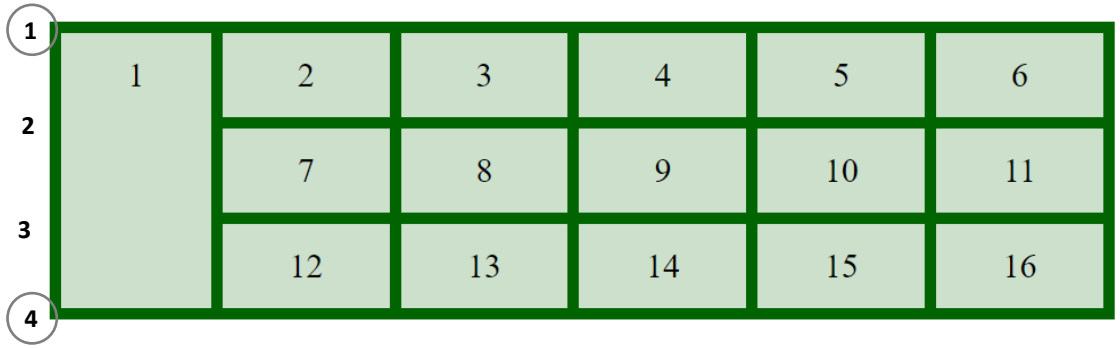
`grid-row` é uma propriedade abreviada para as propriedades:

`grid-row-start`
`grid-row-end`

Propriedade grid-row

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

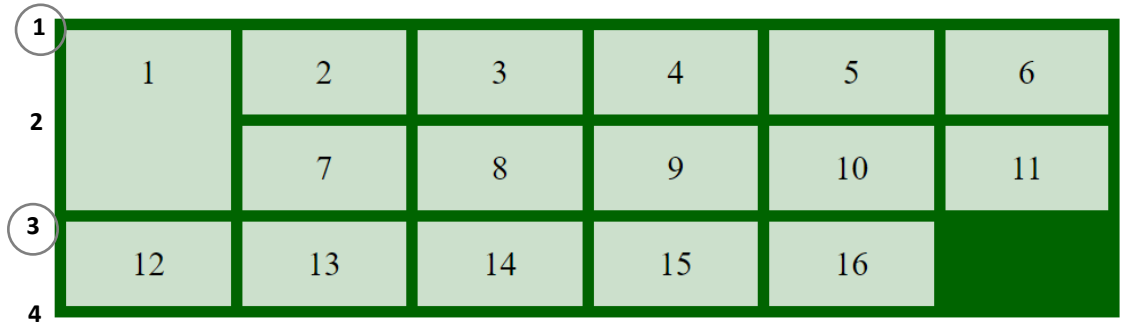
item1 começa na linha 1 e termina antes da linha 4



