IOS – Instituto de Oportunidade Social

CSS 12 - CSS Grid Layout Parte 01

Objetivos

- Compreender a criação de diferentes layouts com o uso do **Grid CSS**;

- Conhecer as diversas **propriedades** do Grid CSS;

- Aplicar os **recursos** do Grid CSS nas folhas de estilo.

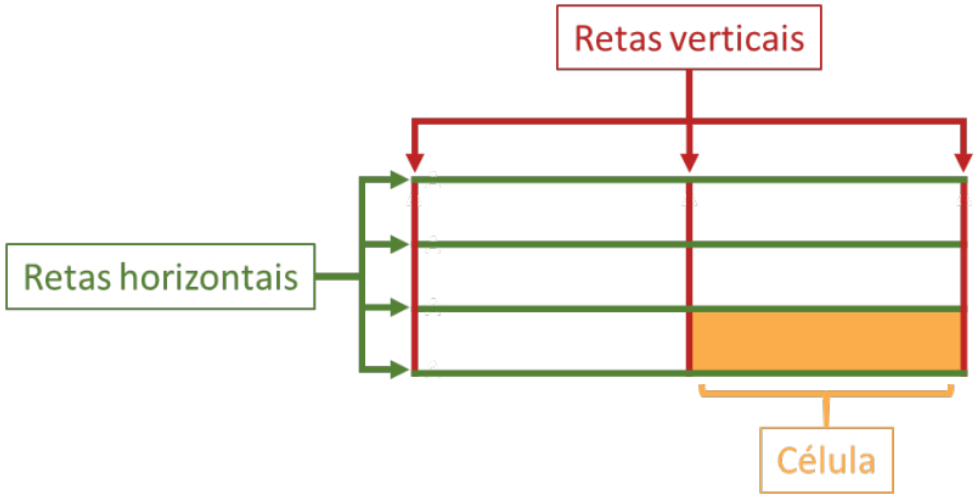
IOS – Instituto de Oportunidade Social

CSS Grid Layout

CSS Grid Layout

O **CSS Grid Layout** é um recurso que permite dividir uma página em regiões principais definindo o **relacionamento em termos de medidas**, **posicionamento e camadas**.

O Grid é um **layout** formado por **retas verticais e horizontais**, que formam as **células do grid**. A imagem abaixo mostra um grid 3x2 (3 linhas e 2 colunas) com três retas verticais e quatro horizontais, totalizando seis células.



Exemplo de grid com reta horizontais e verticais.

CSS Grid Layout

A proposta do Grid é definir o **layout bidimensional** de uma página web criando **itens fixos ou flexíveis** que podem ser configurados individualmente. Como tabelas, o layout Grid permite alinhar elementos em linha e colunas por meio de um método bidimensional para criação de layout utilizando. Os principais termos são:

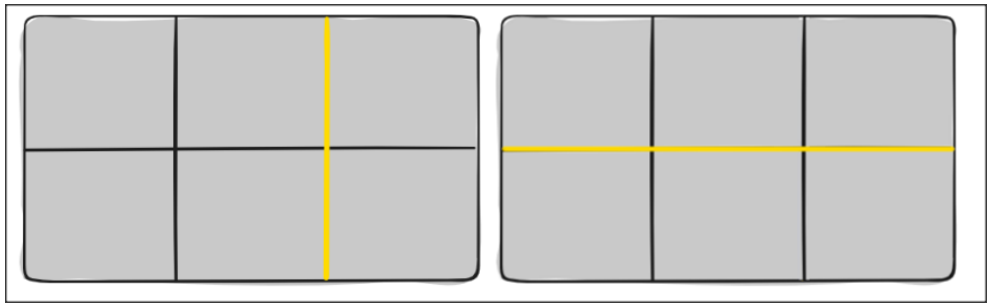
Grid Container: é o elemento ou classe que definimos a propriedade **display: grid**. Então, esse elemento será o pai de todos os itens do Grid. Por exemplo, podemos configurar uma classe **.container** em um elemento e vários itens dentro dessa seção:

```
<section class="container">
        <div class="item item-1"> </div>
        <div class="item item-2"> </div>
        <div class="item item-3"> </div>
</section>
```

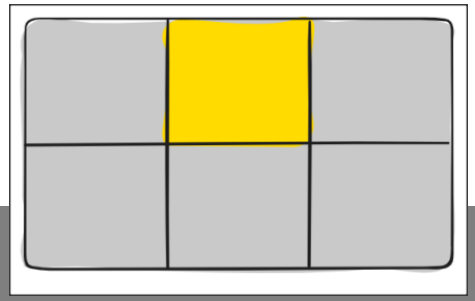
CSS Grid Layout

Grid Item: são os filhos do grid container.

Grid Line: são as linhas divisórias que formam a estrutura do grid. Elas podem ser as retas verticais ou horizontais e estar em qualquer lado da linha ou coluna.

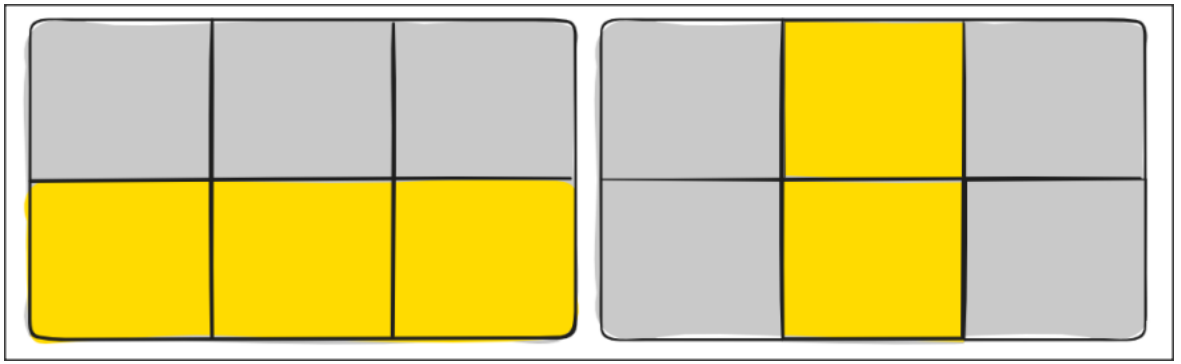


Grid Cell: é a célula do grid (unidade única do grid), ou seja, o espaço entre duas retas horizontais ou verticais adjacentes. Abaixo é mostrado o Grid Cell entre as retas 1 e 2 horizontais e as retas 2 e 3 verticais.

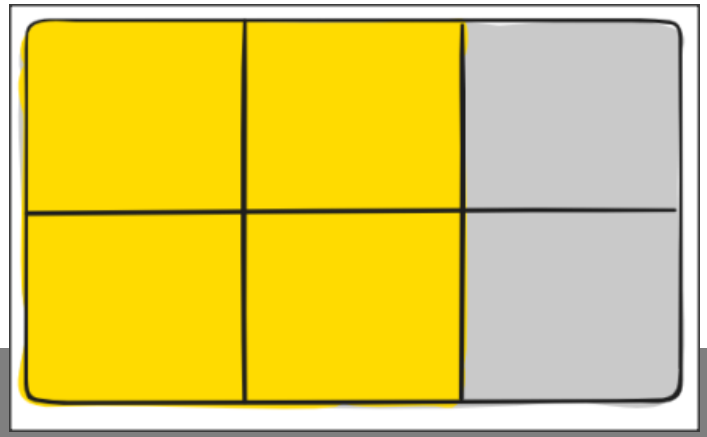


CSS Grid Layout

Grid Track: é uma trilha do grid, que pode ser uma linha ou uma coluna inteira.



Grid Area: O espaço total rodeado por quatro retas do grid. Grid Area pode ser composta de qualquer número de células do grid. Por exemplo, abaixo a Grid Area está entre as retas horizontais 1 e 3 e as retas verticais 1 e 3.



CSS Grid Layout

Propriedade display

A propriedade **display** configura um elemento do HTML como um container grid e determina um novo grid formatando o contexto para o seu conteúdo. Os valores possíveis são:

grid: gera um block grid
inline-grid: gera um grid alinhado

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

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

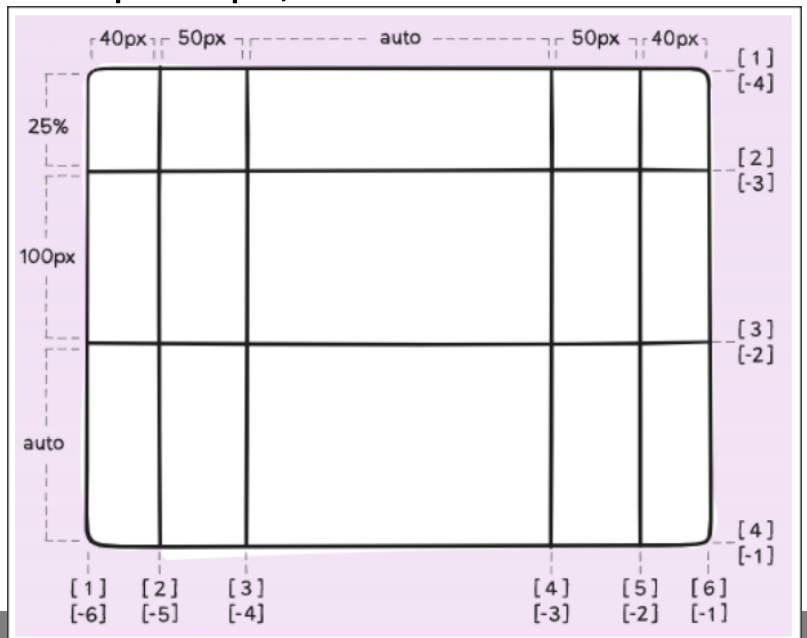
IOS – Instituto de Oportunidade Social

Propriedades grid-template

Propriedade grid-template

As propriedades **grid-template-columns** e **grid-template-rows configura a largura e altura das colunas e linhas do grid**. Você deve colocar os valores desejados separados por espaços e os valores representam o tamanho do **Grid Track** (trilha). Por exemplo:

```
.container {
        grid-template-columns: 40px 50px auto 50px 40px;
        grid-template-rows: 25% 100px auto;
}
```



Resultado das trilhas criadas no Grid.

Propriedade grid-template

Definindo valores de cada trilha:

```
.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]; }
```

Dividindo em partes iguais:

```
.container {
        grid-template-columns: repeat(3, 20px [col-start]); }
.container {
        grid-template-columns: repeat(4, 1fr); }
```

Propriedade grid-template

A propriedade **grid-template-areas** define um modelo de grid referenciando os nomes das áreas do grid que são especificadas pela propriedade **grid-area**. Ex:

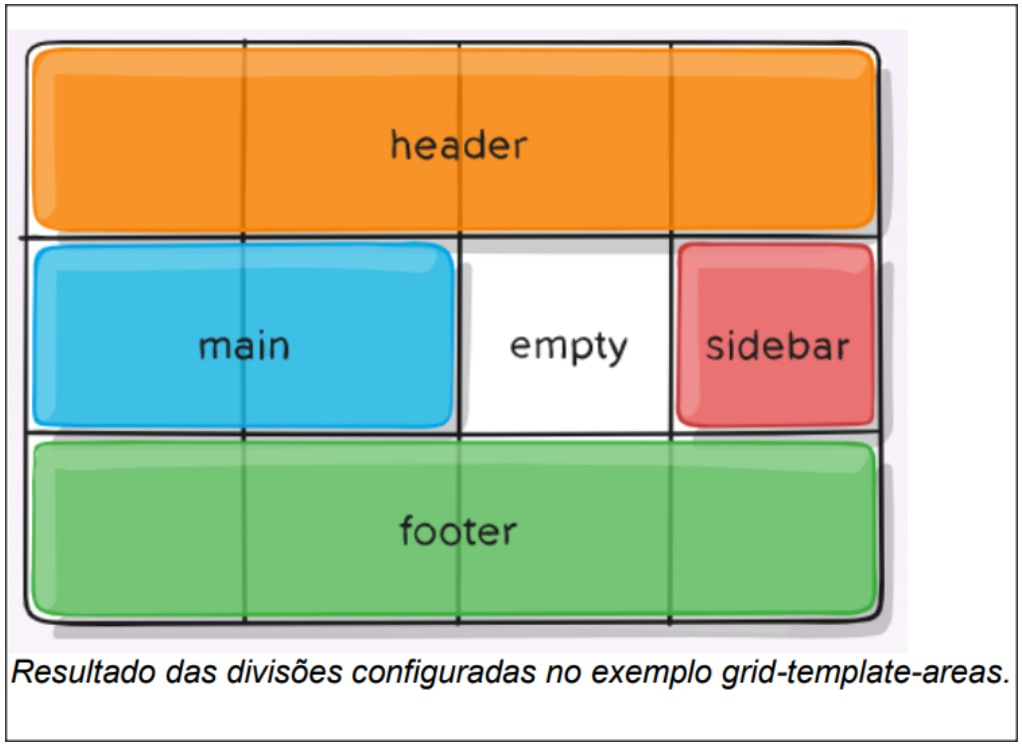
```
.item-a { grid-area: header; }

.item-b { grid-area: main; }

.item-c { grid-area: sidebar; }

.item-d { grid-area: footer; }

.container {
        display: grid;
grid-template-columns: repeat(4, 1fr);
        grid-template-rows: auto;
grid-template-areas: "header header header header" "main main . sidebar" "footer footer footer footer" ; }
```



Resultado das divisões configuradas no exemplo grid-template-areas.

Propriedade grid-template

A propriedade **grid-template** é uma abreviação das propriedades **grid-template-rows, grid-template-columns e grid-template-areas**:

```
.container {
        grid-template: [row1-start] "header header header" 25px [row1-end]
        [row2- start] "footer footer footer" 25px [row2-end] / auto 50px auto;
}
```

Equivalente a:

```
.container {
        grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end];
        grid-template-columns: auto 50px auto;
        grid-template-areas: "header header header" "footer footer footer";
}
```