Exemplo_06.html

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

item1 começa na linha 1 e se espalha por 2 linhas

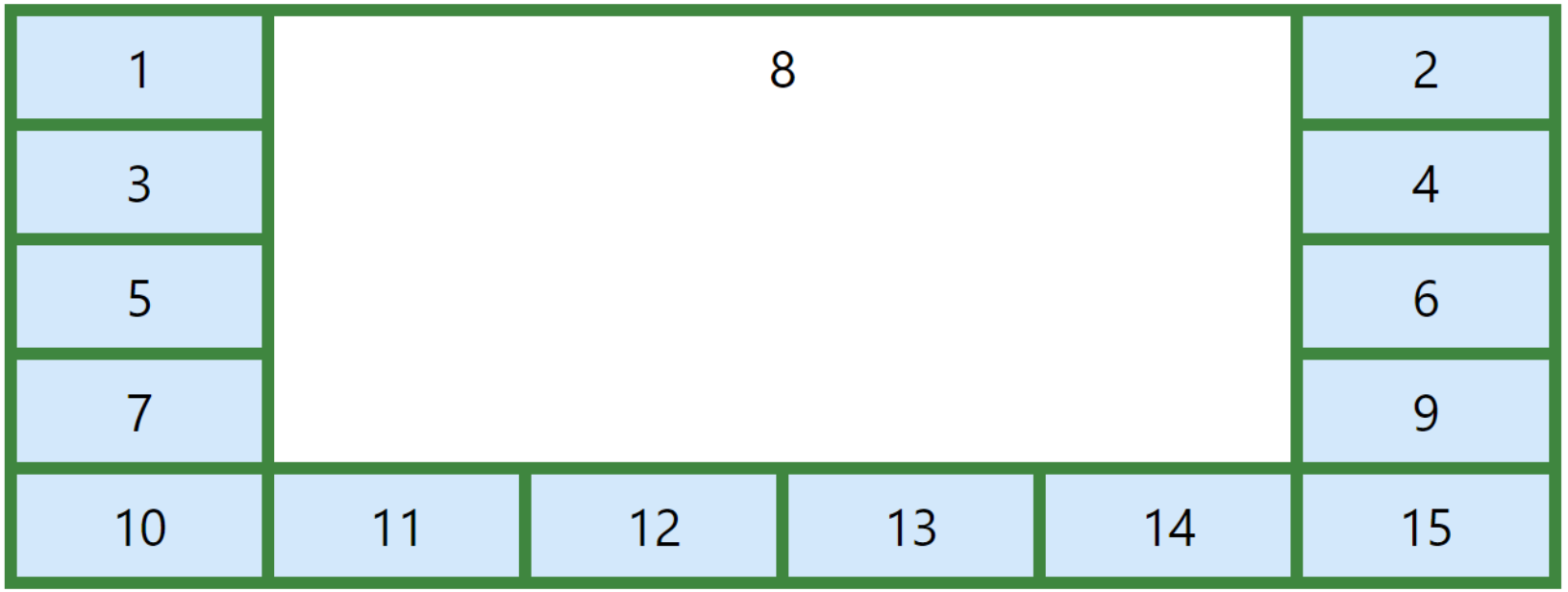



```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="">2</div>
  <div class="">3</div>
  <div class="">4</div>
  <div class="">5</div>
  <div class="">6</div>
  <div class="">7</div>
  <div class="">8</div>
  <div class="">9</div>
  <div class="">10</div>
  <div class="">11</div>
  <div class="">12</div>
  <div class="">13</div>
  <div class="">14</div>
  <div class="">15</div>
  <div class="">16</div>
</div>
```

Propriedade `grid-area`

Permite definir **uma área do `grid`**.

É uma abreviação que permite o uso em conjunto das propriedades:
`grid-column-start`, `grid-column-end`, `grid-row-start` e **`grid-row-end`**.



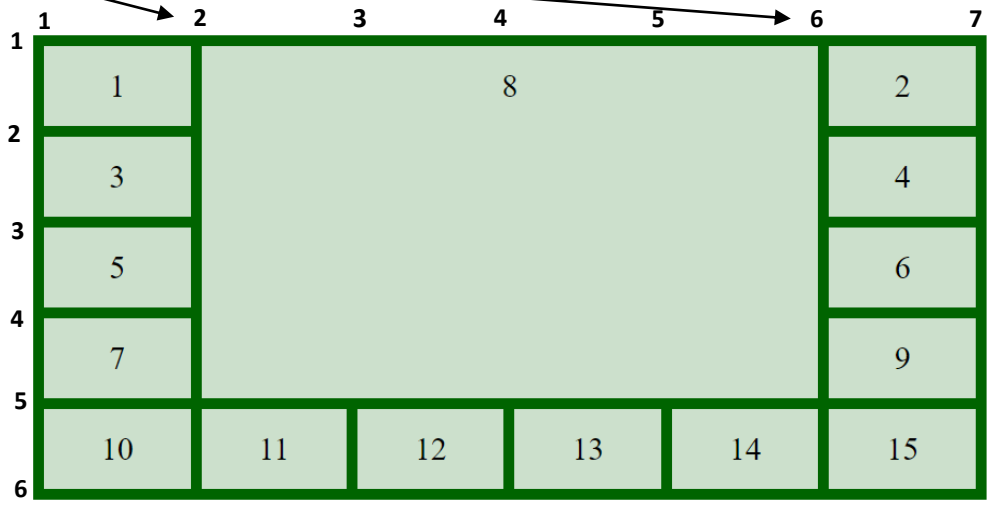
Propriedade grid-area

grid-row-start / grid-column-start / grid-row-end / grid-column-end

item1 começa na coluna 2 e termina na 6

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

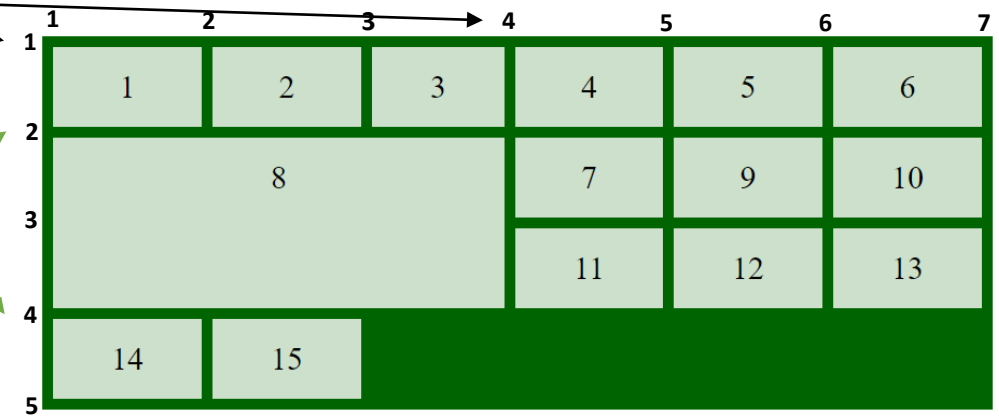
item8 começa na linha 1 e termina na linha 5



item8 começa na coluna 1 e se espalha por 3 elementos

item8 começa na linha 2 e se espalha por 2 elementos

```
.item8 {
  grid-area: 2  / 1  / span 2 / span 3;
}
```

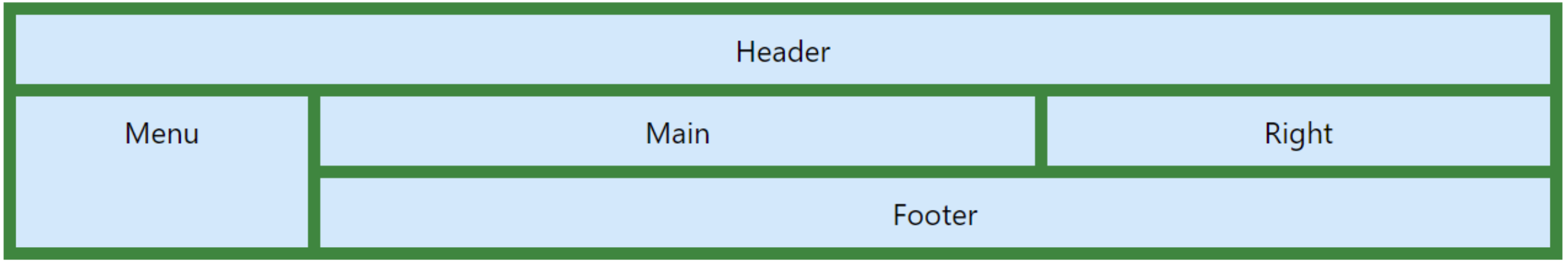


Exemplo_07.html

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="">2</div>
  <div class="">3</div>
  <div class="">4</div>
  <div class="">5</div>
  <div class="">6</div>
  <div class="">7</div>
  <div class="item8">8</div>
  <div class="">9</div>
  <div class="">10</div>
  <div class="">11</div>
  <div class="">12</div>
  <div class="">13</div>
  <div class="">14</div>
  <div class="">15</div>
</div>
```

Nomeando os grid items

A propriedade **grid-area** também pode ser usada para **atribuir nomes aos grid items**.



Os **grid items** com nomes podem ser referenciados pela propriedade **grid-template-areas** do grid container.

Nomes com grid-area

grid-area fornece um nome a um item **para que ele possa ser referenciado por um modelo** criado com a propriedade **grid-template-areas**.

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="">2</div>
  <div class="">3</div>
  <div class="">4</div>
  <div class="">5</div>
  <div class="">6</div>
</div>
```