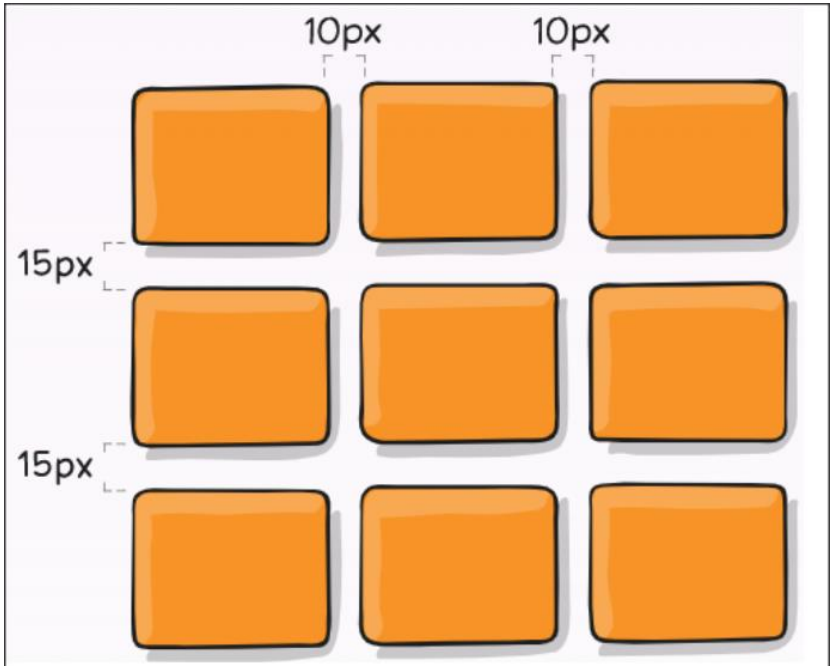
IOS – Instituto de Oportunidade Social

Propriedades gap

Propriedade gap

Propriedades **column-gap, row-gap, grid-column-gap e grid-row-gap**
Elas especificam o **espaçamento** (**gap**) entre as linhas e as colunas do grid, ou seja, a espessura das retas verticais ou horizontais. O prefixo grid- pode ser omitido. Ao invés de usar grid-row-gap, podemos usar somente row-gap. Exemplo:

```
.container {
        grid-template-columns: repeat(3, 1fr);
        grid-template-rows: 80px auto 80px;
        column-gap: 10px;
        row-gap: 15px;
}
```



Resultado do exemplo row0gap e column-gap.

Propriedade gap

Propriedade gap ou grip-gap

A propriedade **grid-gap** é uma abreviação das propriedades **row-gap e column-gap**. Os valores de ser na ordem: primeiro e em segundo . Por exemplo:

```
.container {
        grid-template-columns: repeat(3, 1fr);
        grid-template-rows: 80px auto 80px;
        grid-gap: 15px 10px;
}
```

IOS – Instituto de Oportunidade Social

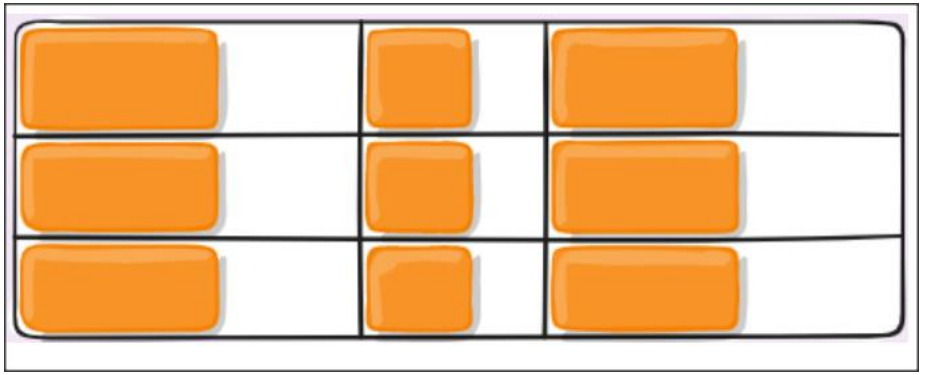
Propriedade justify-items

Propriedade justify-items

A propriedade **justify-items** alinha **horizontalmente** os itens do grid:

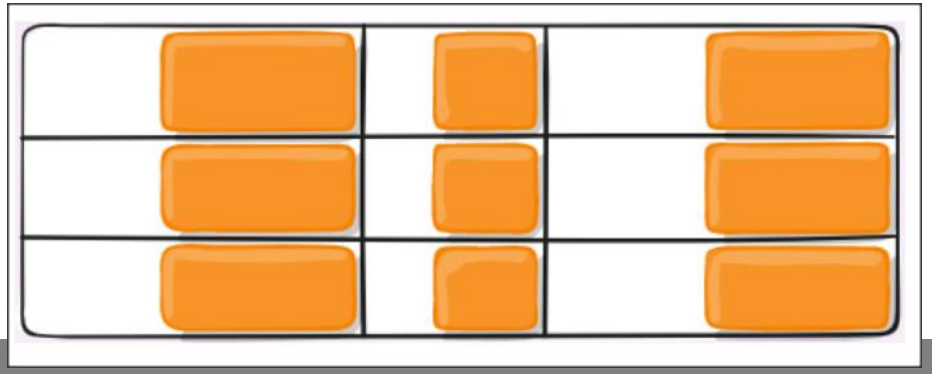
start: os itens ficam alinhados na reta esquerda da célula do grid.

```
.container {
        justify-items: start;
}
```



end: os itens ficam alinhados na reta direita da célula do grid.

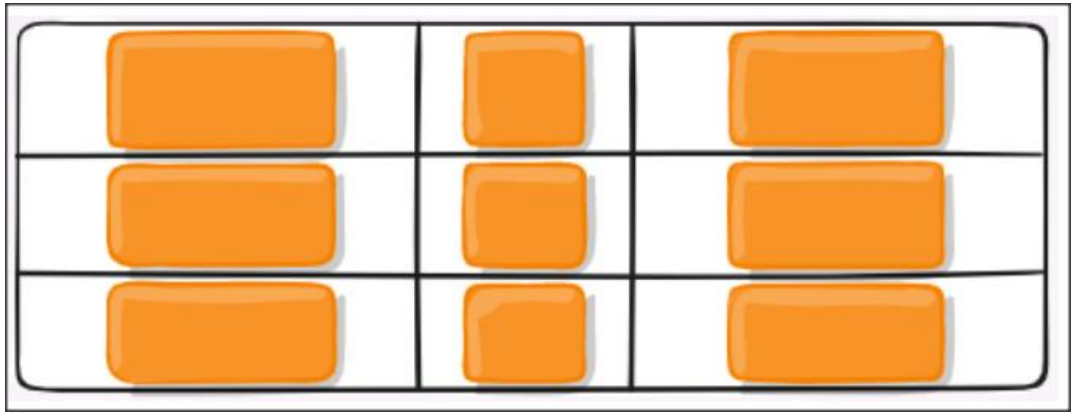
```
.container {
        justify-items: end;
}
```



Propriedade justify-items

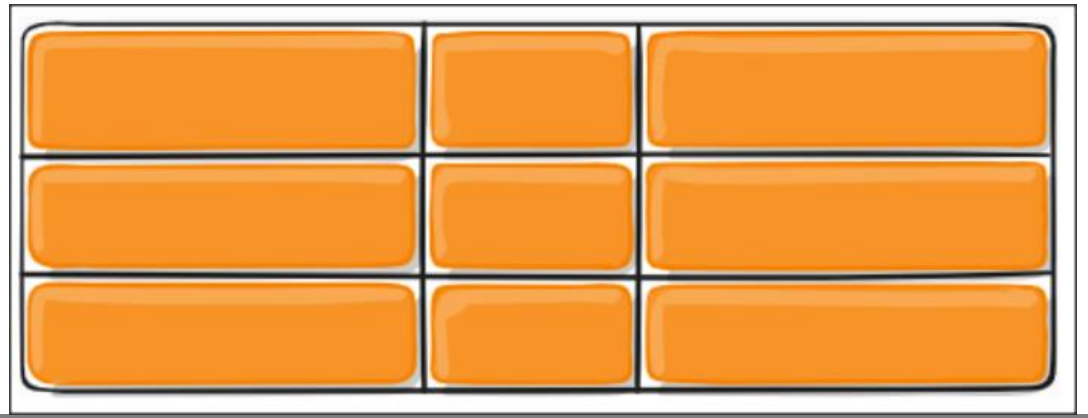
center: os itens ficam centralizados horizontalmente na célula do grid.

```
.container {
        justify-items: center;
}
```



stretch: os itens preenchem todo o espaço da célula do grid.

```
.container {
        justify-items: stretch;
}
```