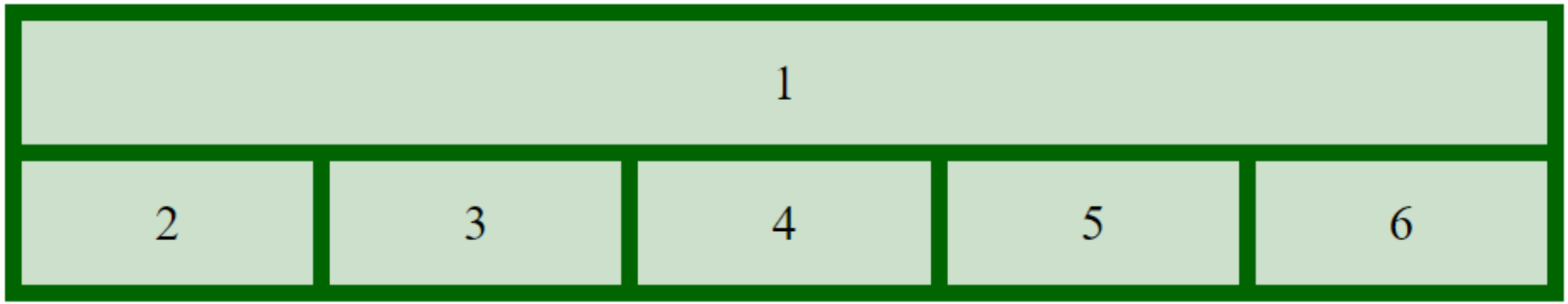
```
.item1 {
  grid-area: minhaArea;
}

.grid-container {
  display: grid;
  grid-template-areas: 'minhaArea minhaArea minhaArea minhaArea minhaArea';
  grid-gap: 10px;
  background-color: darkgreen;
  padding: 10px;
}
```

item1, é chamado de "**minhaArea**" e ocupará o lugar de todas as cinco colunas:

coluna 1 coluna 2 coluna 3 coluna 4 coluna 5

Exemplo_08.html

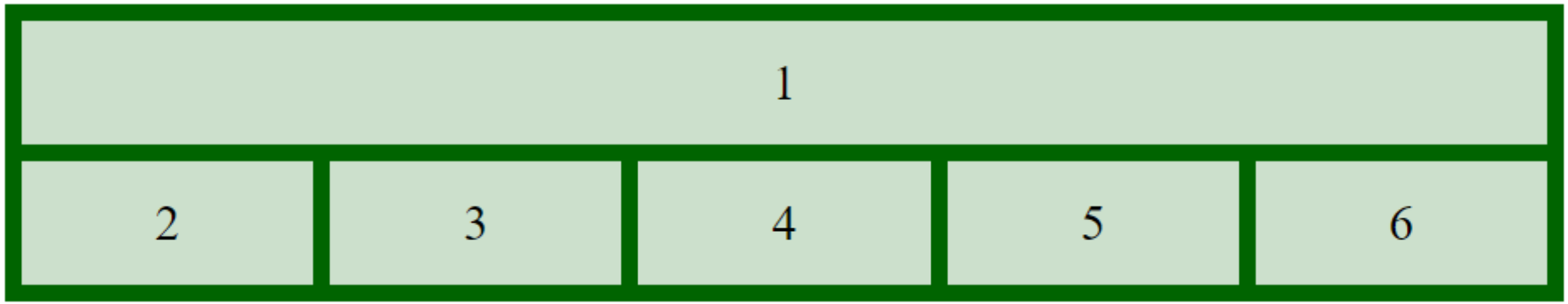


Sintaxe `grid-template-areas`

Cada fileira é definida por apóstrofos: **‘row’**

```
grid-template-areas: 'minhaArea minhaArea minhaArea minhaArea minhaArea';
```

As **colunas** de cada fileira são delimitadas pelos apóstrofos, separadas por um espaço.



grid item anônimo

Um **ponto final (.)** representa um ***grid item* sem nome**.

```
.item1 {
  grid-area: minhaArea;
}

.grid-container {
  grid-template-areas: 'minhaArea minhaArea . . .';
}
```

col1 col2 col3 col4 col5

minhaArea vai ocupar duas colunas em um layout de grid composto por cinco colunas

1		2	3	4
5	6	7	8	9

Exemplo_08-2.html

Como ocupar duas linhas?

Para definir duas linhas, é preciso definir a **coluna da segunda linha dentro de outro conjunto de apóstrofos**.

```
.item1 {
  grid-area: minhaArea;
}
```

```
.grid-container {
  grid-template-areas: 'minhaArea minhaArea . . .' 'minhaArea minhaArea . . .';
}
```

minhaArea ocupa o lugar de duas **colunas (de cinco).**

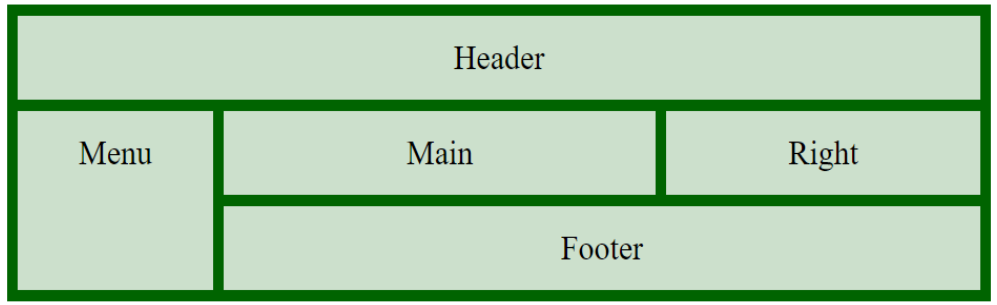
E se estende por **duas linhas**.

1	2	3	4
	5	6	7

Exemplo_08-3.html

Montando um *layout*...

Exemplo_09.html



1) Definir um `grid container`

```
<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>
```

2) Definir os `grid items` e nomeá-los com `grid area`

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

3) Estilo das `divs` que serão o `grid items`, dentro do `grid-container`

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

4) Criar o grid container, e definir na `grid-template-areas` a organização do layout.

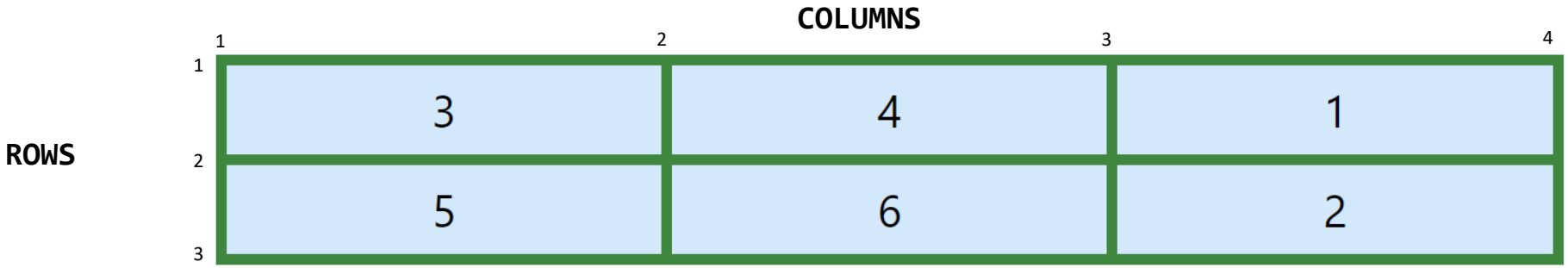
```
.grid-container {
  display: grid;
  grid-template-areas:
    'header header  header  header header header'
    'menu   main    main    main   right  right'
    'menu   footer  footer  footer footer footer';
  grid-gap: 10px;
  background-color: darkgreen;
  padding: 10px;
}
```

6 colunas

Linha_1
Linha_2
Linha_3

grid-area - Ordem dos itens

O *Grid Layout* permite **posicionar os itens no lugar desejado**.