IOS – Instituto de Oportunidade Social

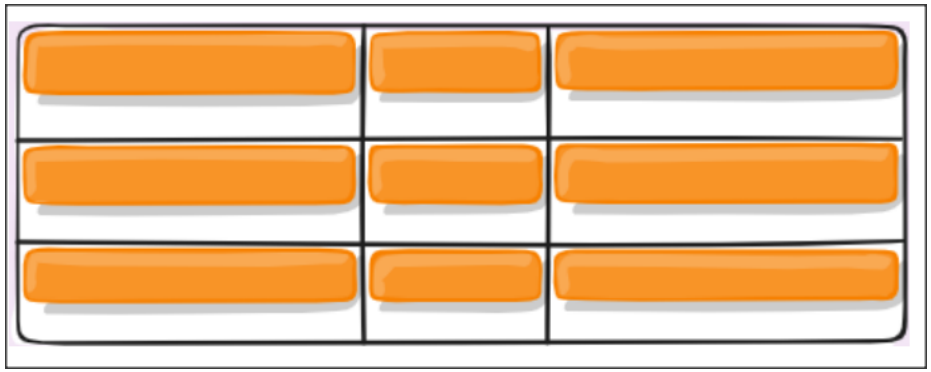
Propriedade align-items

Propriedade align-items

A propriedade **align-items** alinha **verticalmente** os itens do grid:

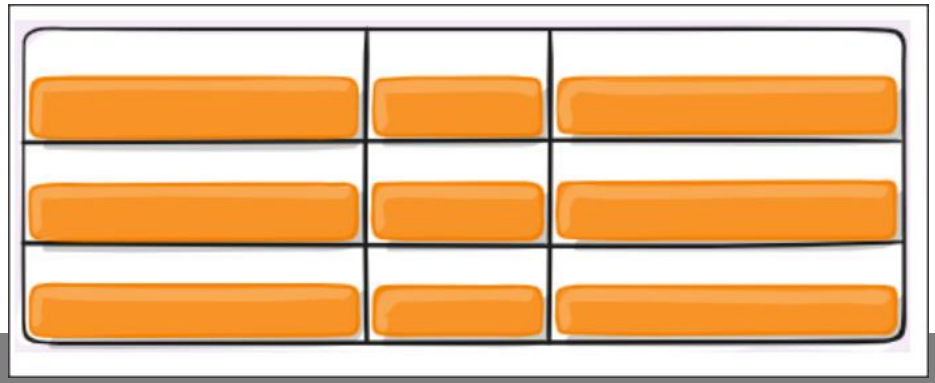
start: os itens ficam alinhados na reta superior da célula do grid.

```
.container {
        align-items: start;
}
```



end: os itens ficam alinhados na reta inferior da célula do grid.

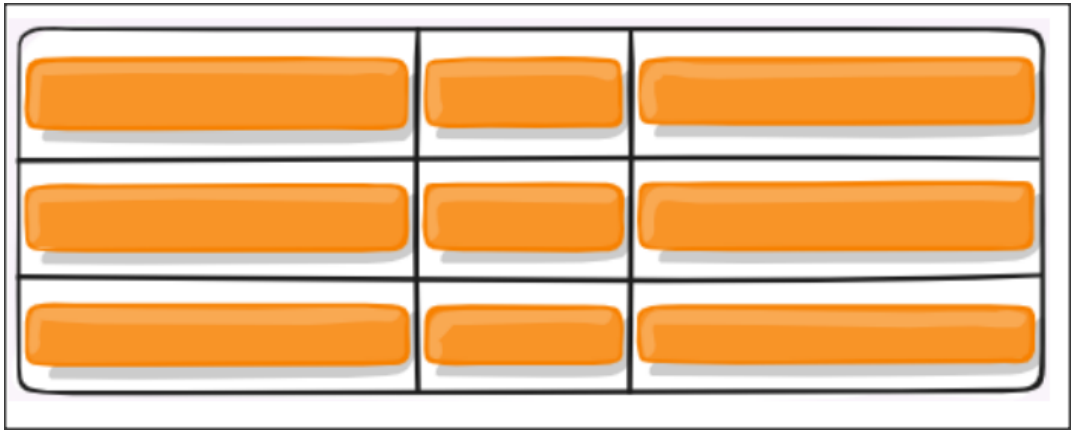
```
.container {
        align-items: end;
}
```



Propriedade align-items

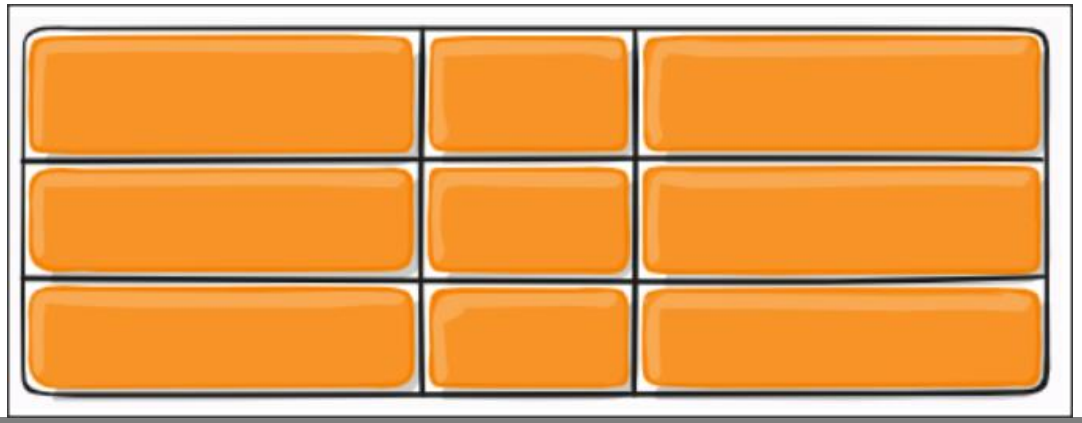
center: os itens ficam centralizados verticalmente na célula do grid.

```
.container {
        align-items: center;
}
```



stretch: os itens preenchem todo o espaço da célula do grid.

```
.container {
        align-items: stretch;
}
```



IOS – Instituto de Oportunidade Social

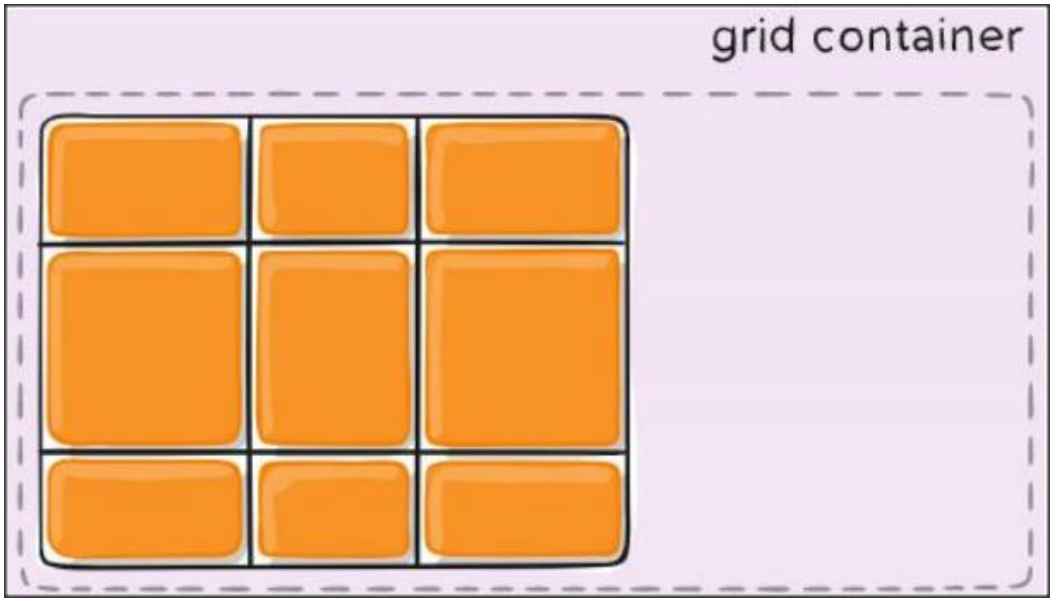
Propriedade justify-content

Propriedade justify-content

A propriedade **justify-content** alinha o grid completo ao longo do **eixo horizontal** do container do grid. Os valores possíveis para configurar essa propriedade são:

start: o grid fica alinhado na reta esquerda do container.

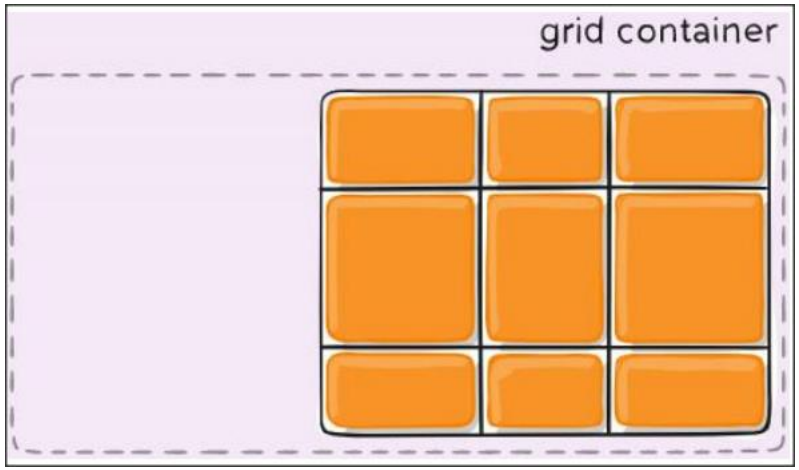
```
.container {
        justify-content: start;
}
```



Propriedade justify-content

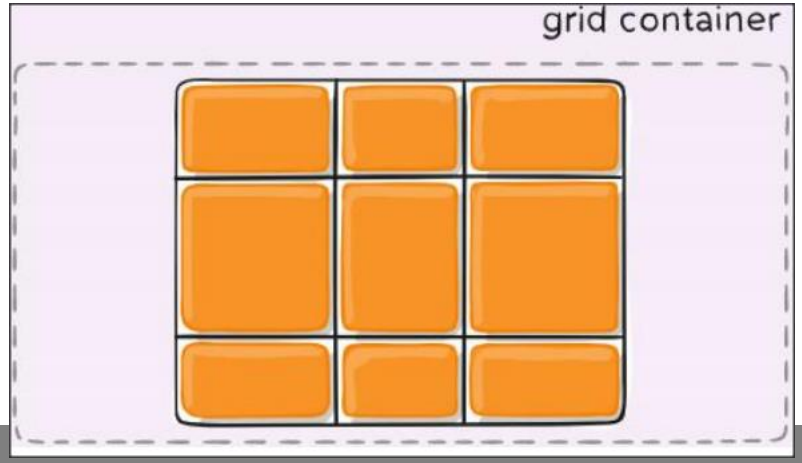
end: o grid fica alinhado na borda inferior do container.

```
.container {
        align-content: end;
}
```



center: o grid fica centralizado verticalmente no container.

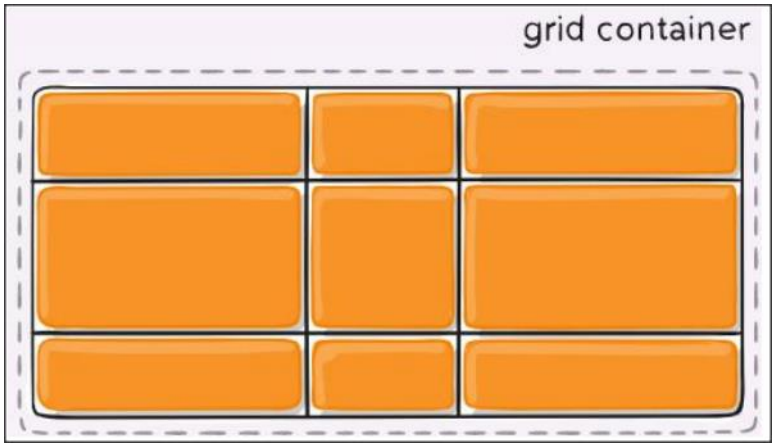
```
.container {
        align-content: center;
}
```



Propriedade justify-content

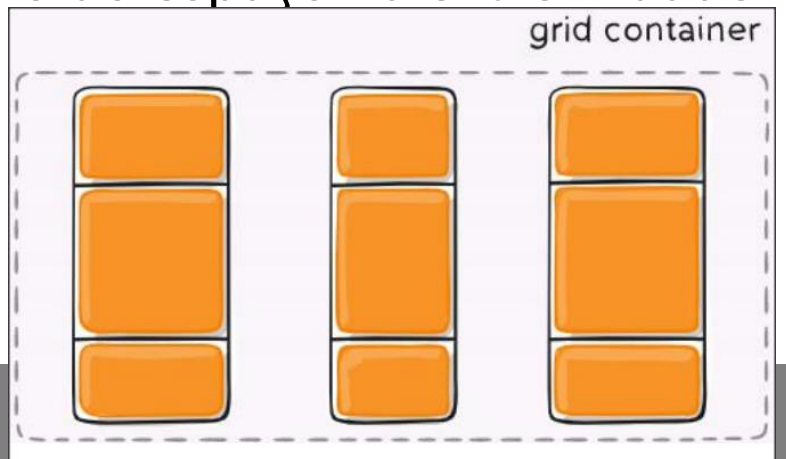
stretch: o grid preenche todo o espaço do container.

```
.container {
        justify-content: stretch;
}
```



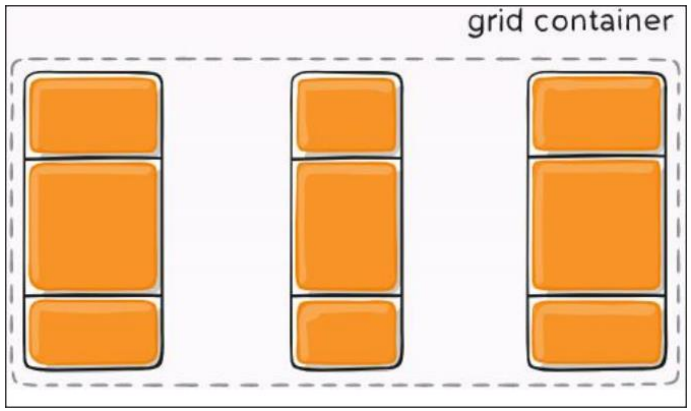
space-around: coloca uma quantidade uniforme de espaço entre cada item do grid e metade do tamanho do espaço na extremidade.

```
.container {
        justify-content: space-around;
}
```



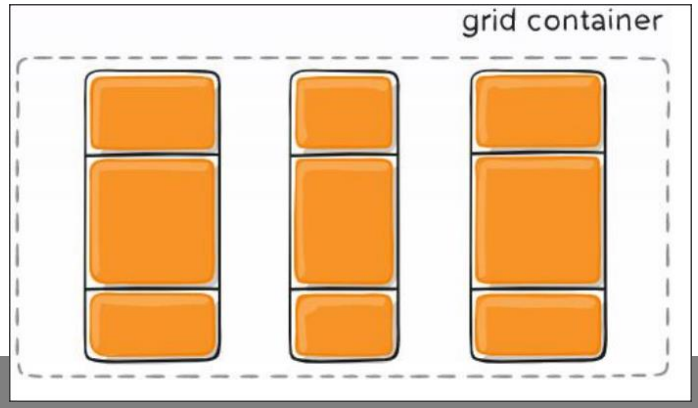
Propriedade justify-content

space-between: coloca uma quantidade uniforme de espaço entre cada item do grid e sem espaço nas extremidades.




```
.container {
        justify-content: space-between;
}
```

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




```
 .container {
        justify-content: space-evenly;
}
```

IOS – Instituto de Oportunidade Social

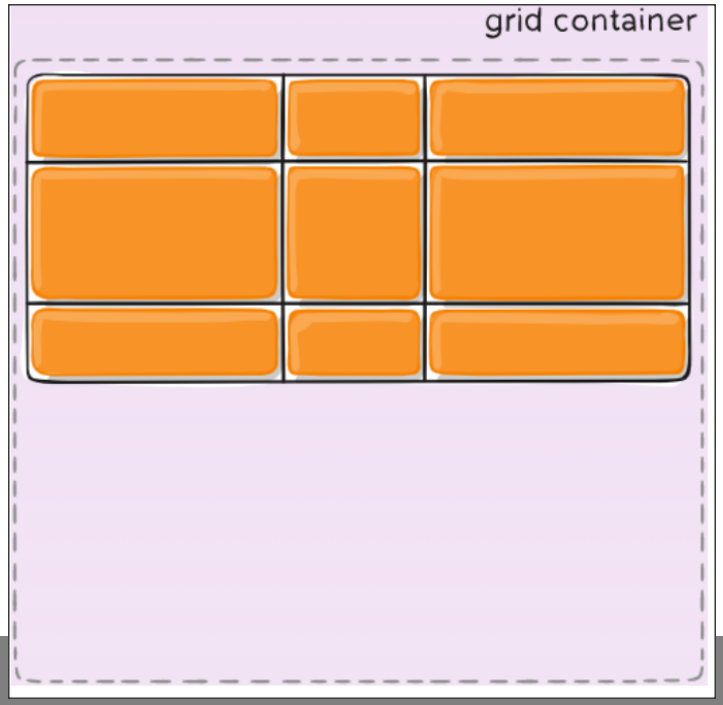
Propriedade align-content

Propriedade align-content

A propriedade **align-content** alinha o grid completo ao longo do **eixo vertical** do container do grid. Os valores possíveis para configurar essa propriedade são:

start: o grid fica alinhado na borda superior do container.

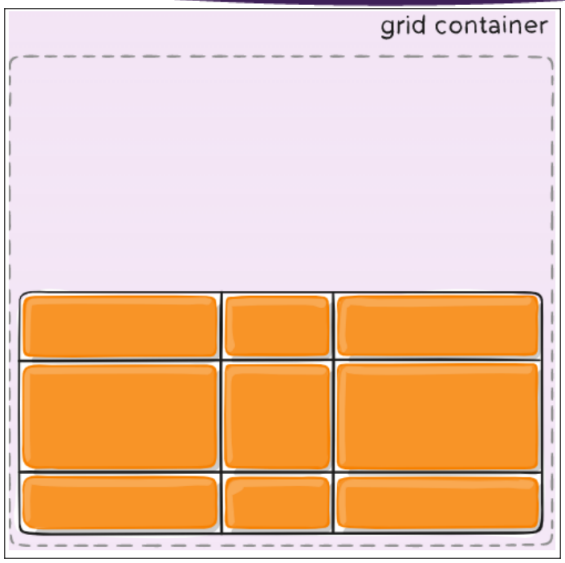
```
.container {
        align-content: start;
}
```



Propriedade align-content

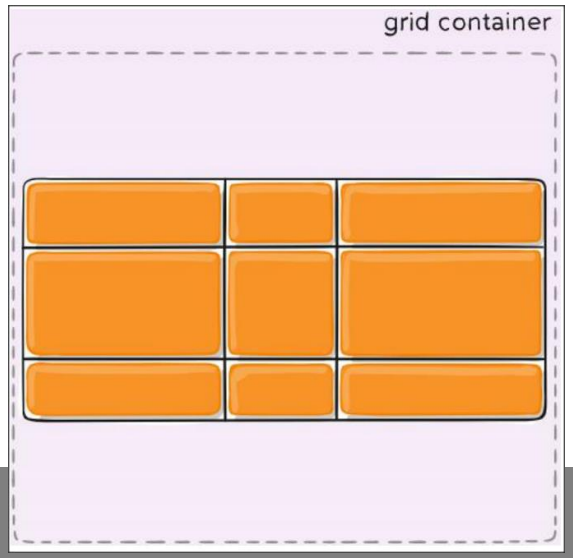
end: o grid fica alinhado na borda inferior do container.

```
.container {
        align-content: end;
}
```



center: o grid fica centralizado verticalmente no container.

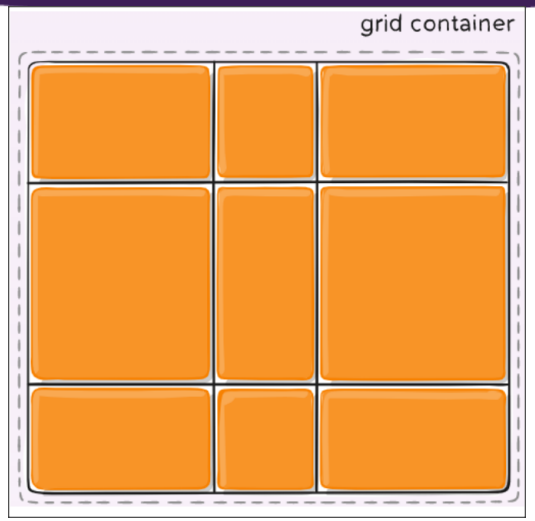
```
.container {
        align-content: center;
}
```



Propriedade align-content

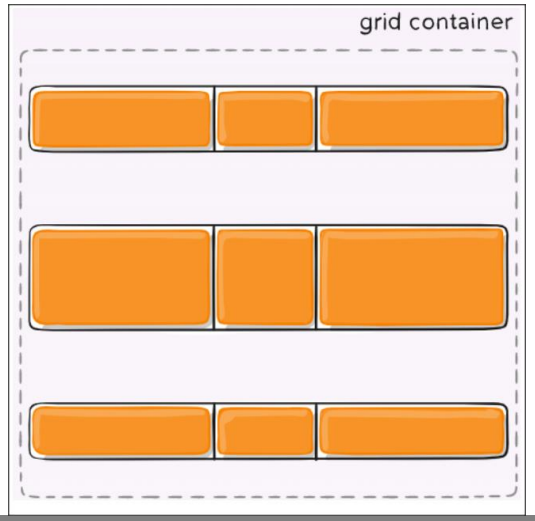
stretch: o grid preenche todo o espaço do container.

```
.container {
        align-content: stretch;
}
```



space-around: coloca um espaço uniforme entre cada item do grid e metade do espaço nas extremidades.

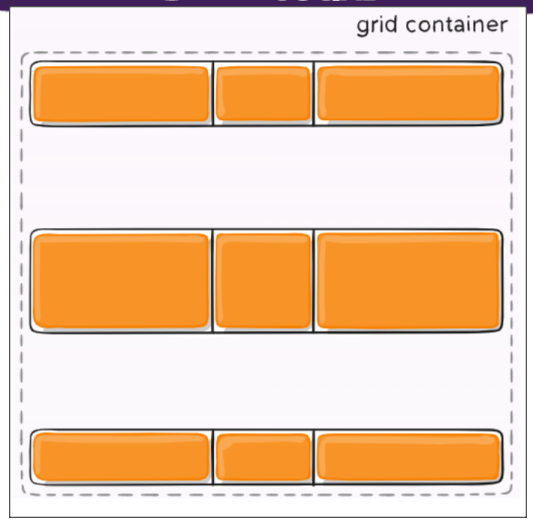
```
.container {
        align-content: space-around;
}
```



Propriedade align-content

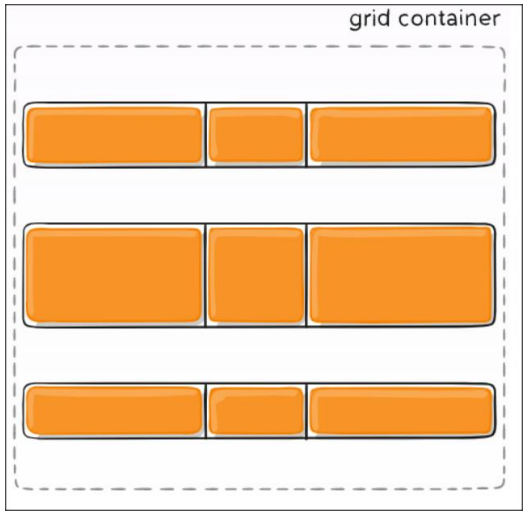
space-between: coloca um espaço uniforme entre cada item do grid e sem espaço nas extremidades.

```
.container {
        align-content: space-between; }
```



space-evenly: coloca um espaço uniforme entre cada item do grid, incluindo as extremidades.

```
.container {
        align-content: space-evenly;
}
```



IOS – Instituto de Oportunidade Social

Propriedades grid-auto

Propriedades grid-auto

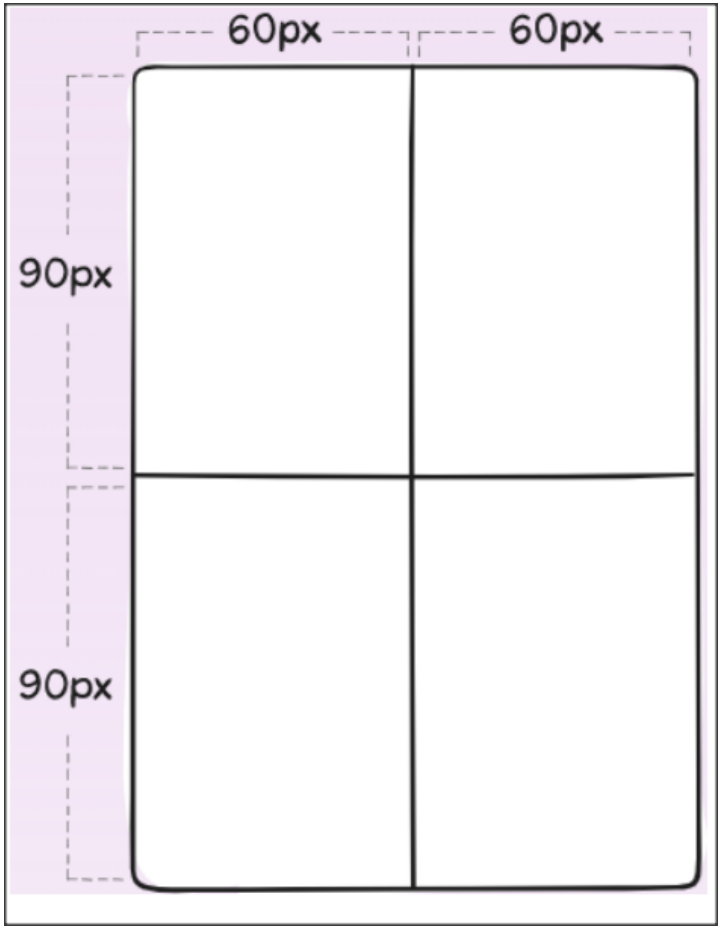
As propriedades **grid-auto-columns** e **grid-auto-rows** especificam o **tamanho** de quaisquer **trilhas** do grids geradas automaticamente, que podem também ser chamadas de **trilha implícitas** do grid. Trilhas implícitas são criadas quando **há mais itens no grid do que células disponíveis** ou quando um **item do grid é colocado fora da grade explícita**. Os valores possíveis para configurar essa propriedade são:

<track-size>: que pode ser um tamanho, uma porcentagem ou uma fração (usando a unidade **fr**) do espaço livre no grid.

Propriedades grid-auto

Para ilustrar melhor, vamos ver como trilhas implícitas são criadas. Vamos incialmente criar um **grid de 2x2**:

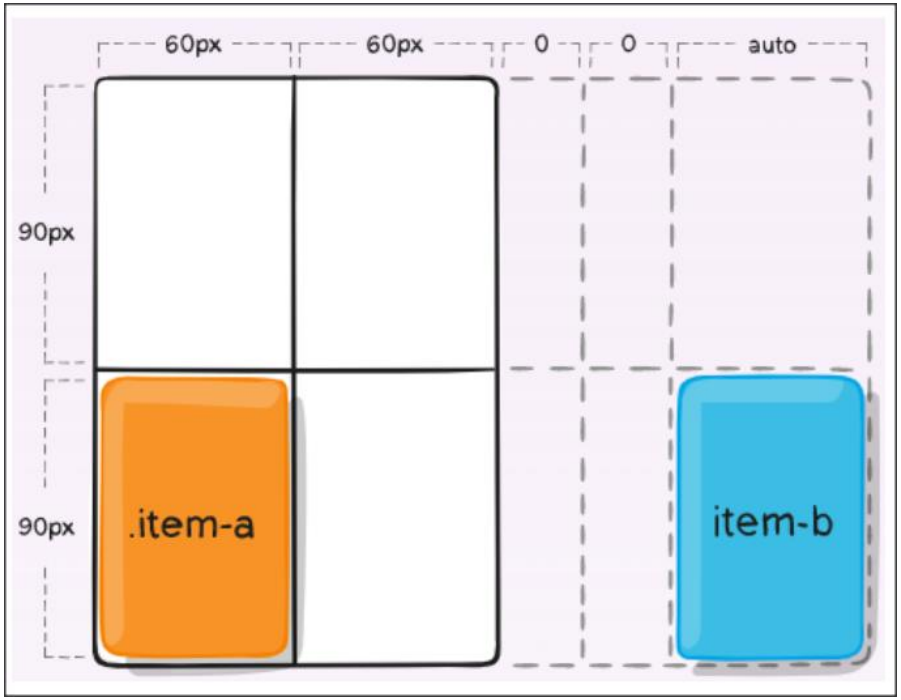
```
.container {
        grid-template-columns: 60px 60px;
        grid-template-rows: 90px 90px;
}
```



Propriedades grid-auto

Agora, imagine que você use as propriedades grid-column e grid-row para posicionar os itens da seguinte forma.

```
.item-a {
        grid-column: 1 / 2;
        grid-row: 2 / 3;
}
.item-b {
        grid-column: 5 / 6;
        grid-row: 2 / 3;
}
```

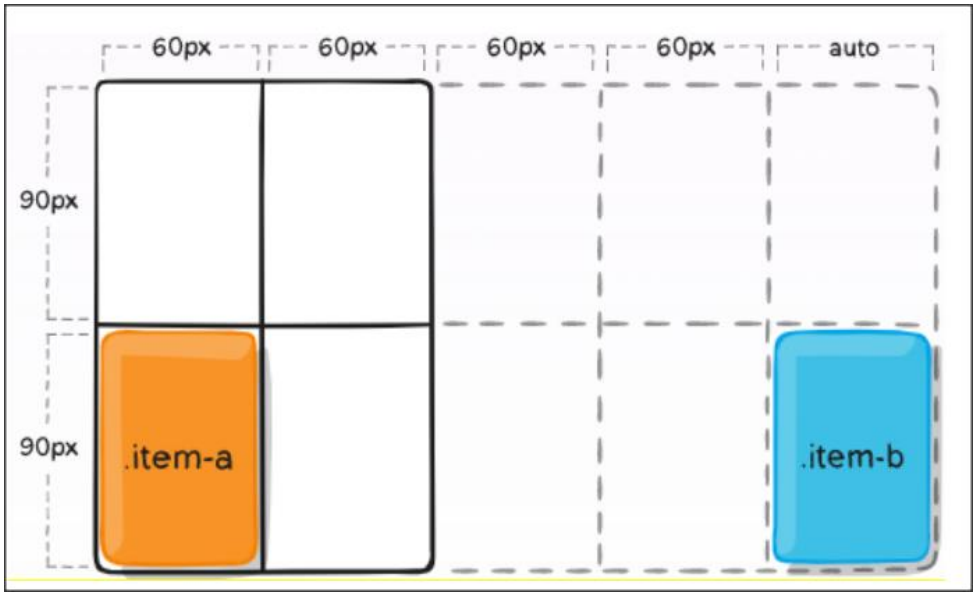


Nesse caso, o **item-b começa na reta da coluna 5 e termina na reta coluna 6**, mas nós **não definimos anteriormente** essa parte (nosso grid é **2x2**). Sendo assim, ao referenciar uma reta não existente teremos **trilhas implícitas** criadas com **tamanho 0** para o gap de trilhas.

Propriedades grid-auto

Caso quisermos, nós podemos mudar a primeira declaração, quando criamos o grid 2x2 na classe .container para que sejam criadas **trilhas implícitas com tamanho especificado**, isso é o que as propriedades grid-auto-columns e grid-auto-rows fazem. Por exemplo:

```
.container {
        grid-auto-columns: 60px;
}
```



IOS – Instituto de Oportunidade Social

Propriedade grid-auto-flow

Propriedade grid-auto-flow

Se você tiver itens do grid que não foram colocados explicitamente no grid, o algoritmo de **auto posicionamento** entra em ação para colocar os itens automaticamente. A propriedade **grid-auto-flow** controla como o algoritmo funciona. Os valores possíveis para configurar essa propriedade são:

row (default): define que o algoritmo de auto posicionamento irá preencher cada linha por vez, adicionando novas linhas se necessário.

column: define que o algoritmo de auto posicionamento irá preencher cada coluna por vez, adicionando novas colunas se necessário.

dense: define que o algoritmo de auto posicionamento use um formato de compactação "denso", ou seja, ele irá tentar preencher primeiro as lacunas no grid se itens menores surgirem.

Propriedade grid-auto-flow

Vamos entender melhor. Suponha que tenhamos o seguinte trecho no HTML:

```
<section class="container">
        <div class="item-a">item-a</div>
        <div class="item-b">item-b</div>
        <div class="item-c">item-c</div>
        <div class="item-d">item-d</div>
        <div class="item-e">item-e</div>
</section>
```

Então, especificamos os pontos para dois desses itens.

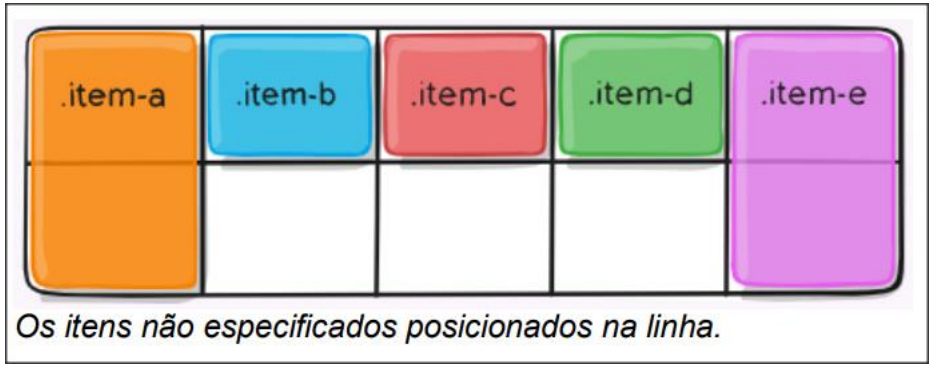
```
.item-a {
        grid-column: 1;
        grid-row: 1 / 3;}

.item-e {
        grid-column: 5;
        grid-row: 1 / 3;}
```

Propriedade grid-auto-flow

E no CSS, tenhamos o grid com cinco colunas e duas linhas e define a propriedade grid-auto-flow com o valor row.

```
.container {
        display: grid;
        grid-template-columns: 60px 60px 60px 60px 60px;
        grid-template-rows: 30px 30px;
        grid-auto-flow: row;
}
```



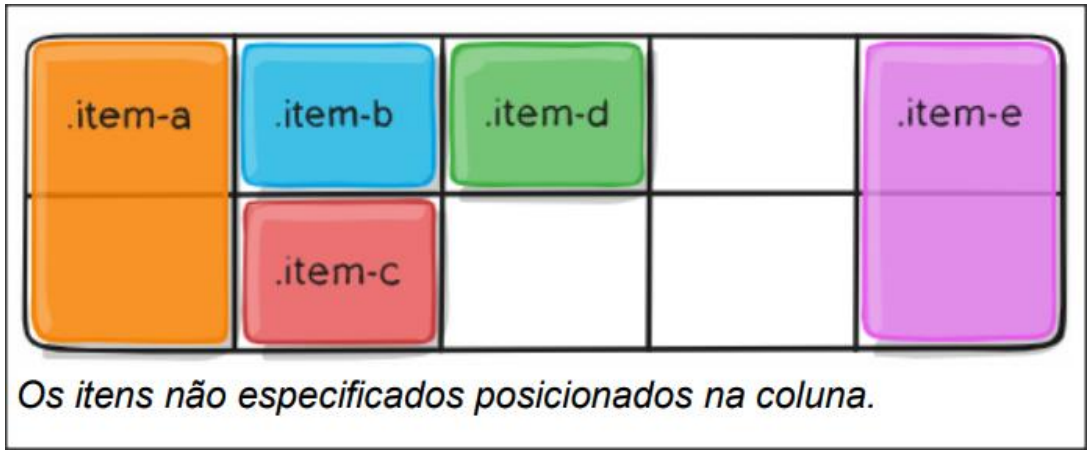
Os itens não especificados posicionados na linha.

Como a propriedade grid-auto-flow foi configurada com o valor row, o nosso grid será exibido com os três itens (item-b, item-c e item-d) no grid, que nós não especificamos, posicionados pela linha.

Propriedade grid-auto-flow

Se ao invés do valor row colocássemos o valor **column**, os itens seriam posicionados nas colunas.

```
.container {
        display: grid;
        grid-template-columns: 60px 60px 60px 60px 60px;
        grid-template-rows: 30px 30px;
        grid-auto-flow: column;
}
```



Os itens não especificados posicionados na coluna.

IOS – Instituto de Oportunidade Social

Vamos Praticar

Vamos Praticar

Apostila de CSS:

- CSS Grid 1 e 2

Páginas 165 a 174

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de Oportunidade Social

Exercícios

Exercícios

Montar uma página HTML **cssGridLayoutP1.html** com seu respectivo par CSS.

- Criar 7 templates de Grid utilizando as propriedades:

1) Utilizar **grid-template**
2) Utilizar **column-gap, row-gap, grid-column-gap e grid-row-gap**
3) Utilizar **justify-items**
4) Utilizar **align-items**
5) Utilizar **justify-content**
6) Utilizar **align-content**
7) Utilizar **grid-auto**