O primeiro item no código HTML não precisa aparecer como o primeiro `grid item`.

grid-row-start grid-column-start grid-row-end grid-column-end

```
<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>
```

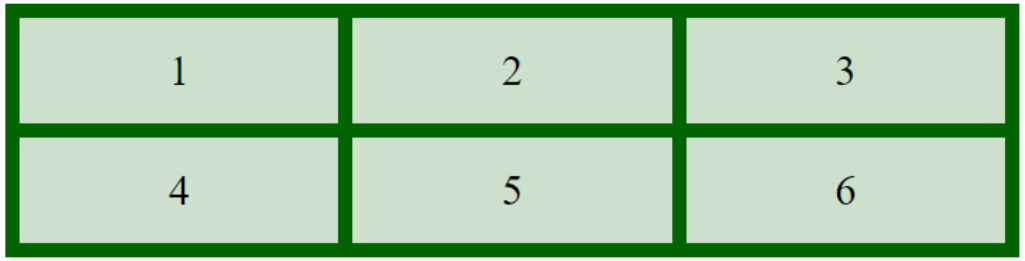
```
.item1 { grid-area: 1 / 3 / 2 / 4; }
.item2 { grid-area: 2 / 3 / 3 / 4; }
.item3 { grid-area: 1 / 1 / 2 / 2; }
.item4 { grid-area: 1 / 2 / 2 / 3; }
.item5 { grid-area: 2 / 1 / 3 / 2; }
.item6 { grid-area: 2 / 2 / 3 / 3; }
```

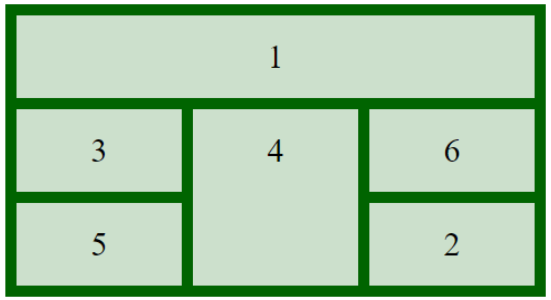
Exemplo_10.html

Usando *Media Queries*

É possível reorganizar a ordem dos elementos para determinados tamanhos de tela, usando *media queries*.

```
<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>
```






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

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

```
@media only screen and (max-width: 700px) {
  .item1 { grid-area: 1 / span 3 / 2 / 4; }
  .item2 { grid-area: 3 / 3 / 4 / 4; }
  .item3 { grid-area: 2 / 1 / 3 / 2; }
  .item4 { grid-area: 2 / 2 / span 2 / 3; }
  .item5 { grid-area: 3 / 1 / 4 / 2; }
  .item6 { grid-area: 2 / 3 / 3 / 4; }
}
```

Exemplo 11.html

Demais propriedades

Propriedade	Descrição
`column-gap`	Especifica a lacuna entre as colunas
`gap`	Abreviação para as propriedades de gap de linha e de coluna
`grid`	Abreviação para as propriedades `grid-template-rows`, `grid-template-columns`, `grid-template-areas`, `grid-auto-rows`, `grid-auto-columns`, e `grid-auto-flow`
`grid-area`	Ou especifica um nome para o *grid item,* ou é uma abreviação para as propriedades `grid-row-start`, `grid-column-start`, `grid-row-end` e `grid-column-end`
`grid-auto-columns`	Especifica um tamanho de coluna padrão
`grid-auto-flow`	Especifica como os itens colocados automaticamente são inseridos na grade
`grid-auto-rows`	Especifica um tamanho de linha padrão
`grid-column`	Abreviação para as propriedades `grid-column-start` e `grid-column-end`
`grid-column-end`	Especifica onde terminar o `grid item`
`grid-column-gap`	Especifica o tamanho da lacuna entre as colunas
`grid-column-start`	Especifica onde iniciar o `grid item`

Demais propriedades

Propriedade	Descrição
`grid-gap`	Abreviação para as propriedades **`grid-row-gap`** e **`grid-column-gap`**
`grid-row`	Abreviação para as propriedades **grid-row-start** e **grid-row-end**
`grid-row-end`	Especifica onde terminar o **`grid item`**
`grid-column-start`	Especifica onde iniciar o **`grid item`**
`grid-row-start`	Especifica onde iniciar o **`grid item`**
`grid-template`	Abreviação para as propriedades **`grid-template-rows`**, **`grid-template-columns`** e **`grid-áreas`**
`grid-template-areas`	Especifica como exibir colunas e linhas, usando **`grid item`** nomeados
`grid-template-columns`	Especifica o tamanho das colunas e quantas colunas em um grid layout
`grid-template-rows`	Especifica o tamanho das linhas em um grid layout
`row-gap`	Especifica a lacuna entre as **`grid rows`**

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS
Grid Layout

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas